

UNIVERSIDADE PAULISTA - CAMPUS SANTOS-RANGEL

JOÃO PAULO MENEZES SANTOS - N861347

EIKE SANDES SANTOS - G496HG1

ARTHUR MARIANO SOARES - G518276

PEDRO HENRIQUE ANDRADE BORGES DOS SANTOS - G4472B6

JOSÉ SIQUEIRA NETO - N9564H7

KAIQUE MELO NÓBREGA - F348849

RECONHECIMENTO DE IMAGEM

O desenvolvimento de uma aplicação para reconhecimento facial

Santos - SP

2024

JOÃO PAULO MENEZES SANTOS - N861347
EIKE SANDES SANTOS - G496HG1
ARTHUR MARIANO SOARES - G518276
PEDRO HENRIQUE ANDRADE BORGES DOS SANTOS - G4472B6
JOSÉ SIQUEIRA NETO - N9564H7
KAIQUE MELO NÓBREGA - F348849

RECONHECIMENTO DE IMAGEM

O desenvolvimento de uma aplicação para reconhecimento facial

Relatório de Trabalho de Atividades
práticas supervisionadas apresentado no
curso de Ciência da Computação da
Universidade Paulista no Campus
Santos-Rangel para o Professor José
França da disciplina de Processamento de
Imagens e Visão Computacional

Santos - SP

2024

ÍNDICE

1- Objetivo e motivação do trabalho.....	1
2- Introdução.....	2-3
3- Fundamentos das principais técnicas biométricas.....	4-10
3.1- Reconhecimento de Digitais.....	4-5
3.2- Reconhecimento de Orelhas.....	5
3.3- Reconhecimento de Voz.....	6
3.4- Reconhecimento Facial.....	6-7
3.5- Reconhecimento por Iris.....	7-8
3.6- Reconhecimento por Digitação.....	8-9
3.7- Reconhecimento por Retina.....	9-10
4- Plano de desenvolvimento da aplicação.....	11-15
4.1- Linguagem de Programação.....	11-12
4.2- IDE.....	12-13
4.3- Banco de Dados.....	13
4.4- SGBD.....	14
4.4- Editor do Documento.....	15
5- Projeto e estrutura do programa.....	16-18
5.1- Interface.....	16-17
5.2- Reconhecimento de Imagem.....	17
5.3- Banco de Dados.....	18
6- Relatório com as linhas de código.....	19-28
7- Software em Funcionamento.....	29-32
REFERÊNCIAS.....	33-35
FICHA DA APS.....	36

OBJETIVO E MOTIVAÇÃO DO TRABALHO

Para o desenvolvimento das **APS (Atividades Práticas Supervisionadas)** deste semestre foi solicitado de nós alunos no Documento apresentando o tema da APS neste semestre (2024, p. 2) desenvolver um sistema pensando no seguinte cenário:

“Pede-se aos alunos que desenvolvam em Java ou C# uma ferramenta de identificação e autenticação biométrica que restrinja o acesso a uma rede com banco de dados do Ministério do Meio Ambiente. As informações são estratégicas sobre as propriedades rurais que utilizam agrotóxicos proibidos por causarem grandes impactos nos lençóis freáticos, rios e mares. As informações de nível 1 todos podem ter acesso; as de nível 2 são restritas aos diretores de divisões; as de nível 3 somente são acessadas pelo ministro do meio ambiente.”

Apesar de o documento enviado oficialmente restringir as escolhas de Linguagem de Programação apenas para **Java** e **C#**, o professor deu a nós alunos a liberdade para o uso de uma terceira opção, neste caso o **Python**, que está sendo atualmente utilizado para o desenvolvimento dos códigos de manipulação de imagem usados em aula na disciplina de **PIVC (Processamento de Imagens e Visão Computacional)**, assim expandindo o nosso leque de opções para o desenvolvimento do software e também para a escolha de bibliotecas que serão utilizadas no projeto, com nós tendo que trabalhar com o carregamento de imagens e vídeos para análise pelo sistema, também nos dando a opção de caso nós quisermos, nós possamos trabalhar com leitura em tempo real por meio de scanners ou de uma câmera. Além do desenvolvimento do software solicitado anteriormente, também foi exigido de nós alunos dissertar sobre todos os elementos utilizados para o desenvolvimento do projeto, sobre como o desenvolvimento do sistema afetou a nossa formação e sobre como ele demonstra traços de interdisciplinaridade mesmo que esteja ligado com a disciplina de PIVC antes mencionada. O nível de refinamento, detalhe, funcionalidade, tratamento de erros, funções extras não solicitadas e os relatórios adicionais implementados no sistema têm efeito direto na nota que será dada para o projeto, esta que será atribuída às Atividades Práticas Supervisionadas.

INTRODUÇÃO

No mercado de tecnologia dos dias modernos e em todos os ramos empresariais da modernidade, uma das principais preocupações por parte das empresas e dos empresários é com certeza a segurança de seus dados, sistemas e infraestrutura, já que, o menor dos problemas, é capaz de paralisar as operações de um prédio empresarial inteiro, entre as várias soluções de segurança digital existem entre elas a criptografia dos dados, softwares anti-malware para defesa dos computadores que integram a infraestrutura da empresa de sofrer problemas com Vírus e outros tipos de softwares maliciosos, e é claro, **reconhecimento facial** e **biométrico** para o acesso de dados importantes para a corporação, com o último sendo o que será tratado neste trabalho.



(Imagem meramente ilustrativa de como seria um reconhecimento facial no futuro e suas capacidades. Disponível em:

<<https://www.hagana.com.br/wp-content/uploads/2021/05/biometria-facial-878x440.jpg>>)

Atualmente, o reconhecimento facial é uma tecnologia extremamente difundida, sendo utilizado em diversas situações tanto no dia-a-dia quanto na situação empresarial, sendo uma tecnologia que permite não só o aumento da segurança, exigindo que apenas aquele que está autorizado a acessar algo e apenas por meio de seu rosto consiga fazer a manipulação, leitura e ter no geral o acesso a essas informações, e além disso, permite acima de tudo, maior conveniência quando se trata de aplicação de segurança para os sistemas, pois, é uma opção mais rápida e fácil para autenticar um usuário de um sistema do que por exemplo o preenchimento de um PIN ou Senha.

As utilidades da biometria facial são diversas, entre elas por exemplo, proteger um celular de ser acessado sem autorização do dono por meio de colocar um bloqueio de acesso tendo a biometria facial como forma de desbloqueio; ela

também pode ser utilizada no reforço da lei seja para encontrar criminosos foragidos que estão vagando pelas ruas, seja para encontrar pessoas desaparecidas qual as forças de segurança estão apoiando a procura; também é usado no controle de aeroportos para facilitar o processo de verificação da documentação dos passageiros dos aviões e é claro, reforçar a segurança dos funcionários e passageiros, o que se tornou uma prioridade principalmente após os atentados acontecidos em aviões e aeroportos desde o infeliz 11 de Setembro de 2001, ocorrido nos Estados Unidos da América; Marketing e Publicidade também são áreas onde o reconhecimento facial demonstra grande utilidade graças ao fato de que ela pode ser utilizada para aprimorar a experiência do consumidor; As empresas também podem utilizar o reconhecimento facial nos seus setores de RH para facilitar a gestão de pessoal que trabalha na empresa, permitindo por exemplo uma forma mais rápida de bater o ponto eletrônico e identificar os horários em que o funcionário esteve dentro da empresa; A tecnologia de reconhecimento facial também é importante para o uso por serviços de saúde, Os prestadores de serviços de saúde estão testando o uso de identificação facial para acessar registros dos pacientes, aumentar a velocidade do registro de pacientes, detectar emoções e dores nos pacientes e identificar patologias geneticamente específicas. AICure desenvolveu um software que usa reconhecimento facial para garantir que as pessoas tomem seu medicamento conforme prescrito pelos médicos, como a tecnologia biométrica ficou mais barata e acessível do que em tempos anteriores, estima-se que a adoção no setor de saúde aumente; Como último exemplo também é válido mencionar a relevância da identificação facial para as transações bancárias, permitindo que os clientes consigam suprir suas necessidades no banco sem a necessidade de uso de senhas de uso único ou qualquer outro tipo de senhas e/ou códigos.

A tecnologia de reconhecimento facial, uma conquista notável no campo da inteligência artificial e biometria, revolucionou a maneira como identificamos pessoas e protegemos dados. Utilizando IA e aprendizado de máquina, o software de reconhecimento facial teve uma adoção generalizada em vários setores. Por exemplo, agências de aplicação da lei usam tecnologia de reconhecimento facial para identificar suspeitos, comparando suas imagens com um banco de dados de registros criminais. Além disso, a tecnologia se estende ao uso diário, como visto em smartphones e sistemas de segurança, onde permite que os usuários desbloqueiem seus dispositivos por meio de detecção e reconhecimento facial, uma alternativa mais intuitiva e segura às senhas tradicionais ou reconhecimento de voz.

FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS

Existem hoje em dia diversas técnicas biométricas diferentes para serem utilizadas no meio da tecnologia, entre elas podem ser listadas **7 (sete)** que serão mencionadas neste texto, mas antes de qualquer coisa, deve-se fazer a si mesmo a seguinte pergunta: “O que é biometria?” De forma resumida, biometria se trata da aplicação de métricas a características e informações de origem biológica, para fins de estudo e identificação de uma pessoa. A biometria tem como função controlar o acesso físico de pessoas a certos setores e salas, identificar e localizar possíveis criminosos, e impedir que pessoas sem autorização acessem digitalmente dados mantidos sob sigilo, protegidos pelos autores ou aqueles que os mantêm.

RECONHECIMENTO DE DIGITAIS



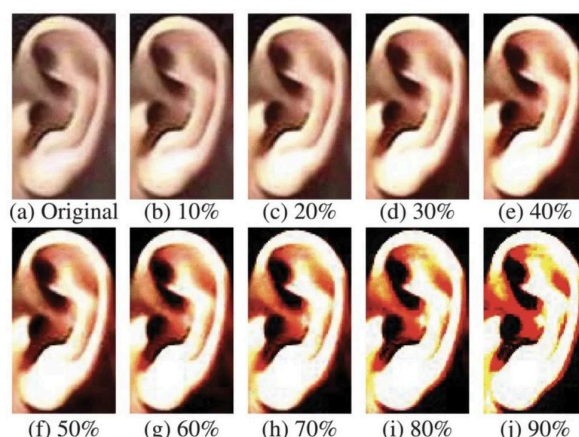
(Representação ilustrativa da ideia do reconhecimento de digitais. Disponível em:
<<https://files.tecnoblog.net/wp-content/uploads/2019/01/thedigitalway-fingerprint-pixabay-700x473.jpg>
>)

O reconhecimento biométrico por meio de **impressão digital** é o mais antigo e de menor custo para implementação. É extremamente confiável, dada a praticamente inexistente mutabilidade dos dados ao longo do tempo de vida do usuário. Migrou de forma suave dos meios analógicos para o digital, sendo utilizado já há muito tempo por diversas empresas em diversas situações e necessidades. Por digitais se manterem as mesmas por praticamente toda a vida, a única possibilidade do sistema apresentar problemas é no caso da pessoa perder as suas digitais, independente do motivo ou origem da perda. Por isso mesmo, o método continua a ser utilizado sozinho ou combinado com outros. A impressão digital é um dos métodos mais confiáveis e econômicos de identificação. Estudos indicam que a identificação por meio das linhas das pontas dos dedos, conhecida como digital, precede qualquer tecnologia de digitalização desse processo. Já no ano **800 D.C.**, na dinastia Tang na China, as digitais eram registradas em tábuas de barro para auxiliar em investigações criminais e no registro de compradores. No Brasil, a cédula de identidade utiliza esse método há gerações.

A segurança dessa técnica advém da estabilidade dos padrões digitais ao longo do tempo, sendo únicos para cada indivíduo. A coleta é simples e não

invasiva, podendo ser realizada sem contato físico, conhecida como biometria touchless. A diferenciação das digitais ocorre pela análise das minúcias, que são pequenas irregularidades nos dedos, formando o desenho dactiloscópico, utilizado para identificar impressões digitais. Embora seja um método único e seguro, algumas pessoas têm dificuldades em ser identificadas por essa biometria devido à falta de características distintas nas linhas dos dedos, causadas por fatores naturais ou externos, como cicatrizes ou desgaste por produtos químicos. Diante dessas situações, torna-se essencial o desenvolvimento de outros métodos biométricos para garantir a identificação de todos os indivíduos.

RECONHECIMENTO DE ORELHAS



(Representação do reconhecimento de orelhas. Disponível em:

<[https://s2-techtudo.glbimg.com/tGgsQgHyXUabzuidGte4bNQPrJI=/0x0:1024x758/984x0/smart/filters:strip_icc\(\)/i.s3.glbimg.com/v1/AUTH_08fbf48bc0524877943fe86e43087e7a/internal_photos/bs/2022/4/B/E87xUJSyiSAvGW9ztnwA/whatsapp-image-2022-12-13-at-17.03.49-1-.jpeg](https://s2-techtudo.glbimg.com/tGgsQgHyXUabzuidGte4bNQPrJI=/0x0:1024x758/984x0/smart/filters:strip_icc()/i.s3.glbimg.com/v1/AUTH_08fbf48bc0524877943fe86e43087e7a/internal_photos/bs/2022/4/B/E87xUJSyiSAvGW9ztnwA/whatsapp-image-2022-12-13-at-17.03.49-1-.jpeg)>)

O **reconhecimento de orelhas** tem raízes fascinantes. A ideia de usar a forma das orelhas como meio de identificação surgiu da necessidade de encontrar métodos de segurança biométrica que fossem eficazes, mas também menos invasivos e suscetíveis a falhas. Estudos mostraram que, assim como as impressões digitais, cada orelha é única. Essa peculiaridade, juntamente com a capacidade de serem fotografadas sem muito contato físico, fez das orelhas uma candidata promissora para sistemas de reconhecimento biométrico. Além disso, a importância desse método é crescente. Num mundo cada vez mais preocupado com segurança e privacidade, métodos como o reconhecimento facial podem enfrentar resistência devido à privacidade e variabilidade de expressões. O reconhecimento de orelhas, com sua menor variabilidade e alta precisão, se torna uma alternativa sólida.

Uma das áreas mais significativas de aplicação é na identificação de recém-nascidos. Como os traços faciais dos bebês mudam rapidamente, as orelhas, que mantêm características mais estáveis, podem ser usadas para garantir que os registros médicos e de identificação sejam precisos desde os primeiros dias de vida.

RECONHECIMENTO DE VOZ



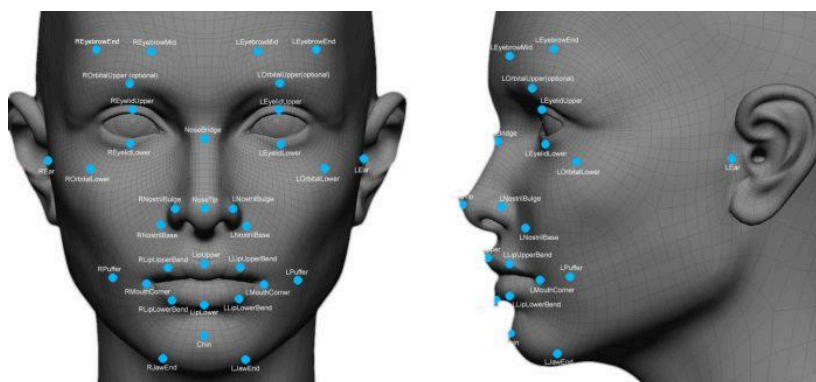
(Representação do reconhecimento de voz. Disponível em:

<<https://istoedinheiro.com.br/wp-content/uploads/sites/17/2019/05/din1121-mercado3.jpg?x46096>>)

Assim como o reconhecimento de orelhas, o **reconhecimento de voz** é uma tecnologia fascinante que está ganhando cada vez mais espaço. Enquanto o reconhecimento de orelhas se destaca pela estabilidade e uniformidade das características, o reconhecimento de voz traz um toque humano único. Cada pessoa tem um perfil sonoro exclusivo, formado pela combinação de parâmetros físicos como as cordas vocais e a laringe, e comportamentais, como sotaques e entonação.

O método de reconhecimento por voz faz uma análise desses parâmetros, resultando em uma "assinatura" biométrica única. Com um custo de implementação relativamente baixo, ele ainda enfrenta desafios significativos, como interferências de ruído no ambiente, mudanças na voz devido a problemas de saúde ou envelhecimento. Apesar dessas barreiras, o reconhecimento de voz continua a evoluir, mostrando-se uma ferramenta promissora para complementar outras formas de biometria e aumentar a precisão na identificação.

RECONHECIMENTO FACIAL



(Representação do reconhecimento facial. Disponível em:

<<https://files.tecnoblog.net/wp-content/uploads/2019/01/fbi-facial-recognition-700x323.jpg>>)

Tecnologia esta que será utilizada para o desenvolvimento do nosso projeto, a tecnologia de **reconhecimento facial** é uma conquista notável no campo da

inteligência artificial (IA) e biometria, tendo revolucionado a forma como identificamos pessoas e garantimos a segurança de dados importantes.

O mecanismo da tecnologia de reconhecimento facial é um misto entre visão computacional, aprendizado de máquina e inteligência artificial. O sistema de reconhecimento facial captura uma imagem ou vídeo contendo rostos de diferentes pessoas. O software então detecta e isola cada rosto presente na imagem. Esse processo envolve identificar vários atributos faciais chave, como os olhos, nariz, boca e o contorno da mandíbula, cruciais para distinguir uma pessoa da outra não só para a IA mas também para nós seres humanos.

Uma vez que essas características faciais são identificadas, o software de reconhecimento facial mapeia o rosto. Isso é feito por meio da análise de vários aspectos do rosto, por exemplo, a distância entre os dois olhos, o quão profundo são as cavidades oculares, o formato das maçãs do rosto e o contorno das orelhas e do queixo. Essas medições são convertidas em uma representação digital, uma assinatura facial única para cada indivíduo.

RECONHECIMENTO POR IRIS



(Representação do reconhecimento por íris. Disponível em:

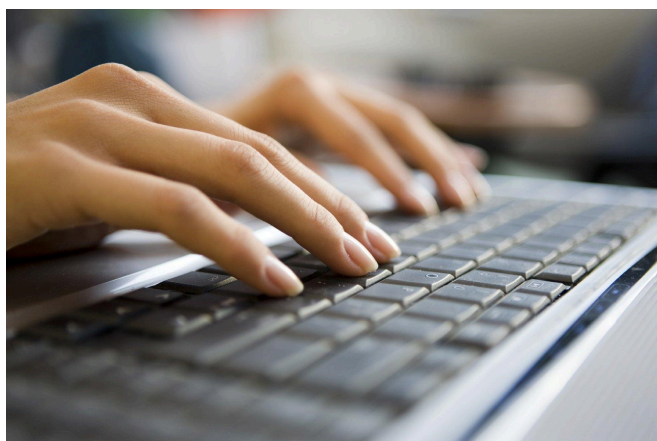
<<https://files.tecnoblog.net/wp-content/uploads/2019/01/tbit-iris-scanning-pixabay-700x422.jpg>>)

Outra das várias outras formas de tecnologia para reconhecimento biométrico é a o **reconhecimento por íris**, qual assim como o nome indica tem como objeto de reconhecimento a íris do olho (a parte colorida do mesmo), o reconhecimento por íris é extremamente confiável, já que a membrana permanece a mesma a vida inteira. Diferente do método de leitura da retina (que será posteriormente mencionado), ele é bem menos invasivo e oferece grande confiabilidade a quem o utiliza, além de ser bastante difícil de contornar.

Entretanto, a implementação do método tem grande custo monetário para quem desejar a utilizar, embora cogita-se que a leitura de íris será o método de biometria mais usado a médio prazo, superando a impressão digital. Muitos celulares, em especial modelos **high-end**, trazem um scanner de íris já embutido na câmera que vêm com o telefone, oferecendo tal opção de segurança para proteger os dados do usuário do aparelho.

O reconhecimento da íris trabalha em 4 partes: com a primeira sendo a aquisição da imagem da íris, a segunda sendo a segmentação da íris, a terceira sendo a análise e representação da textura da íris e, por fim, a comparação das representações da íris. A etapa de aquisição da imagem da íris não é nada mais nada menos do que simplesmente tirar uma fotografia de qualidade da íris, porém, apesar de parecer simples, é um grande desafio capturar uma imagem de alta qualidade do olho sem ser invasivo ou agressivo. A segmentação da íris se trata de localizar a região que a íris ocupa na imagem capturada, não se pode esperar que a captura contenha somente a imagem de uma íris completa e perfeita nesta parte. A etapa de análise e representação da íris pode ser dividida em duas subetapas menores: a normalização e a codificação, A normalização é responsável por transformar a área da íris identificada para permitir a extração de informações relevantes e a realização de comparações de dados, a codificação é a subetapa em que a íris é processada pelo sistema para que seu modelo biométrico seja criado. Na quarta e última etapa, a comparação das representações da íris, o reconhecimento de íris acontece.

RECONHECIMENTO POR DIGITAÇÃO



(Pessoa digitando em um laptop. Disponível em:

<<https://files.tecnoblog.net/wp-content/uploads/2019/01/picjumbo-com-woman-typing-pixabay-700x466.jpg>>)

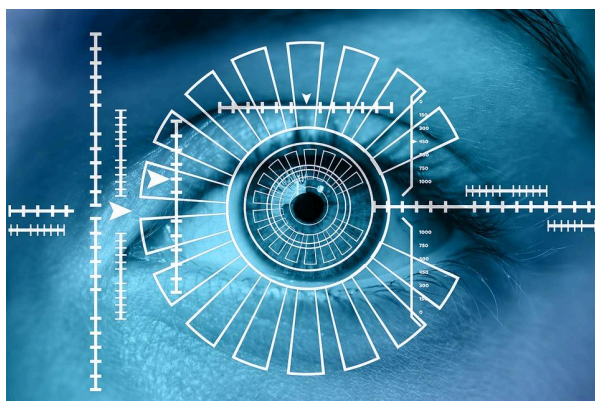
O **reconhecimento por digitação** é um processo definido por acadêmicos e pela indústria para medir e avaliar o ritmo de digitação em dispositivos digitais, como teclados de computador, celulares e painéis sensíveis ao toque.

Uma forma notável de medição de digitação é o reconhecimento de pressionamento de tecla, frequentemente referido como **"dinâmica de pressionamento de tecla"**. Esta técnica utiliza informações temporais detalhadas que descrevem precisamente quando cada tecla é pressionada e liberada durante a digitação. Embora a biometria geralmente depende de características físicas, como impressões digitais ou faciais, ou comportamentais, a dinâmica de pressionamento de tecla é muitas vezes considerada uma forma de biometria.

O **Biometrics Research Group, Inc.** define biometria como as propriedades físicas e comportamentais mensuráveis que permitem a autenticação da identidade de um indivíduo. Assim, a biometria é usada como um termo abrangente para as tecnologias que medem as características únicas de uma pessoa para autenticar sua identidade. A dinâmica de pressionamento de tecla identifica indivíduos com base em um modelo biométrico único que analisa o padrão, ritmo e velocidade de digitação. As medições fundamentais para essa técnica são conhecidas como **"tempo de permanência"** e **"tempo de voo"**. O tempo de permanência refere-se à duração que uma tecla fica pressionada, e o tempo de voo é o intervalo entre o pressionamento de teclas consecutivas. Portanto, a dinâmica de pressionamento de tecla pode ser descrita como um algoritmo de software que mede esses tempos para autenticar a identidade.

Em 2004, pesquisadores do **MIT** estudaram a autenticação por biometria de pressionamento de tecla, identificando vantagens e desvantagens significativas no uso desta biometria para autenticação.

RECONHECIMENTO POR RETINA



(Representação do que seria o reconhecimento por Retina. Disponível em: <<https://www.controlid.com.br/blog/wp-content/uploads/2020/05/retina.jpg>>)

O **reconhecimento de retina** é um processo que envolve a digitalização da retina e a comparação da imagem obtida com um banco de dados de características retinianas únicas. Esta técnica é uma das mais distintas formas de identificação biométrica disponíveis. Uma imagem retiniana de qualidade pode apresentar até 400 pontos de dados únicos para análise, em contraste com os cerca de quarenta pontos de dados encontrados em uma impressão digital.

A tecnologia de reconhecimento retiniano foi desenvolvida inicialmente na **década de 1970**, mas somente no início dos **anos 1980** o primeiro dispositivo biométrico de reconhecimento de retina chegou ao mercado. Antes mesmo de sua comercialização, dois estudos pioneiros já haviam confirmado a unicidade dos padrões vasculares da retina.

O primeiro estudo, publicado em **1935** por **Dr. Carleton Simon** e **Dr. Isodore Goldstein**, revelou que cada retina possui um padrão vascular singular. Em um

trabalho subsequente, eles propuseram a utilização de fotografias desses padrões como um método de identificação.

O segundo estudo, realizado na **década de 1950** pelo **Dr. Paul Tower**, demonstrou que até mesmo entre gêmeos idênticos, os padrões vasculares retinianos são únicos e distintos, reforçando a ideia de que não existem duas retinas iguais. As características singulares de cada retina oferecem vantagens evidentes para o reconhecimento retiniano. Por isso, foi natural para a indústria tecnológica explorar essa biometria única para fins de identificação segura. Contudo, apesar de suas potencialidades, a aplicação dessa tecnologia no mercado permanece limitada até os dias atuais.

PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

O grupo de 6 membros foi desmembrado em 2 para o desenvolvimento do projeto, sendo 3 dos membros (**João Paulo**, **José** e **Eike**) focados na parte da documentação; e os 3 restantes (**Kaique**, **Arthur** e **Pedro**) ficaram responsáveis pelo desenvolvimento do *software* em si. Para desenvolver a aplicação de reconhecimento e autenticação facial usamos diversas ferramentas incluindo obviamente a **linguagem de programação** e a **IDE** necessária para se trabalhar com ela, várias **bibliotecas da linguagem** e outras ferramentas que permitiram a nós desenvolvedores desenvolver a aplicação de forma bem-sucedida. E esse tópico do trabalho será iniciado com a descrição delas:

LINGUAGEM DE PROGRAMAÇÃO



(Logotipo do Python, Disponível em:

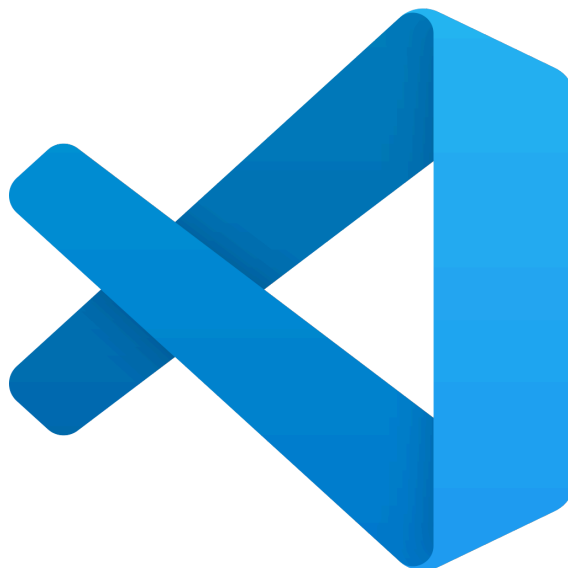
<<https://logodownload.org/wp-content/uploads/2019/10/python-logo.png>>)

A **linguagem de programação** que será utilizada é o **Python**, antes mencionada no início do trabalho. Python se trata de uma linguagem de programação amplamente utilizada atualmente principalmente em **plataformas web**, **ciência de dados** e **machine learning (ML)**. Python tem como principais características na atualidade a sua facilidade em ser aprendida e ser multiplataforma, o que atrai bastante os desenvolvedores. Outra vantagem do Python é o fato de que a linguagem tem diversas bibliotecas de desenvolvimento diferentes, o que permite a nós desenvolvedores desenvolver as mais variadas aplicações utilizando a linguagem.

As bibliotecas de **Python** que estamos utilizando atualmente são as bibliotecas **NumPy**, **cv2/opencv**, **DeepFace**, **Blinker** e **MediaPipe** (os dois últimos com auxílio do **cvzone**). **NumPy** se trata de uma das principais bibliotecas do Python na atualidade, sendo uma biblioteca focada em utilizar o Python para se

trabalhar com vetores e matrizes em cálculos matemáticos, algo que originalmente não existe na linguagem. O **cv/opencv** se trata de uma biblioteca centrada na manipulação, tratamento e utilização de imagens no geral dentro do Python, sendo uma ferramenta bem útil para o propósito do software, que lida diretamente com a questão de reconhecimento e comparação de imagens, no caso, as faces dos usuários. O **DeepFace** se trata de uma biblioteca dedicada especificamente a lidar com rostos. **Blinker** é basicamente uma biblioteca que permite “conversar” com outro documento através de sinais, sem precisar importar outro documento. A quinta e última biblioteca, que junto com a anterior vem junto ao **cvzone**, se trata da biblioteca **MediaPipe**, uma biblioteca centrada no processamento, análise e extração das informações de mídias, como vídeo, áudio e imagem, batendo com a proposta do trabalho.

IDE



(Logotipo atual do Visual Studio Code. Disponível em:

https://upload.wikimedia.org/wikipedia/commons/thumb/9/9a/Visual_Studio_Code_1.35_icon.svg/1024px-Visual_Studio_Code_1.35_icon.svg.png)

Para desenvolvimento da aplicação também será utilizada a **IDE** do **Visual Studio Code**, software da Microsoft que se trata de um dos principais ambientes de desenvolvimento de softwares dos dias atuais, qual foi escolhido graças a sua capacidade de instalar plugins e extensões personalizadas para assim adicionar mais funcionalidades a **IDE**, permitindo um desenvolvimento mais facilitado e com mais funções para serem aplicadas graças às várias funcionalidades e extensões disponíveis para a IDE além de sua compatibilidade total com o Python, a linguagem de programação antes mencionada. Existem ferramentas melhores para se utilizar quando se trabalha com Python, como por exemplo o **PyCharm**, mas a preferência da equipe de desenvolvimento pesou em favor do **Visual Studio Code** mesmo em teoria sendo um software mais limitado. Outra vantagem do Visual Studio Code que agrada mais aos membros do grupo é o que eles consideraram uma maior

facilidade para se utilizar do terminal do que nas outras plataformas, podendo se utilizar dos comandos normais do **CMD** para se manipular o que está presente no terminal.

BANCO DE DADOS



(Logotipo atual do MySQL. Disponível em: <https://pngimg.com/uploads/mysql/mysql_PNG23.png>)

O Banco de dados que será utilizado é o **MySQL**, um banco de dados bastante popular na atualidade. O **MySQL** é um sistema de gerenciamento de **banco de dados relacional** que usa **SQL** para consulta e manipulação de dados. Originalmente projetado para pequenas e médias aplicações, ele pode agora lidar com muito mais do que os cem milhões de registros por tabela sugeridos inicialmente. Sua facilidade de uso e compatibilidade com diversos sistemas operacionais tornaram o MySQL uma escolha popular para empresas como a **NASA** e a **Sony**. Além disso, o MySQL continua a evoluir, com atualizações regulares para atender melhor aos seus usuários. O MySQL também se destaca graças a sua compatibilidade com diversas ferramentas diferentes, principalmente quando se trata de desenvolvimento Web, onde a maioria dos pacotes de servidor Apache (ex: **XAMPP**, **Wampserver**, **USBWebServer**) vem tendo como seu banco de dados ou o MySQL ou uma variante de código aberto do mesmo, o **MariaDB**.

Um dos principais critérios para a escolha do MySQL como banco de dados sobre seus concorrentes foi o fato de que grande parte dos membros do grupo já tinham costume com a utilização do MySQL, seja em experiências acadêmicas anteriores ou em experiências profissionais atuais, isso permite para nós um trabalho mais rápido e tranquilo, além é claro de que o MySQL é uma opção muito viável para utilização quando trabalhado em conjunto com a linguagem de programação Python.

SGBD



(Logo do MySQL Workbench. Disponível em:
<https://cdn.icon-icons.com/icons2/1381/PNG/512/mysqlworkbench_93532.png>)

O Sistema Gerenciador de Banco de Dados (**SGBD**) escolhido para o desenvolvimento deste trabalho é o **MySQL Workbench**. Esta escolha se deve a várias razões que tornam o *Workbench* a opção mais adequada para a equipe.

Primeiramente, o MySQL Workbench é o SGBD nativo do MySQL, uma das plataformas de banco de dados mais populares e robustas disponíveis no mercado. Comparado a outras alternativas, como o SQL Server, que foi utilizado em algumas aulas anteriores, o MySQL Workbench se destacou por ser uma ferramenta com a qual os membros do grupo já tinham familiaridade e experiência prévia.

O MySQL Workbench é uma ferramenta gráfica completa para modelagem, desenvolvimento e administração de bancos de dados MySQL. Sua interface intuitiva e amigável facilita a manipulação e o gerenciamento de dados, mesmo para aqueles que estão entrando em contato com a manipulação de bancos de dados pela primeira vez. Além disso, oferece um conjunto abrangente de funcionalidades, incluindo a modelagem de dados, construção de consultas, administração de banco de dados e geração de relatórios.

Outra vantagem significativa do MySQL Workbench é sua capacidade de integração com várias outras ferramentas e tecnologias, proporcionando uma flexibilidade e eficiência que são essenciais para o desenvolvimento ágil e colaborativo de projetos. A segurança e confiabilidade do MySQL como sistema de banco de dados subjacente também foram fatores determinantes para a escolha desta ferramenta.

Por fim, o suporte e a documentação extensiva disponíveis para o MySQL Workbench foram considerados extremamente valiosos. Tais recursos facilitam a resolução de problemas e a melhoria contínua das habilidades da equipe na administração de bancos de dados.

EDITOR DO DOCUMENTO



(Logotipo do Google Docs, disponível em:

<https://upload.wikimedia.org/wikipedia/commons/thumb/0/01/Google_Docs_logo_%282014-2020%29.svg/1481px-Google_Docs_logo_%282014-2020%29.svg.png>)

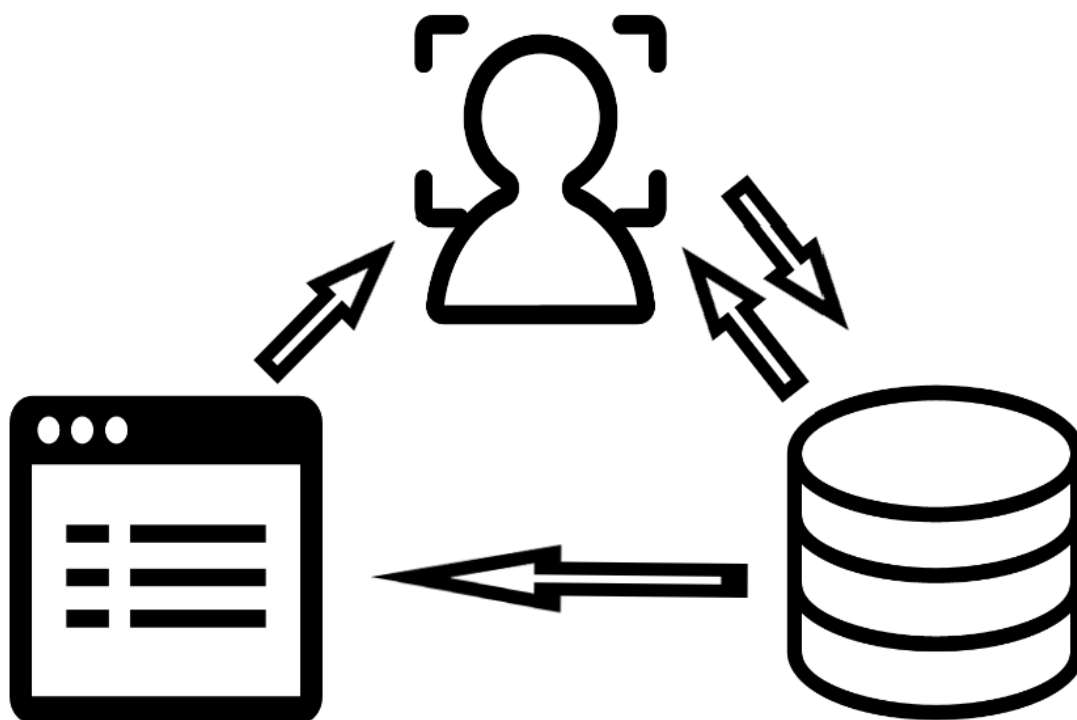
Para a **documentação** do projeto/aplicação, optamos por utilizar o Google Docs devido à sua robustez e flexibilidade. **Google Docs** é uma ferramenta online de processamento de texto que permite edição colaborativa em tempo real, o que é crucial para a nossa equipe que trabalha de forma distribuída. Além disso, por pertencer à **Google**, uma empresa reconhecida pela sua inovação e comprometimento com a segurança e privacidade dos dados, temos a confiança de que nossos documentos estarão protegidos contra acessos não autorizados. A facilidade de acesso de qualquer dispositivo com conexão à internet também contribui para a agilidade no desenvolvimento do projeto.

O Google Docs possui integração com outras ferramentas do Google Workspace, como o **Google Drive**, que facilita o armazenamento e compartilhamento de arquivos de forma segura. Os recursos de histórico de versões permitem acompanhar todas as alterações feitas no documento, assegurando que possamos recuperar qualquer versão anterior caso necessário.

Em resumo, o Google Docs não só facilita a edição colaborativa como também oferece um ambiente seguro e integrado com diversas funcionalidades que suportam o desenvolvimento ágil e eficiente do nosso projeto.

PROJETO E ESTRUTURA DO PROGRAMA

Para desenvolver o software pode-se dizer que o sistema foi separado em três partes diferentes que se comunicam umas com as outras para possibilitar o funcionamento do programa. As três partes diferentes seriam: **Interface**, **Reconhecimento de Imagem** e **Banco de Dados**. A forma como o sistema opera pode ser interpretado com o auxílio do seguinte diagrama:



(Diagrama meramente ilustrativo para demonstrar como seria a comunicação entre as diferentes partes do sistema, feito pelo próprio grupo)

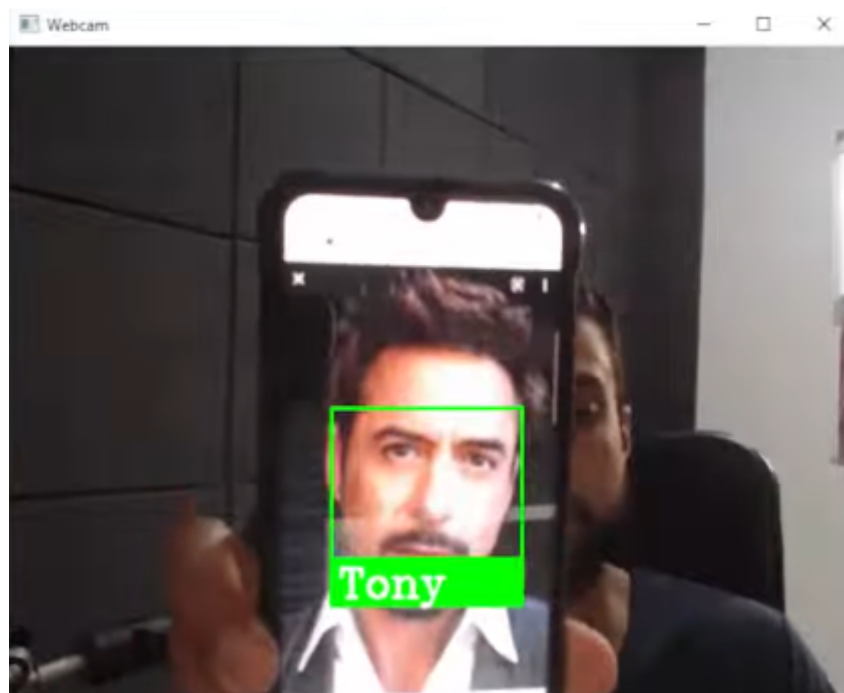
INTERFACE

A **interface** seria o contato inicial do usuário com o sistema, assim como seu próprio nome indica. Seria utilizada pelo usuário para navegar e executar as funções principais do sistema de reconhecimento facial por parte do usuário, sendo basicamente o que o usuário vê do sistema.

Ao usuário navegar pela interface e entrar com o seu rosto para possibilitar o reconhecimento, a interface iniciaria o contato com o sistema de reconhecimento de imagem que utiliza a linguagem de programação **python** e suas bibliotecas **opencv** e **MediaPipe** para fazer a manipulação e processamento da imagem para assim iniciar a lógica para o reconhecimento da imagem. A interface por não ser o foco do sistema, pois para esse trabalho estamos focando muito mais na questão do funcionamento do Backend do que no visual, já que o foco do trabalho estabelecido no tema é obviamente o reconhecimento facial em si e não na interface. Porém a interface tem sim o mínimo possível para possibilitar uma navegação clara e uma

operação que não seja complexa do Software. Para ser feita a interface foi adicionada mais uma biblioteca ao sistema, o **tkinter**.

RECONHECIMENTO DE IMAGEM



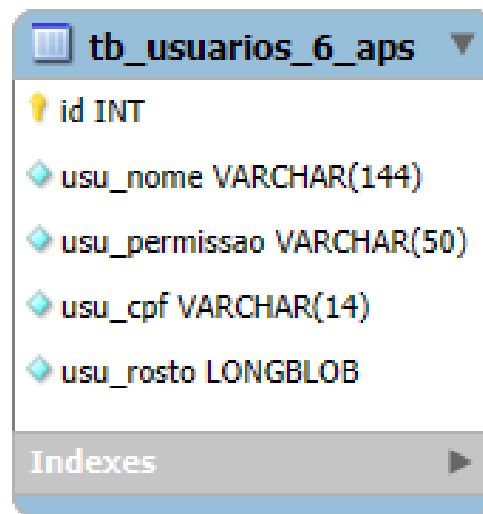
(Screenshot de um vídeo onde tecnologia de reconhecimento facial é utilizada. Disponível em: <<https://www.youtube.com/watch?v=h4yCzclMOug>>)

Para fazer o reconhecimento facial na imagem, foram empregadas a linguagem de Programação **python** e as bibliotecas **NumPy**, **DeepFace**, **cv2/opencv** e **MediaPipe** assim como foi mencionado no item 4.1 do índice.

O Backend para fazer o funcionamento do sistema irá seguir um processo relativamente simples: inicialmente A Detecção de Rosto inicializa a câmera do usuário, verificando se a porcentagem de detecção do rosto (o quão bem o rosto está visível por parte do algoritmo) é superior a 93%, se é igual ou maior que 93% ele manda respostas para interagir com o usuário para ele permanecer no lugar para ser feito o armazenamento da imagem, que é salva na forma de um array. Posteriormente o Array é transformado em um PNG, e é feita a conexão com o banco de dados para o armazenamento das informações faciais do usuário para possibilitar a comparação.

Para fazer a verificação do usuário com base no rosto não se foge muito disto também, sendo basicamente apenas uma comparação do que está sendo visto na câmera com o que é armazenado no banco de dados, o que é feito por meio do DeepFace, sendo algo que graças a biblioteca é bastante facilitado.

BANCO DE DADOS



(A única tabela que será utilizada para o armazenamento dos dados.)

O banco de dados MySQL contará com apenas uma tabela para ser trabalhada pelo sistema, a tabela de usuários, que tem a função simples porém essencial de armazenar as informações dos usuários do sistema, incluindo a parte mais importante de todas, os dados faciais do usuário, presentes na linha **usu_rosto**, que tem o tipo de dados **LONGBLOB** que é um tipo de dado específico para o armazenamento de dados binários, o que será importante para armazenar a imagem no BD. O **id** é uma chave primária não-nula que se auto-incrementa, graças ao fato de ser basicamente o atributo identificador da tabela, já todas as outras colunas tem como restrição serem não-nulas. Os atributos **usu_nome** e **usu_cpf** tem apenas a função de diferenciar um usuário do outro de forma mais visual e óbvia do que o **id**, não sendo algo essencial para o projeto mas foi feito para permitir maior refinamento, já o atributo **usu_permissão** armazena a permissão de acesso do usuário do sistema, sendo essencial para o atendimento ao solicitado pelo tema do projeto.

RELATÓRIO COM AS LINHAS DE CÓDIGO

(Projeto no Google Drive:

https://drive.google.com/drive/folders/1V1usCKXbo-DrLNYzFgxpBdliLgfFyBL-?usp=drive_link)

conectaBanco.py

```
import mysql.connector

def enviarAoBanco(nome, cpf, rosto):
    conexao = mysql.connector.connect(host='localhost', database='db_aps_6_semestre',
    user='root', password='ServidorAPS')

    if conexao.is_connected():
        coordenador = conexao.cursor()

        inserirTbAps = "INSERT INTO tb_usuarios_6_aps
        (usu_nome,usu_permissao,usu_cpf,usu_rosto) VALUES (%s,%s,%s,%s)"
        valoresTbAps = (nome, "usuario", cpf, rosto)
        coordenador.execute(inserirTbAps, valoresTbAps)

        conexao.commit() #salva as alterações no banco

        print ('enviado no banco')
        coordenador.close()
        conexao.close()
    else:
        print ('não conectado')

def pegar_imagem_para_comparar(cpf):
    conexao = mysql.connector.connect(host='localhost', database='db_aps_6_semestre',
    user='root', password='ServidorAPS')
    if conexao.is_connected():
        coordenador = conexao.cursor()

        coordenador.execute("SELECT usu_rosto FROM tb_usuarios_6_aps WHERE usu_cpf = %s",
        (cpf,))

        resultado = coordenador.fetchone()

        if resultado:
            imagem_bytes_banco = resultado[0]
            return imagem_bytes_banco
        else:
            print("Nenhuma imagem encontrada para o CPF especificado.")

        coordenador.close()
        conexao.close()
    else:
        print ('não conectado')

def ir_para_plataformaMeioAmbiente(cpf):
    conexao = mysql.connector.connect(host='localhost', database='db_aps_6_semestre',
    user='root', password='ServidorAPS')
```

```

        if conexao.is_connected():
            coordenador = conexao.cursor()

            coordenador.execute("SELECT usu_nome, usu_permissao FROM tb_usuarios_6_aps WHERE
usu_cpf = %s", (cpf,))

            resultado = coordenador.fetchone()

            if resultado:
                nome = resultado[0]
                permissao = resultado[1]

                array_nome_permissao = [nome, permissao]

                return array_nome_permissao
            else:
                print("Nenhuma imagem encontrada para o CPF especificado.")

            coordenador.close()
            conexao.close()
        else:
            print ('não conectado')

```

loginMeioAmbiente.py

```

import tkinter as tk
from testeTela import DetectorRosto
# from conectaBanco import enviarAoBanco, pegar_imagem_para_comparar
from blinker import signal
import re
import cv2

#CENTRALIZAR NA JANELA
def centralizar_janela(root, largura, altura):
    largura_tela = root.winfo_screenwidth()
    altura_tela = root.winfo_screenheight()
    pos_x = (largura_tela // 2) - (largura // 2)
    pos_y = (altura_tela // 2) - (altura // 2)
    root.geometry(f"{largura}x{altura}+{pos_x}+{pos_y}")

def formatar_cpf(texto):
    texto = re.sub(r'\D', '', texto) # Remove caracteres que não são números
    if len(texto) <= 3:
        return texto
    elif len(texto) <= 6:
        return f"{texto[:3]}.{texto[3:]}"
    elif len(texto) <= 9:
        return f"{texto[:3]}.{texto[3:6]}.{texto[6:]}"
    else:
        return f"{texto[:3]}.{texto[3:6]}.{texto[6:9]}-{texto[9:11]}"

# MASCARA PARA CPF
def aplicar_mascara_cpf(event=None):
    cpfCadastro_texto = cpfCadastro.get()
    cpfCadastro_formatado = formatar_cpf(cpfCadastro_texto)
    cpfCadastro.delete(0, tk.END)

```

```

        cpfCadastro.insert(0, cpfCadastro_formatado)

        cpfLogin_texto = cpfLogin.get()
        cpfLogin_formatado = formatar_cpf(cpfLogin_texto)
        cpfLogin.delete(0, tk.END)
        cpfLogin.insert(0, cpfLogin_formatado)

#PRIMEIRO PARAMETRO ESCONDE, SEGUNDO, APARECE
def mostrar_tela(frame_atual_esconde, frame_novo_aparece):
    frame_atual_esconde.pack_forget()
    frame_novo_aparece.pack(fill="both", expand=True)

#ROOT
def configurar_root():
    root = tk.Tk()
    root.title("Exemplo Tkinter")
    root.configure(bg="#32858C")
    centralizar_janela(root, 900, 700)
    return root

#INICIAL
def criar_tela_inicial():
    inicial = tk.Frame(root, bg="#003817")

    top_filler = tk.Frame(inicial, bg="#003817")
    top_filler.grid(row=0, column=0, columnspan=2, sticky="nsew")

    buttonlogin = tk.Button(inicial, text="LOGIN", bg="#026842", bd=0, fg="white",
font=("Helvetica", 18, 'bold'), width=20, command=lambda: mostrar_tela(inicial,
tela_logar))
    buttonlogin.grid(row=1, column=0, columnspan=2, pady=10)

    cadastrar = tk.Button(inicial, text="Não fez o cadastro? Clique, aqui!",
bg="#003817", bd=0, fg="white", command=lambda: mostrar_tela(inicial, cadastro))
    cadastrar.grid(row=2, column=0, columnspan=2, pady=5)

    bottom_filler = tk.Frame(inicial, bg="#003817")
    bottom_filler.grid(row=6, column=0, columnspan=2, sticky="nsew")

    # Configuração de layout para centralização
    inicial.grid_rowconfigure(0, weight=1)
    inicial.grid_rowconfigure(6, weight=1)
    inicial.grid_columnconfigure(0, weight=1)
    inicial.grid_columnconfigure(1, weight=1)

    return inicial

#TELA DE LOGIN
def tela_login():
    tela_logar = tk.Frame(root, bg="#003817")

    top_filler = tk.Frame(tela_logar, bg="#003817")
    top_filler.grid(row=0, column=0, columnspan=2, sticky="nsew")

```



```

        labelCpfCadastro = tk.Label(tela_logar, text="CPF:", bg="#003817", fg="white",
font=("Helvetica", 16))
        labelCpfCadastro.grid(row=1, column=0, padx=(0, 10), pady=10, sticky="e")

        global cpfLogin
        cpfLogin = tk.Entry(tela_logar, width=40, font=("Helvetica", 24))
        cpfLogin.grid(row=1, column=1, padx=(10, 0), pady=10, sticky="w")
        cpfLogin.bind("<KeyRelease>", aplicar_mascara_cpf)

        global mensagem_login
        mensagem_login = tk.Label(tela_logar, text='', font=("Helvetica", 16, "bold"),
fg='#dc143c', bg="#003817")
        mensagem_login.grid(row=2, column=0, columnspan=2, pady=10)

        login = tk.Button(tela_logar, text="ENTRAR", font=("Helvetica", 16),
command=entrar_login)
        login.grid(row=3, column=0, columnspan=2, pady=10)

        bottom_filler = tk.Frame(tela_logar, bg="#003817")
        bottom_filler.grid(row=6, column=0, columnspan=2, sticky="nsew")

        tela_logar.grid_rowconfigure(0, weight=1)
        tela_logar.grid_rowconfigure(6, weight=1)
        tela_logar.grid_columnconfigure(0, weight=1)
        tela_logar.grid_columnconfigure(1, weight=1)

        return tela_logar

#ESCONDE A TELA DE LOGIN E CHAMA A TELA FACIAL JUNTO COM PARÂMETRO PARA SABER DE QUAL
TELA FOI CHAMADO
def entrar_login():
    if cpfLogin.get() != "":
        mostrar_tela(tela_logar, criar_tela_cadastro_facial("login"))
    else:
        mensagem_login.config(text="PREENCHER TODOS OS CAMPOS!")
        mensagem_login.after(3000, lambda: mensagem_login.config(text=""))

#CADASTRO NO BANCO
def criar_tela_cadastro():
    cadastro = tk.Frame(root, bg="#003817")

    top_filler = tk.Frame(cadastro, bg="#003817")
    top_filler.grid(row=0, column=0, columnspan=2, sticky="nsew")

    labelNomeCadastro = tk.Label(cadastro, text="NOME:", bg="#003817", fg="white",
font=("Helvetica", 16))
    labelNomeCadastro.grid(row=1, column=0, padx=(0, 10), pady=10, sticky="e")

    global nomeCadastro
    nomeCadastro = tk.Entry(cadastro, width=40, font=("Helvetica", 24))
    nomeCadastro.grid(row=1, column=1, padx=(10, 0), pady=10, sticky="w")

    labelCpfCadastro = tk.Label(cadastro, text="CPF:", bg="#003817", fg="white",
font=("Helvetica", 16))
    labelCpfCadastro.grid(row=2, column=0, padx=(0, 10), pady=10, sticky="e")

```

```

global cpfCadastro
cpfCadastro = tk.Entry(cadastro, width=40, font=("Helvetica", 24))
cpfCadastro.grid(row=2, column=1, padx=(10, 0), pady=10, sticky="w")
cpfCadastro.bind("<KeyRelease>", aplicar_mascara_cpf)

global mensagem_cadastro
mensagem_cadastro = tk.Label(cadastro, text='', font=("Helvetica", 16, "bold"),
fg='#dc143c', bg="#003817")
mensagem_cadastro.grid(row=3, column=0, columnspan=2, pady=10)

botao_enviar_banco = tk.Button(cadastro, text="ENVIAR", font=("Helvetica", 20),
command=enviar_cadastro)
botao_enviar_banco.grid(row=4, column=0, columnspan=2, pady=10)

bottom_filler = tk.Frame(cadastro, bg="#003817")
bottom_filler.grid(row=6, column=0, columnspan=2, sticky="nsew")

cadastro.grid_rowconfigure(0, weight=1)
cadastro.grid_rowconfigure(6, weight=1)
cadastro.grid_columnconfigure(0, weight=1)
cadastro.grid_columnconfigure(1, weight=1)

return cadastro

#ESCONDE A TELA DE CADASTRO E CHAMA A TELA FACIAL JUNTO COM PARÂMETRO PARA SABER DE QUAL
TELA FOI CHAMADO
def enviar_cadastro():
    if nomeCadastro.get() != "" and cpfCadastro.get() != "":
        mostrar_tela(cadastro, criar_tela_cadastro_facial("cadastrar"))
    else:
        mensagem_cadastro.config(text="PREENCHER TODOS OS CAMPOS!")
        mensagem_cadastro.after(3000, lambda: mensagem_cadastro.config(text=""))

#TELA FACIAL
def criar_tela_cadastro_facial(tipo):
    global sair_facial
    cadastroFacial = tk.Frame(root, bg="#171a4a")

    button_iniciar = tk.Button(cadastroFacial, text="INICIAR CAPTURA FACIAL",
command=iniciar_captura, font=("Helvetica", 16))
    button_iniciar.pack(pady=20)

    aviso = tk.Label(cadastroFacial, text="Certifique-se de possuir uma câmera.",
font=("Helvetica", 12), fg='white', bg="#171a4a")
    aviso.pack(pady=3)

    global mensagem_resultado
    mensagem_resultado = tk.Label(cadastroFacial, font=("Helvetica", 16), fg='white',
bg="#171a4a")
    mensagem_resultado.pack(pady=10)

    global imagem_label
    imagem_label = tk.Label(cadastroFacial, bg="#171a4a")
    imagem_label.pack(pady=10)

```

```

global sinal_qual_botao_clicou
sinal_qual_botao_clicou = tipo

#IMPORTADO DO TESTE TELA
global detector
detector = DetectorRosto(mensagem_resultado, imagem_label) #SERVE PARA EXIBIR AS
MENSAGENS INTERATIVAS E O ROSTO DA PESSOA

sair_facial = cadastroFacial

return cadastroFacial

#INICIA A CAPTURA DO ROSTO
def iniciar_captura():
    mensagem_resultado.config(text="")
    detector.iniciar_captura()

#COM BASE NO PARÂMETRO 'TIPO' DA TELA FACIAL, SERÁ ACEITA UMA DAS CONDIÇÕES
def voltar_cadastro_login(sender, **kwargs):
    if (sinal_qual_botao_clicou == "logar"):

        sinal_para_comparar_imagem = signal('chamar_agora_comparador')
        comparar_cpf = cpfLogin.get()
        detector.verificar_sinais(sinal_para_comparar_imagem, comparar_cpf=comparar_cpf)

    elif (sinal_qual_botao_clicou == "cadastrar"):

        sinal_para_enviar_ao_banco = signal('chamar_agora_enviar_banco')
        nome_cadastro = nomeCadastro.get()
        cpf_cadastro = cpfCadastro.get()
        detector.verificar_sinais(sinal_para_enviar_ao_banco,
        nome_cadastro=nome_cadastro, cpf_cadastro=cpf_cadastro)
        return

#SINAL EMITIDO PELO BACK, PARA SABER QUE A CAPTURA TERMINOU
def conectar_sinal_voltar():
    voltar_para_cadastro_login = signal('voltar_cadastro_login')
    voltar_para_cadastro_login.connect(voltar_cadastro_login)

#SINAL QUE É RECEBIDO APÓS SER VALIDADO O LOGIN
def sinal_meioAmbiente():
    ir_plataformaMeioAmbiente = signal('ir_plataformaMeioAmbiente')
    ir_plataformaMeioAmbiente.connect(informacoes_para_meioAmbiente)

#PEGA NOME E PERMISSÃO DO USUARIO E ENVIA PARA A ULTIMA TELA
def informacoes_para_meioAmbiente(sender, **kwargs):
    cpf = cpfLogin.get()
    array_banco = detector.informacoes_banco_plataformaMeioAmbiente(cpf)

    nome_banco = array_banco[0]
    permissao_banco = array_banco[1]

    if (nome_banco != '' and permissao_banco != ''):
        mostrar_tela(sair_facial, plataformaMeioAmbiente(nome_banco, permissao_banco))
    else:

```

```

        print("array_nome_permissao está vazio")

#PLATAFORMA MEIO AMBIENTE (ULTIMA TELA)
def plataformaMeioAmbiente(nome, permissao):
    meioAmbiente = tk.Frame(root, bg="#003817")

    top_filler = tk.Frame(meioAmbiente, bg="#003817")
    top_filler.grid(row=0, column=0, columnspan=2, sticky="nsew")

    labelMeioAmbiente = tk.Label(meioAmbiente, text="BEM VINDO! %s, %s da Plataforma
Meio Ambiente." % (nome, permissao), bg="#003817", fg="white", font=("Helvetica", 16))
    labelMeioAmbiente.grid(row=2, column=0, columnspan=2, pady=10)

    bottom_filler = tk.Frame(meioAmbiente, bg="#003817")
    bottom_filler.grid(row=6, column=0, columnspan=2, sticky="nsew")

    # Configuração de layout para centralização
    meioAmbiente.grid_rowconfigure(0, weight=1)
    meioAmbiente.grid_rowconfigure(6, weight=1)
    meioAmbiente.grid_columnconfigure(0, weight=1)
    meioAmbiente.grid_columnconfigure(1, weight=1)

    return meioAmbiente

root = configurar_root()
inicial = criar_tela_inicial()
cadastro = criar_tela_cadastro()
tela_logar = tela_login()
meioAmbiente = plataformaMeioAmbiente(None, None)

#FUNÇÃO SEMPRE SERÁ CHAMADA PARA SABER QUANDO É FEITO O SINAL NO BACKEND
conectar_sinal_voltar()
sinal_meioAmbiente()

#CHAMA A TELA INICIAL
inicial.pack(fill="both", expand=True)
root.mainloop()

```

testeTela.py

```

import time

import cv2
import numpy as np
from blinker import signal
from cvzone.FaceDetectionModule import FaceDetector
from PIL import Image, ImageTk
from deepface import DeepFace;

from conectaBanco import enviarAoBanco, pegar_imagem_para_comparar,
ir_para_plataformaMeioAmbiente

```

```

class DetectorRosto():
    def __init__(self, label_mensagem, label_imagem):
        self.label_mensagem = label_mensagem
        self.label_imagem = label_imagem
        self.video = cv2.VideoCapture(0)
        self.detector = FaceDetector()
        self.start_time = 0
        self.tempo_estipulado = 8
        self.mensagem = ""
        self.img_salvar_banco = np.array([])
        self.mask = None # Variável para a máscara
        self.voltar_cadastro_login = signal('voltar_cadastro_login')
        self.sinal_comparador = signal('chamar_agora_comparador')
        self.sinal_para_enviar_ao_banco = signal('chamar_agora_enviar_banco')
        self.ir_plataformaMeioAmbiente = signal('ir_plataformaMeioAmbiente')

    def iniciar_captura(self):
        if not self.video.isOpened():
            self.video = cv2.VideoCapture(0)
            self.start_time = 0
            self.iniciar_captura()

        self.start_time = 0
        self.atualizar_video()

    def criar_mascara(self, height, width):
        """Cria uma máscara em forma de elipse."""
        self.mask = np.zeros((height, width), dtype=np.uint8)
        center, axes = (width // 2, height // 2), (width // 5, height // 3)

        cv2.ellipse(self.mask, center, axes, 0, 0, 360, 255, -1)

    def atualizar_video(self):
        success, img = self.video.read()
        if not success:
            print("Falha ao capturar imagem.")
            return

        height, width, _ = img.shape
        if self.mask is None:
            self.criar_mascara(height, width)

        # Aplica a máscara
        img_masked = cv2.bitwise_and(img, img, mask=self.mask)
        mostra_mask_rosto = cv2.bitwise_and(img, img, mask=self.mask)

        img_masked, bboxes = self.detector.findFaces(img_masked, draw=True)

        if bboxes:
            confidence = bboxes[0]["score"][0] * 100
            tempo_atual = time.time() - self.start_time

            if confidence < 89:
                self.mensagem = "Fique no centro para ser visualizado."
            elif confidence >= 93:

```

```

        if tempo_atual <= 1.5:
            self.mensagem = "Aproxime-se para ser reconhecido"
        elif 1.7 <= tempo_atual <= 3.7:
            self.mensagem = "Rosto detectado, aguarde..."
        elif 3.8 <= tempo_atual <= 7.5:
            self.mensagem = "Mantenha-se posicionado."
        elif self.start_time == 0:
            self.start_time = time.time()
        elif time.time() - self.start_time >= self.tempo_estipulado:
            self.mensagem = "Capturando imagem..."
            cv2.waitKey(1)
            time.sleep(1)
            self.img_salvar_banco = mostra_mask_rosto
            self._, self.rosto_png = cv2.imencode('.png',
self.img_salvar_banco)
            self.mostrar_resultado(mostra_mask_rosto)
            return
        else:
            self.start_time = 0
    else:
        self.mensagem = "Nenhum rosto detectado."

# Atualiza a mensagem no Tkinter
self.label_mensagem.config(text=self.mensagem)

# Atualiza a imagem no Label
img_rgb = cv2.cvtColor(img_masked, cv2.COLOR_BGR2RGB) # Usar a imagem mascarada
img_pil = Image.fromarray(img_rgb)
img_tk = ImageTk.PhotoImage(img_pil)
self.label_imagem.config(image=img_tk)
self.label_imagem.image = img_tk # Mantém uma referência da imagem

# Chama novamente esta função após um breve intervalo
self.label_imagem.after(100, self.atualizar_video)

def mostrar_resultado(self, img):
    # Atualiza a mensagem e a imagem
    self.label_mensagem.config(text="Imagem capturada!")
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_pil = Image.fromarray(img_rgb)
    img_tk = ImageTk.PhotoImage(img_pil)
    self.label_imagem.config(image=img_tk)
    self.label_imagem.image = img_tk

self.video.release() # Libera a câmera
self.voltar_cadastro_login.send()

def verificar_sinais(self, sender, **kwargs):
    if sender == self.sinal_comparador:

        comparar_cpf = kwargs.get('comparar_cpf')
        # print(comparar_cpf)

```

```

        self.comparar_imagens(comparar_cpf)
    elif sender == self.sinal_para_enviar_ao_banco:

        nome_cadastro = kwargs.get('nome_cadastro')
        cpf_cadastro = kwargs.get('cpf_cadastro')
        self.enviar_dados_banco(nome_cadastro, cpf_cadastro)
    else:
        print("Sinal desconhecido.")

def comparar_imagens(self, cpf_interface):
    self.label_mensagem.config(text="Analisando imagem, aguarde...")

    imagem_banco_bytes = pegar_imagem_para_comparar(cpf_interface)
    imagem_banco_array = cv2.imdecode(np.frombuffer(imagem_banco_bytes, np.uint8),
cv2.IMREAD_COLOR)

    imagem = self.img_salvar_banco
    imagem2 = imagem_banco_array

    if imagem is None or imagem2 is None:
        print("Erro ao decodificar uma das imagens.")
        return

    try:
        resultado = DeepFace.verify(imagem, imagem2, model_name='Facenet',
enforce_detection=False)
        if resultado.get("verified"):
            print("As imagens correspondem: são a mesma pessoa.")
            print(resultado["verified"])
            self.ir_plataformaMeioAmbiente.send()
        else:
            print("As imagens não correspondem: não são a mesma pessoa.")
            print(resultado["verified"])
    except Exception as e:
        print(f"Ocorreu um erro durante a verificação: {e}")

def informacoes_banco_plataformaMeioAmbiente(self, cpf_interface):
    array_nome_permissao = ir_para_plataformaMeioAmbiente(cpf_interface)
    return array_nome_permissao

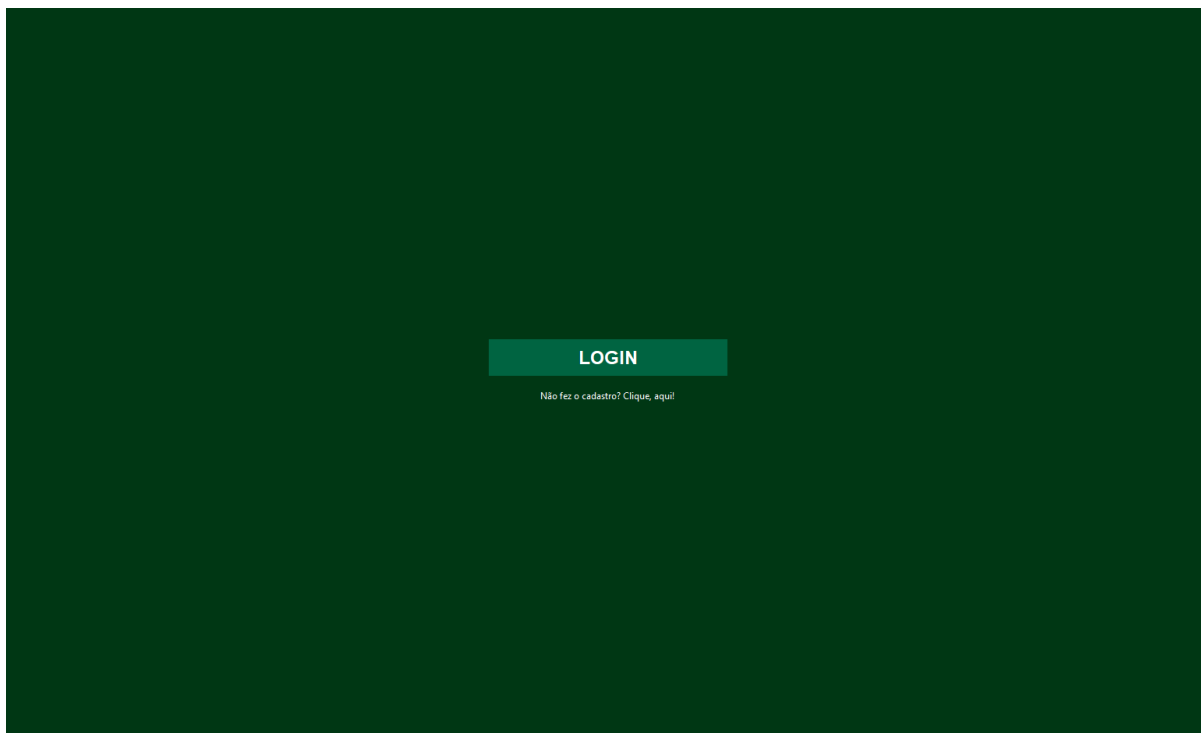
def enviar_dados_banco(self, enviar_nomeCadastro, enviar_cpf):

    enviar_rostoCadastro_binario = self.rosto_png.tobytes()
    enviarAoBanco(enviar_nomeCadastro, enviar_cpf, enviar_rostoCadastro_binario)

```

SOFTWARE EM EXECUÇÃO

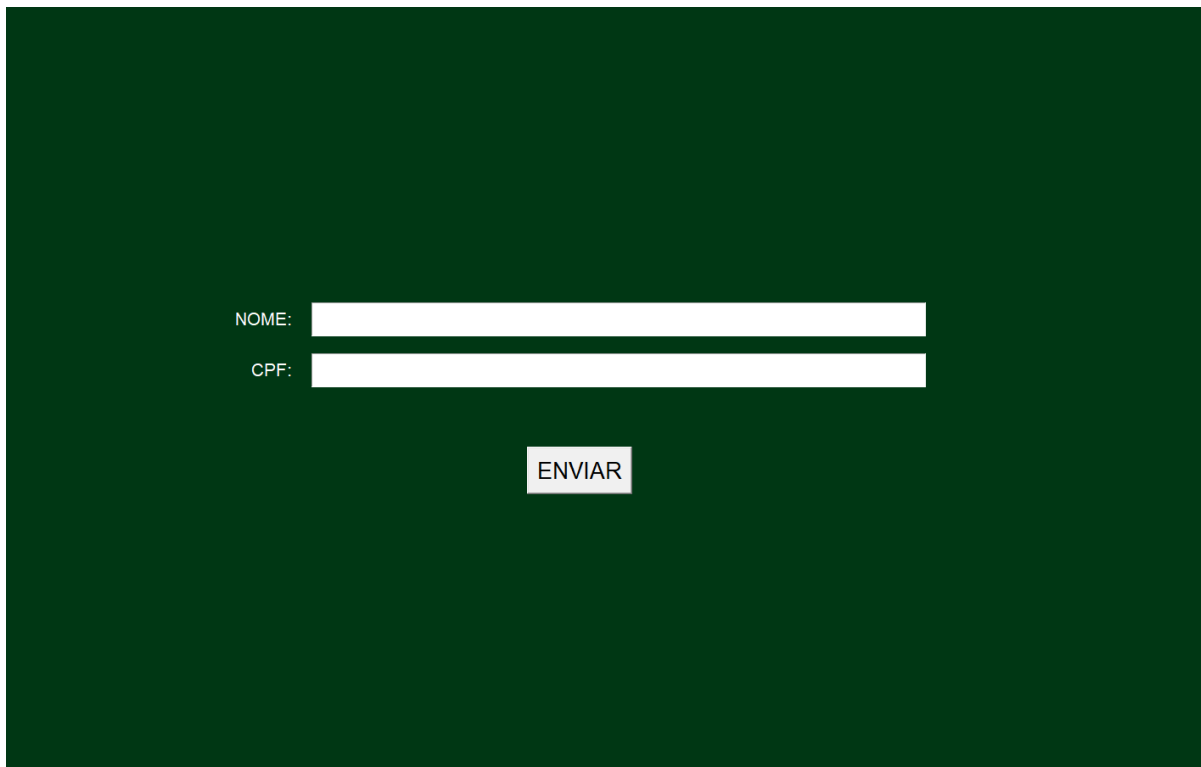
Home

The image shows a dark green rectangular area representing a login form. In the center, there is a light green rectangular button with the word "LOGIN" in black, uppercase letters. Below the button, there is a small line of text in a lighter green color that reads "Não fez o cadastro? Clique, aqui!".

LOGIN

Não fez o cadastro? Clique, aqui!

Registro

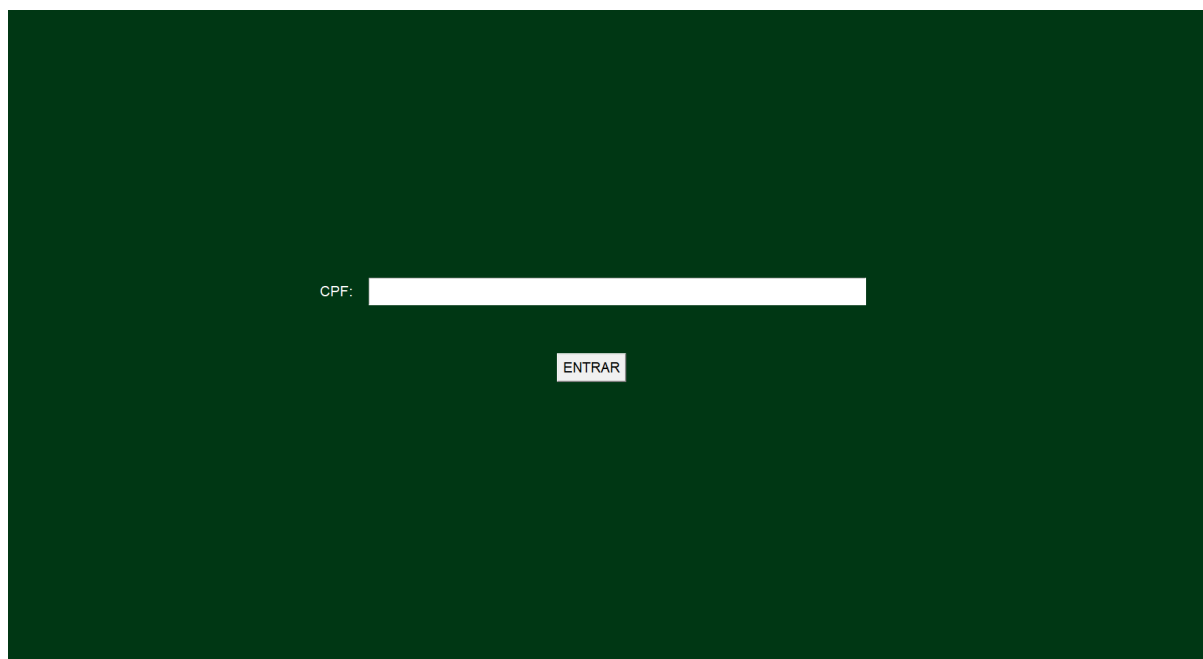
The image shows a dark green rectangular area representing a registration form. It contains two input fields with labels to their left. The first label is "NOME:" and the second is "CPF:". Below these fields is a light green rectangular button with the word "ENVIAR" in black, uppercase letters.

NOME:

CPF:

ENVIAR

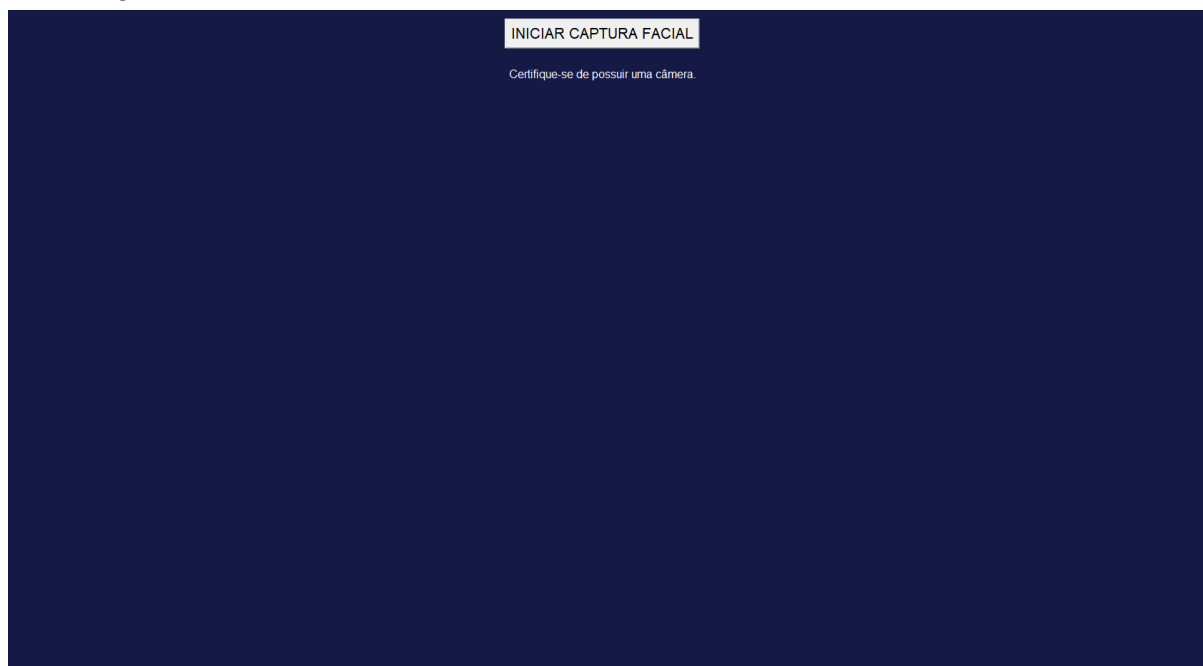
LOGIN



CPF:

ENTRAR

DETECÇÃO DE FACE



INICIAR CAPTURA FACIAL

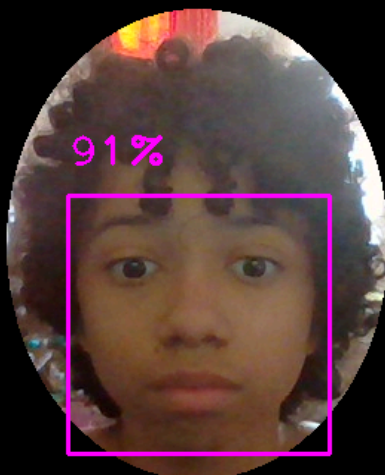
Certifique-se de possuir uma câmera.

ACESSO DE ADMINISTRADOR

INICIAR CAPTURA FACIAL

Certifique-se de possuir uma câmera.

Rosto detectado, aguarde...



BEM VINDO! Pedro, administrador da Plataforma Meio Ambiente.

ACESSO DE USUÁRIO



BEM VINDO! Kaique, usuario da Plataforma Meio Ambiente.

REFERÊNCIAS

IMAGE recognition with Python, OpenCV, OpenAI CLIP and pgvector. aiven.io. Disponível em: <https://aiven.io/developer/find-faces-with-pgvector>. Acesso em: 5 de Out.

PGVECTOR. [github.com](https://github.com/pgvector/pgvector). Disponível em: <https://github.com/pgvector/pgvector>. Acesso em: 5 de Out.

FACE-RECOGNITION 1.3.0. [pypi.org](https://pypi.org/project/face-recognition/). Disponível em: <https://pypi.org/project/face-recognition/>. Acesso em: 5 de Out.

O que é reconhecimento facial – definição e explicação. [kaspersky.com.br](https://www.kaspersky.com.br/resource-center/definitions/what-is-facial-recognition). Disponível em: <https://www.kaspersky.com.br/resource-center/definitions/what-is-facial-recognition>. Acesso em: 6 de Out.

BIOMETRIA facial: 3 razões para aderir em sua empresa. [hagana.com.br](https://www.hagana.com.br/blog/biometria-facial-3-razoes-para-aderir-em-sua-empresa). Disponível em: <https://www.hagana.com.br/blog/biometria-facial-3-razoes-para-aderir-em-sua-empresa>. Acesso em: 6 de Out.

ENTENDENDO a Tecnologia de Reconhecimento Facial: Como Funciona e Exemplos. [visionplatform.ai](https://visionplatform.ai/pt/entendendo-a-tecnologia-de-reconhecimento-facial-como-funciona-e-exemplos). Disponível em: <https://visionplatform.ai/pt/entendendo-a-tecnologia-de-reconhecimento-facial-como-funciona-e-exemplos>. Acesso em: 6 de Out.

GOGONI, Ronaldo. O que é biometria? Os 6 tipos mais usados na tecnologia. [tecnoblog.net](https://tecnoblog.net/responde/o-que-e-biometria-tecnologia). Disponível em: <https://tecnoblog.net/responde/o-que-e-biometria-tecnologia>. Acesso em: 12 de Out.

KATO, Nathalie. Tipos de biometria: conheça os principais. [vsoft.com.br](https://www.vsoft.com.br/post/principais-tipos-de-biometria). Disponível em: <https://www.vsoft.com.br/post/principais-tipos-de-biometria>. Acesso em: 12 de Out.

RECONHECIMENTO da orelha? Conheça inovações em biometria e no reconhecimento de usuários! [dialogando.com.br](https://dialogando.com.br/inovacao/reconhecimento-da-orelha-conheca-inovacoes-em-biometria-e-no-reconhecimento-de-usuarios). Disponível em: <https://dialogando.com.br/inovacao/reconhecimento-da-orelha-conheca-inovacoes-em-biometria-e-no-reconhecimento-de-usuarios>. Acesso em: 12 de Out.

FELIX, Victor Hugo. Como funciona o reconhecimento de voz? [tecnoblog.net](https://tecnoblog.net/responde/como-funciona-o-reconhecimento-de-voz/). Disponível em: <https://tecnoblog.net/responde/como-funciona-o-reconhecimento-de-voz/>. Acesso em: 12 de Out.

COMO funcionam os sistemas biométricos de reconhecimento pela íris? [eupercebo.unb.br](https://eupercebo.unb.br/2022/07/27/como-funcionam-os-sistemas-biometricos-de-reconhecimento-pela-iris/). Disponível em: <https://eupercebo.unb.br/2022/07/27/como-funcionam-os-sistemas-biometricos-de-reconhecimento-pela-iris/>. Acesso em: 12 de Out.

KING, Rawlson. Explainer: Keystroke recognition. [biometricupdate.com](https://www.biometricupdate.com/201612/explainer-keystroke-recognition). Disponível em: <https://www.biometricupdate.com/201612/explainer-keystroke-recognition>. Acesso em: 12 de Out.

<https://www.rootstrap.com/blog/retinal-recognition-the-ultimate-biometric>. Acesso em: 12 de Out.

MONGODB Atlas. MongoDB totalmente gerenciada na cloud. [mongodb.com](https://www.mongodb.com/pt-br/lp/cloud/atlas/try4?utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_core_prosp-brand_gic-null_amers-br_ps-all_desktop_pt-br_lead&utm_term=mongodb%20atlas&utm_medium=cpc_paid_search&utm_ad=e&utm_ad_campaign_id=20378068769&adgroup=154980291281&cq_cmp=20378068769&gad_source=1&gclid=CjwKCAjw3624BhBAEiwAkxgTOnl4uWg7XiZFc_mtZNIxKI01J25cb1g9kgffygov7jAupY11UoSYaBoCtEoQAvD_BwE). Disponível em: https://www.mongodb.com/pt-br/lp/cloud/atlas/try4?utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_core_prosp-brand_gic-null_amers-br_ps-all_desktop_pt-br_lead&utm_term=mongodb%20atlas&utm_medium=cpc_paid_search&utm_ad=e&utm_ad_campaign_id=20378068769&adgroup=154980291281&cq_cmp=20378068769&gad_source=1&gclid=CjwKCAjw3624BhBAEiwAkxgTOnl4uWg7XiZFc_mtZNIxKI01J25cb1g9kgffygov7jAupY11UoSYaBoCtEoQAvD_BwE. Acesso em: 13 de Out.

O que é python? [aws.amazon.com](https://aws.amazon.com/pt/what-is/python/). Disponível em: <https://aws.amazon.com/pt/what-is/python/>. Acesso em: 19 de Out.

NUMPY Introduction. [w3schools.com](https://www.w3schools.com/python/numpy/numpy_intro.asp). Disponível em: https://www.w3schools.com/python/numpy/numpy_intro.asp. Acesso em: 19 de Out.

OPENCV-PYTHON 4.10.0.84. [pypi.org](https://pypi.org/project/opencv-python/). Disponível em: <https://pypi.org/project/opencv-python/>. Acesso em: 19 de Out.

BLINKER 1.9.0. [pypi.org](https://pypi.org/project/blinker/). Disponível em: <https://pypi.org/project/blinker/>. Acesso em: 19 de Out.

APRENDA a Utilizar o Mediapipe com Python para Processamento de Mídia. awari.com.br. Disponível em: <https://awari.com.br/aprenda-a-utilizar-o-mediapipe-com-python-para-processamento-de-midia>. Acesso em: 19 de Out.

VISUAL Studio Code documentation. code.visualstudio.com. Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 19 de Out.

DOCUMENTOS Google: editor de documentos on-line. gmail.google.com. Disponível em: <https://gmail.google.com/docs/about>. Acesso em: 19 de Out.

DOCUMENTOS Google. caracteristicas.pt. Disponível em: <https://caracteristicas.pt/documentos-google/>. Acesso em: 19 de Out.

MELO, Welber. Para que serve o Google Docs? digitalmentetech.com. Disponível em: <https://digitalmentetech.com/glossario/para-que-serve-o-google-docs/>. Acesso em: 19 de Out.

MYSQL Workbench. dev.mysql.com. Disponível em: <https://dev.mysql.com/doc/workbench/en/>. Acesso em: 19 de Out.

JOSÉ. Introdução ao MySQL: Principais Características desse SGBD. [devmedia.com.br](https://www.devmedia.com.br). Disponível em: <https://www.devmedia.com.br/introducao-ao-mysql/27799>. Acesso em: 19 de Out.

MYSQL 8.4 Reference Manual. dev.mysql.com. Disponível em: <https://dev.mysql.com/doc/refman/8.4/en/>. Acesso em: 19 de Out.

DEEPFACE 0.0.93. pypi.org. Disponível em: <https://pypi.org/project/deepface/>. Acesso em: 20 de Out.

FICHA DA APS



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: João Paulo Menezes Santos TURMA: CC 6 P 41 RA: N 861347
 CURSO: Ciência da Computação CAMPUS: Santos - Rangel SEMESTRE: 6º TURNO: Noite
 CÓDIGO DA ATIVIDADE: 77B3 SEMESTRE: 6º ANO GRADE: 2024

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
05/10	Objetivo e motivação do trabalho	1h 30m	João Paulo M. Santos	1h	
06/10	Introdução	1h 30m	João Paulo M. Santos	1h	
12/10	Fundamentos das Principais Técnicas	2h 30m	João Paulo M. Santos	2h	
13/10	Plano de Desenvolvimento	3h	João Paulo M. Santos	2,5h	
20/10	Projeto e Estrutura	2h 30m	João Paulo M. Santos	2h	
26/10	Relatório com Linhas de Código	1h 30m	João Paulo M. Santos	1h	
13/10	Instalação do BD	1h 30m	João Paulo M. Santos	1h	
19/10-04/11	Integração	6h	João Paulo M. Santos	5,5h	
11/10-03/11	Lógica e Código do Software	9h	João Paulo M. Santos	8,5h	

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

TOTAL DE HORAS ATRIBUÍDAS: 24,5 horas

AValiação: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO