Framework de autenticação web via reconhecimento facial

Samuel F. Santos [1], **Mateus Tymburibá** [1]

[1] Departamento de Computação (DECOM)
Centro Federal de Educação Tecnologica de Minas Gerais (CEFET-MG)
Belo Horizonte, MG – Brazil

samuelfds286@gmail.com, mateustymbu@cefetmg.br

Resumo. *A autenticação em sistemas web é um processo essencial para garantir a segurança e a privacidade dos usuários. Uma forma de aprimorar esse processo é utilizar o reconhecimento facial, que permite identificar os usuários com base em suas características faciais. Neste artigo, apresentamos uma aplicação web que utiliza o reconhecimento facial para aprimorar a autenticação dos usuários, usando a FaceAPI da Microsoft Azure. A aplicação foi testada e analisada sob diferentes aspectos, como o formato de imagem, o intervalo de tempo e o modelo de reconhecimento facial. Os resultados mostraram que a aplicação é confiável e eficiente. A aplicação também é robusta frente às mudanças nas expressões faciais e ao uso de máscaras faciais.*

Abstract. *Authentication in web systems is an essential process to ensure the security and privacy of users. One way to enhance this process is to use facial recognition, which allows identifying users based on their facial features. In this article, we present a web application that uses facial recognition to enhance the authentication of users, using the FaceAPI from Microsoft Azure. The application was tested and analyzed under different aspects, such as the image format, the time interval and the facial recognition model. The results showed that the application is reliable and efficient. The application is also robust against changes in facial expressions and the use of facial masks.*

1. Introdução

A *Internet* e a *World Wide Web* revolucionaram as comunicações globais, democratizando a informação e a oferta de produtos e serviços [Monteiro 2001]. Porém, para acessar certas funcionalidades, é preciso autenticar e identificar os usuários. Muitas vezes, eles criam credenciais fracas [Avast 2019] em vários serviços [TiLT 2019], comprometendo a segurança e facilitando ataques cibernéticos. A quebra de autenticação é um dos principais riscos em aplicações web, segundo o projeto *Top Ten* da OWASP (*Open Web Application Security Project*) [OWASP 2021]. Isso preocupa os desenvolvedores *web*, pois os usuários nem sempre seguem as medidas de segurança recomendadas.

No mundo físico, o reconhecimento facial é uma forma comum de identificação entre os humanos. Logo, seria natural usar o rosto do usuário como um meio confiável e seguro de autenticação na *web*. Com os avanços na tecnologia, algoritmos de visão computacional [Fajardo 2019], inteligência artificial e aprendizado de máquina, é possível criar aplicações que usam o reconhecimento facial como método autenticador. Contudo, apesar dessa funcionalidade ser comum em dispositivos móveis, como *smartphones*, seu

uso ainda é pouco explorado em aplicações *web*. Isso implica na falta de análise sobre seus benefícios, limitações e questões de privacidade das imagens dos usuários [TEG6 2021].

Diante disso, este trabalho propõe uma aplicação *web* que usa o reconhecimento facial para melhorar a segurança do usuário na autenticação. Além disso, a linguagem *JavaScript*, na forma nativa *ECMA6*, facilita a replicação do projeto por outros desenvolvedores interessados. Por fim, foi disponibilizado um *framework*[1] que fornece aos desenvolvedores os códigos, arquitetura e etapas do processo usados na aplicação. Isso permitirá a criação de sistemas *web* mais seguros e intuitivos, que serão usados pelos usuários para reconhecimento e autenticação diários.

2. Visão Geral

Este projeto criou um *framework* de autenticação *web* usando o reconhecimento facial como uma alternativa para outros desenvolvedores aprimorarem a autenticação na *web*. Para reduzir custos e facilitar sua distribuição, o protótipo se limita a usar *API's* e *softwares* gratuitos com documentações acessíveis.

O projeto segue o modelo de estrutura distribuída entre cliente-servidor. Assim, recursos do lado do cliente, como a câmera do dispositivo do usuário, e do lado do servidor, como um serviço de armazenamento de dados, são necessários para garantir a viabilidade do seu uso.

A ideia geral do sistema consiste na captação da imagem do usuário pela câmera do dispositivo, *notebook* ou *webcam*, que fará uma verificação com a imagem armazenada, por um cadastro prévio, no banco de dados. Com isso, se a verificação for válida e retornar os parâmetros necessários que confirmem que as credenciais faciais são as mesmas, o processo de *login* será efetivado.

Conceitualmente, cada usuário possui uma credencial com uma imagem pessoal. No processo de *login*, o servidor busca no banco os dados vinculados ao usuário para iniciar a sessão de gravação. Em seguida, a captura de imagens do usuário é executada. A foto atual e a foto do banco são enviadas para a *API*, que realiza o processo de verificação entre as duas faces, detectando as características semelhantes. Ela retorna o percentual de parentesco das duas imagens. Depois, o nível de confiança é enviado para o servidor. Se o percentual for inferior a 60%, o processo é repetido por 3 vezes. Caso contrário, a autenticação é efetivada. Esse valor de confiança foi estabelecido através de um teste preliminar com usuários, onde verificou-se que a confiabilidade variava entre 65% e 72%. Logo, o valor de 60% viabiliza a autenticação no pior cenário.

Para oferecer um arcabouço tecnológico acessível e gratuito para outros desenvolvedores, é utilizada uma API externa da **Microsoft Azure** chamada **FaceAPI**. Ela tem um amplo uso na autenticação por dispositivos como os *smartphones* e na identificação de usuários em fábricas e empresas por câmeras comuns.

3. Trabalhos Relacionados

O projeto se baseia em ideias já aplicadas em outros ambientes. Ao analisar dois trabalhos referenciados, percebemos que esse projeto representa uma evolução e melhoria das conclusões e abstrações desses trabalhos.

[1]Repositório público GitHub: `https://github.com/Samuel1s/WEB-FaceRecognition`

Um desses trabalhos foi conduzido pela Universidade de Durban, na África do Sul, e apresentado na conferência (*ICTAS*), em inglês. O autor propôs o uso de redes neurais e da linguagem *Web Assembly* para o reconhecimento facial em tempo real na *web*. O autor justifica a escolha dessas tecnologias, mencionando as limitações da linguagem *JavaScript* em relação à implementação de medidas de segurança mais complexas. A utilização de uma linguagem nativa é apontada como uma solução mais eficiente, capaz de otimizar o tempo de resposta e lidar com requisitos de segurança mais rigorosos [Prashaan Pillay 2019].

O processo de reconhecimento facial envolve a leitura, ajuste, caracterização e comparação de múltiplas faces capturadas em tempo real, resultando em uma melhoria na eficiência da autenticação, especialmente diante das mudanças nas características faciais dos usuários. No entanto, o sistema proposto demanda uma grande capacidade de armazenamento e processamento. Além disso, a utilização de uma linguagem menos amigável e as limitações em relação à escalabilidade podem impactar a expansão e acessibilidade da aplicação.

Uma novidade deste projeto é o uso de uma *API* moderna baseada na nuvem (do inglês *Cloud*), que captura os pontos de referência faciais e retorna atributos que definem as características de cada face capturada em tempo real. Essa abordagem permite a externalização do processamento de dados, facilitando a aplicação por meio de chamadas *REST* que fornecem os atributos faciais e os resultados das associações faciais.

Outro trabalho relevante envolve a autenticação de usuários por meio do reconhecimento facial e gestos definidos por eles. Desenvolvida e publicada pela Universidade de Santa Clara, Califórnia, EUA, a autora destaca a criação de uma arquitetura eficiente para a aplicação, bem como a utilização de gestos de mão como uma segunda chave de autenticação. Os resultados obtidos foram bem-sucedidos, indicando uma precisão substancial para a aplicação do reconhecimento facial em processos de autenticação na *web*. No entanto, todos os testes foram realizados em um ambiente controlado, o que não garante sua efetividade e usabilidade em cenários reais [Ruiwen Li 2018].

Com isso, para aumentar a credibilidade dos resultados, o presente projeto utilizou servidores de armazenamento e processamento de imagens provisionados na nuvem (*Internet*). Essa abordagem permite que os testes sejam realizados em um cenário mais próximo da realidade, oferecendo uma visão mais precisa do desempenho da tecnologia de reconhecimento facial em situações reais.

4. Desenvolvimento

O sistema conta com 8 requisitos funcionais e 2 requisitos não funcionais. O *email* único foi utilizado para distinguir os usuários no cadastro, evitando duplicidade de endereços. Isso facilita a busca de dados pelo servidor, que usa o *email* e o *password* como parâmetros de filtragem. Ao abrir uma sessão, o usuário tem um tempo limite para interagir. Assim, caso a ação de sair do sistema seja esquecida, ele não tem seus dados exibidos.

4.1. Componentes do Sistema

Com os requisitos definidos, foi possível escolher a arquitetura necessária para a construção do protótipo. Buscou-se utilizar as tecnologias mais adequadas e presentes

no mercado para desenvolvimento de aplicações *web*, somadas com a implementação de boas práticas de desenvolvimento de software [Consultoria 2016].

Entre os diversos padrões de projetos existentes para criação de aplicações cliente-servidor, o escolhido como arquétipo da arquitetura desse projeto foi o *Model-View-Controller* (**MVC**), que divide a aplicação em três componentes interoperáveis que otimizam a velocidade entre as requisições feitas pelo comando dos usuários [Zucher 2020].

4.2. Arquitetura

A aplicação foi desenvolvida usando HTML, CSS e JavaScript. Dentre as várias tecnologias disponíveis, foi escolhido o **PUG** para este projeto, devido à sua sintaxe otimizada e intuitiva. Essa escolha permitiu que a manipulação do *Document Object Model* (**DOM**) fosse realizada de forma nativa, mantendo a integridade do código [Njah 2022]. Por outro lado, se frameworks javascripts, como *React* e *Vue*, fossem utilizados para construir a interface gráfica, seria necessário configurar transcompiladores para interpretar o código pelo navegador [BabelORG 2015]. Portanto, a utilização do **PUG** facilita a replicação do projeto e reduz as barreiras de aprendizado para outros desenvolvedores.

No lado do servidor, o *NodeJS* foi adotado como o ambiente de execução Javascript fora do navegador. Isso proporcionou a utilização do mesmo conhecimento adquirido no desenvolvimento do frontend e permitiu a configuração de servidores locais. O NodeJS também viabilizou a utilização do NPM (*Node Package Manager*), um conjunto de módulos que simplifica a instalação, desinstalação e atualização de dependências em uma aplicação.

Então, com o NPM a construção da estrutura do backend, incluindo rotas, migração de dados, conexão com banco de dados e a lógica de controle, são facilitados. Logo, as tecnologias utilizadas contribuem para a construção eficiente da interface gráfica e do servidor, permitindo a reutilização de conhecimentos e facilitando a replicação do projeto por outros desenvolvedores.

4.2.1. Agentes Externos: API e Banco de Dados

No processo de desenvolvimento desse trabalho, imaginou-se a criação de um protótipo que representasse um cenário real. Para isso, foi arquitetado um esquema de banco de dados em nuvem para dimensionar o tempo de resposta da aplicação ao carregar estruturas de dados relativamente grandes. Tal fato está associado principalmente ao armazenamento de imagens e sua leitura durante o processo de autenticação.

A escolha do provedor MongoDB Atlas foi estratégica, pois durante a esquematização das possíveis entidades necessárias para o projeto, verificou-se a utilidade de apenas uma. Sendo assim, para viabilidade do projeto, imaginou-se o uso de manipulação de documentos em formato JSON (do inglês *JavaScript Object Notation*), pois garantiria uma rapidez de processamento e a exclusão de processos de integração de tecnologias de mapeamento de dados para diferentes ambientes. O MongoDB é um banco de dados NoSQL que utiliza a manipulação de documentos em formato JSON como esquema de tabelas.

A API escolhida para o protótipo foi a FaceAPI da Microsoft Azure, amplamente

testada e empregada em outras aplicações. A eficiência adquirida com requisições REST e a rápida resposta potencializa a escalabilidade dos recursos envolvidos para a construção de um sistema eficaz. A alocação de um banco de dados e a construção de um servidor para a hospedagem de um algoritmo de reconhecimento facial, ambos na nuvem, podem gerar um alto custo devido ao elevado número de requisições caso o sistema escale. A FaceAPI oferece uma quantidade de requisições gratuitas e, caso a elasticidade do sistema ocorra, a demanda extra custará um preço total proporcional ao uso.

5. Detalhes da Implementação

As tags $< video >, < canvas >$, que definem vídeos e gráficos respectivamente, são necessárias para o seu funcionamento do sistema. Isso se dá pelo fato de ao abrir a câmera do dispositivo (*notebook* ou *webcam*) para filmar o rosto do usuário, os dados captados (imagem) precisam ser renderizados por uma estrutura que os reconheça. E dentro do ambiente *web*, a *tag* $< video >$ é a mais adequada para isso, pois além de suportar a categoria de dado necessário, possui também atributos nativos de controle de vídeo. Sendo assim, para habilitar a câmera do dispositivo e renderizar os dados colhidos pela *tag* $< video >$ foi utilizado um objeto global via *JavaScript*. Esse objeto é o **window.navigator.mediaDevices** que identifica, através do navegador, quais dispositivos de captura de mídia estão presentes fisicamente no computador utilizado. A maioria dos navegadores (do inglês *browser*) atuais, exceto o *Explorer*, dão suporte para a utilização desse objeto viabilizando então, o uso dessa aplicação [CanIUse 2022].

Portanto, ao verificar se o navegador utilizado possui suporte ao objeto *navigator*, o processo de identificação dos dispositivos disponíveis será iniciado. Com o retorno constando os dispositivos de mídia acessíveis pelo objeto *navigator.mediaDevices*, a utilização do método **getUserMedia()** com o parâmetro de entrada **video: true** fará uma solicitação ao usuário para a permissão da abertura da câmera identificada do dispositivo. Com permissão concedida, o processo de captura do vídeo pela câmera do dispositivo será iniciado. Para o usuário visualizar, a *tag* $< video >$ ficará responsável por renderizar (com o atributo global *src*) o conteúdo do objeto retornado (**videoStream**) pelo método **getUserMedia()** [Matoso 2020].

Todavia, o uso desse vídeo não é suficiente para a aplicação. O FaceAPI da Microsoft Azure tem como parâmetro de entrada uma imagem de um rosto/face através de uma URL ou de um arquivo binário. Então, com a utilização da tag $< canvas >$ é possível capturar um frame atual do vídeo em execução que representa um screenshot de um trecho do vídeo que poderá ser convertido para qualquer formato de imagem como (.jpeg ou .png). Com essa conversão, automaticamente todas as características de uma imagem são adquiridas e seus respectivos dados em binário (que representam esta imagem) poderão ser codificados em outra categoria de dado que facilitará a sua leitura. No sistema criado foi utilizada a codificação Base64, comumente usada quando existe uma necessidade de transferência e armazenamento de dados binários para um dispositivo e para a compreensão por métodos e funções.

A vantagem desta codificação é que ela proporciona a redução do conteúdo disposto em binário (01110...) puro para grupos de textos que mapeiam a sequência correspondente. Nativamente a tag $< img >$ em sua propriedade global *src* consegue incorporar esta codificação para o reconhecimento de uma imagem [DelfStack 2022]. Então,

basta usar desta característica para transformar essa imagem codificada em um arquivo binário usando o construtor Blob() da linguagem *JavaScript*. Um objeto do tipo Blob representa um arquivo binário com dados imutáveis e detém características semelhantes de um objeto File. Tal fato contribui para que a API consiga interpretar as imagens enviadas, bastando apenas utilizar o cabeçalho do tipo application/octet-stream, necessário para transferir arquivos binários, na requisição.

Com os dados recebidos, a API retorna um JSON contendo todas as informações detectadas através dos marcadores faciais (LandMarks) contidos na imagem da face do usuário. Essas são organizadas em campos (do inglês *fields*) e são um pré requisito para o processo de verificação que ocorre durante a autenticação. Isso porque o campo FaceID define a localização no servidor das características faciais detectada e é o parâmetro de entrada no processo de verificação.

A verificação consiste na comparação de duas faces e define se são semelhantes ou não. Logo, com a entrada de dois identificadores faciais (FaceIDs) a API analisa as características faciais encontradas e retorna um JSON contendo dois campos que definem o nível de semelhança. Os dois campos isIdentical e confidence determinam se as faces comparadas pertencem a mesma pessoa e o quanto percentualmente essas se assemelham, respectivamente. Porém, há um vínculo entre esses dois campos, pois quando o nível de confiança (confidence) é maior que 50% (0,5) consequentemente o isIdentical é verdadeiro, caso menor, falso. Portanto, como os dois são utilizados pelo servidor da aplicação e são ambíguos, para o processo de autenticação ocorrer o nível de confiança exigido deve ser maior que 60%. Esse acréscimo de 10% foi definido arbitrariamente com o intuito de elevar o crédito da aplicação tornando-a mais exigente no processo de autenticação.

6. Experimentos e Resultados

Com o sistema desenvolvido, foi iniciada a fase de testes para a validação do seu uso. Como expresso nos capítulos anteriores, o sistema foi construído localmente, mas a sua base de dados e a *API* foram alocadas em nuvem. Sendo assim, as métricas mais relevantes são o tempo de resposta da aplicação e o nível de confiança (acurácia que indica a similaridade entre dois rostos) no processo de verificação, visto que os dados e a *API* estão alocados de forma remota.

Sendo assim, buscou-se com os testes a melhor maneira de abstrair da *API* uma maior confiabilidade no processo de verificação, e o aumento da rapidez no processo completo de autenticação sem que isso prejudicasse a integridade do sistema. A Microsoft Azure não disponibiliza o código do FaceAPI, ou seja, não é Open Source. Com isso, não foi possível determinar formas de melhorias com base algorítmica da *API*, então, os testes realizados consistiram em mudar parâmetros de modelo de reconhecimento e uso de formatos diferentes de imagens para averiguar a eficiência e o tempo de resposta.

A título de ilustração, a Tabela 2 mostra os parâmetros necessários para o funcionamento básico das operações de detecção e verificação da FaceAPI e destaca o parâmetro de modelo de reconhecimento analisado durante a fase de experimentos.

6.1. Experimento 1: Mudança no formato de imagem

É notório que ao escolher o formato PNG existe uma garantia na melhoria da qualidade da imagem capturada, que ajudaria, talvez, a API a capturar com maior eficiência uma

Operação	Parâmetro	Valor
detecção/verificação	detectionModel	detection01
detecção/verificação	**recognitionModel**	**recognition01**
detecção	returnFaceAttributes	age, gender, smile, emotion

Tabela 1. Parâmetros de entrada utilizados

quantidade maior de marcadores faciais dado as circunstâncias mínimas de iluminação. Entretanto, ao escolher esse formato, percebeu-se que a imagem codificada possuía um tamanho significativamente maior se comparado com a codificação de um frame no formato JPEG. Por exemplo, ao analisar um documento com a imagem PNG codificada percebeu-se que esse possuía em torno de 2 a 2,5 megabytes já com o formato JPEG houve uma redução de 250 a 320 kilobytes.

Portanto, a discussão levantada consistiu na hipótese de que talvez ao usar o formato PNG o nível de confiabilidade no processo de verificação aumentaria e, apesar da possível lentidão no sistema, a autenticação seria mais segura.

Para isso, a metodologia empregada para essa fase de testes consistiu no cadastro de 5 pessoas diferentes com a imagem de suas faces no formato PNG e posteriormente, para as mesmas, um recadastro, porém, utilizando o formato JPEG. Vale destacar que o modelo utilizado de reconhecimento facial foi o recognition01, padrão da API, e houve 3 tentativas por pessoa para que o nível de confiança fosse maior que 60% e o processo de autenticação ocorresse.

As Tabelas 2 e 3) destacam o nível de confiabilidade e o tempo total gasto no processo completo de autenticação (credenciais + reconhecimento facial).

Pessoa	Autenticação	Confiabilidade (%)	Tempo Total Gasto (S)
P1	Sim	84,78	18,03
P2	Sim	79,52	18,63
P3	Sim	80,33	19,02
P4	Sim	79,57	17,93
P5	Sim	82,94	17,87

Tabela 2. Formato: PNG - Modelo: Recognition01

Pessoa	Autenticação	Confiabilidade (%)	Tempo Total Gasto (S)
P1	Sim	84,52	7,56
P2	Sim	78,62	8,61
P3	Sim	81,28	8,43
P4	Sim	76,39	7,78
P5	Sim	83,67	8,49

Tabela 3. Formato: JPEG - Modelo: Recognition01

Como podemos perceber nas tabelas acima, a premissa levantada não se comprovou verdadeira, pois os níveis de confiabilidade para os dois formatos (JPEG e PNG) mantiveram-se próximos. Ademais, podemos constatar com a Figura 1 que a média entre os níveis de confiabilidade estão próximos e, como previsto, o tempo total gasto por cada autenticação é consideravelmente maior ao utilizar o formato PNG.

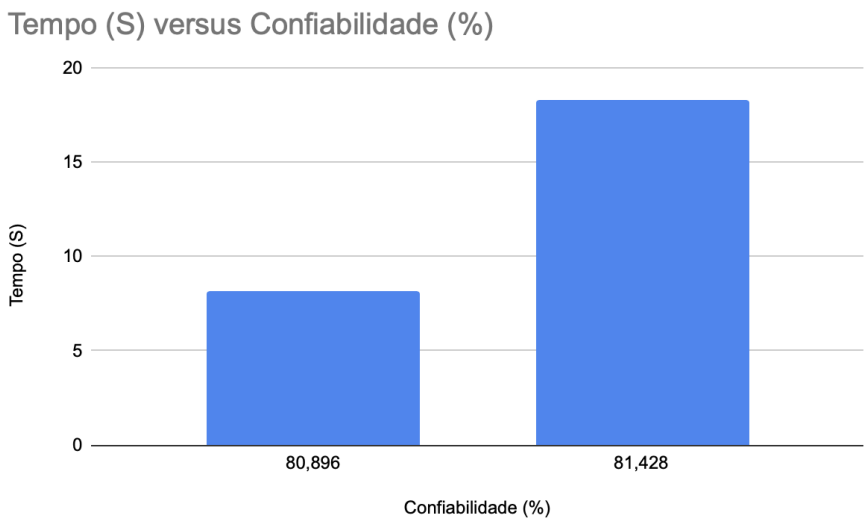

Figura 1. Gráfico: Tempo Gasto x Confiabilidade

Pelo gráfico acima podemos destacar que para alguns casos o tempo médio gasto é cerca de 2 a 2,5 vezes maior. Dois fatores que convém mencionar são que nos testes realizados ambos os processos, cadastro e autenticação, foram realizados no mesmo dia e as pequenas diferenças talvez se deram por outros fatores como, por exemplo, a iluminação, posição do rosto ou alterações nas expressões faciais como um sorriso.

Portanto, verificou-se que não existe uma vantagem substancial ao usar uma imagem no formato PNG, pois, mesmo essa garantindo uma maior qualidade, o nível de confiabilidade manteve-se próximo e como o tempo total gasto foi maior, preferiu-se utilizar o formato JPEG para garantir uma melhor experiência de usuário.

6.2. Experimento 2: Análise da divergência temporal entre imagens

Todavia, como expresso, no teste efetuado ambos os processos (cadastro e autenticação) foram realizados no mesmo dia, mas se imaginarmos que conforme o tempo passa as expressões faciais mudam não seria um absurdo considerarmos que uma foto armazenada no banco de dados há algum tempo atrás divergiria de uma foto capturada tempos depois. Isso porque algumas expressões faciais como o corte de cabelo, sorriso, barba e até rugas podem estar diferentes com o passar do tempo.

Sendo assim, a segunda fase de testes consistiu na análise da confiabilidade, no processo de autenticação, utilizando um intervalo de tempo maior. Ou seja, com a mesma imagem cadastrada no banco de dados no formato JPEG buscou-se validar se essa seria suficiente para a verificação 30 ou mais dias depois dado que possíveis mudanças faciais ocorreriam.

Foram utilizadas as mesmas 5 pessoas, 3 tentativas, exigência mínima de um nível de confiabilidade acima de 60% e o mesmo modelo de reconhecimento, o recognition01.

Analisando a Tabela 4, podemos observar que houve uma queda no nível de confiabilidade após mais de 30 dias passados. Tal fato confirma a premissa que, em um longo intervalo de tempo, a imagem armazenada no cadastro realizado em um tempo anterior pode ser insuficiente no processo de associação com outra imagem, tempos depois. Isso

Pessoa	Autenticação	Confiabilidade (%)	Tempo Total Gasto (S)
P1	Sim	72,48	7,23
P2	Não	54,83	16,98
P3	Não	57,66	17,42
P4	Sim	65,23	7,54
P5	Sim	70,41	8,03

Tabela 4. Tempo: Mais de 30 dias depois - Modelo: Recognition01

porque nesse futuro as possíveis mudanças nas expressões faciais podem ter ocorrido e a imagem capturada durante a execução da gravação será divergente.

Portanto, a partir das análises podemos inferir que a queda da confiabilidade pode ser um problema que pode se agravar ou se manter com o passar do tempo. Então, faz-se necessário utilizar técnicas que consigam minimizar esse problema. Uma ideia interessante seria sempre atualizar a foto contida na base de dados pela foto mais recente usada no processo de autenticação. Porém, não conseguiríamos definir se essa troca seria realizada em tempo hábil, pois se o usuário, devido às suas mudanças nas expressões faciais, tentar autenticar no sistema após um longo tempo esse pode não ser identificado.

Apesar disso, tal ideia não é totalmente descartável, ela pode entrar como um reforço para outras. Podemos observar também que em alguns casos de uso, por exemplo, com a necessidade de fazer a autenticação com algum acessório presente no rosto como a máscara, certamente o nível de confiabilidade cairia e também, talvez, o usuário não seria identificado.

6.3. Experimento 3: Comparação entre os modelos de reconhecimento

Com a recente pandemia de COVID-19 ocorrida após o ano de 2020, a Microsoft Azure buscou formas de melhoria do FaceAPI para cobrir os casos de identificação dos usuários portando máscaras faciais que foram usadas para o combate à propagação da doença. Sendo assim, a API ganhou um novo parâmetro de modelo denominado recognition04, que foi projetado para reconhecer os usuários aumentando a precisão em rostos que usam esse acessório.

Imagina-se que a diferença desse para os anteriores consiste no foco nos pontos de referência na parte superior do rosto (olhos, testa, sobrancelhas), mas não podemos afirmar por falta de informações. A Microsoft Azure garante em sua documentação que geralmente esse modelo é superior aos demais com o retorno de uma precisão mais elevada. Então, no esforço de garantir a efetividade da aplicação, com a redução na queda da confiabilidade no processo de verificação entre imagens armazenadas por um longo tempo e prever situações em que os usuários usam máscaras faciais, foram realizados novos testes repetindo-se as etapas do teste anterior, ou seja, com 5 pessoas, 3 tentativas, com as mesmas fotos faciais armazenadas por mais de 30 dias, mas utilizando esse novo modelo de reconhecimento, o recognition04.

Como podemos observar, Tabela 5, houve uma melhora na confiabilidade com o uso desse parâmetro. A Tabela 6 ilustra a comparação entre os modelos e confirmam que o novo modelo se utilizado como padrão, garante um nível de confiabilidade maior no processo de verificação independente das situações expostas nos parágrafos anteriores.

Pessoa	Autenticação	Confiabilidade (%)	Tempo Total Gasto (S)
P1	Sim	93,30	7,67
P2	Sim	73,47	8,05
P3	Sim	74,61	8,54
P4	Sim	81,62	7,63
P5	Sim	88,21	8,49

Tabela 5. Tempo: Mais de 30 dias depois - Modelo: Recognition04

Modelo	Confiabilidade (%)
01	64,12
04	82,24

Tabela 6. Média comparativa com imagens com mais de 30 dias

De modo a comprovar o ganho de eficiência utilizando o novo modelo, foram realizados novos testes utilizando as mesmas 5 pessoas, 3 tentativas por pessoa, porém com o cadastro e autenticação sendo realizados no mesmo dia. A Tabela 7 demonstra os resultados obtidos.

Pessoa	Autenticação	Confiabilidade (%)	Tempo Total Gasto (S)
P1	Sim	95,59	7,54
P2	Sim	89,08	7,97
P3	Sim	93,80	7,61
P4	Sim	91,87	8,23
P5	Sim	92,90	8,12

Tabela 7. Tempo: Mesmo Dia - Modelo: Recognition04

Logo, pela análise dos resultados acima podemos ressaltar as diferenças com a escolha do modelo recognition04 para as situações descritas. Em todos os casos, houve uma melhora comprovando que esse modelo deve ser usado como padrão. Partindo desse pressuposto, podemos retornar à ideia descrita nos parágrafos anteriores, onde é sugerido o processo de troca da imagem contida no banco a cada autenticação, e usá-la como reforço porque, com esse novo modelo, provavelmente o tempo hábil para essa troca acontecer será maior. Logo, as chances do sistema identificar o usuário, mesmo com algumas mudanças de suas expressões faciais, também serão maiores.

O Gráfico 2 reforça essa hipótese expondo a comparação entre os dois modelos para o processo de verificação com a autenticação realizada no mesmo dia e após 30 dias de cadastro.

Podemos observar que houve um ganho percentual médio maior que 15% no processo de autenticação após 30 dias de cadastro entre os dois modelos e para o cadastro e autenticação realizados no mesmo dia houve um ganho percentual médio de 12% aproximadamente. Logo, estimando uma possível curva com esses valores obtidos imagina-se uma queda de confiabilidade no processo de autenticação com o passar do tempo. Mas utilizando o modelo recognition04 o decrescimento pode ser menor e até se estabilizar com uma taxa de variação inferior se comparado com uma curva utilizando o modelo recognition01.

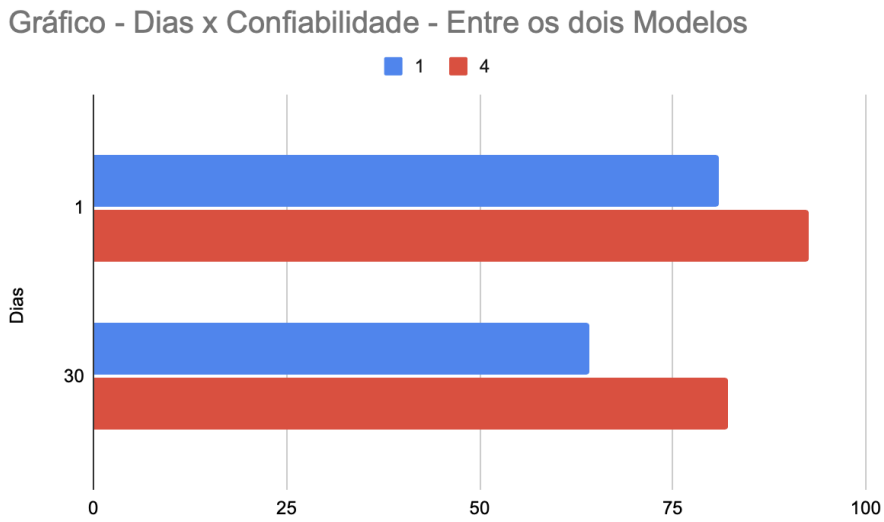

Figura 2. Gráfico: Dias x Confiabilidade

Por fim, após todos os testes realizados e suas análises, a versão final do código disponibilizada contará com todas as mudanças realizadas de modo a garantir aos desenvolvedores todas as melhorias adquiridas ao longo do processo de experimentação.

7. Conclusão

Neste trabalho, desenvolvemos uma aplicação *web* que utiliza o reconhecimento facial como um aprimorador no processo de autenticação, usando a FaceAPI da Microsoft Azure como referencial tecnológico. A aplicação foi testada e analisada sob diferentes aspectos, como o formato de imagem, o intervalo de tempo entre o cadastro e a autenticação, e o modelo de reconhecimento facial utilizado. Os resultados obtidos mostraram que a aplicação é capaz de identificar os usuários com um alto nível de confiabilidade e eficiência, usando o formato JPEG e o modelo recognition04. A aplicação também demonstrou ser robusta frente às possíveis mudanças nas expressões faciais dos usuários ao longo do tempo, bem como ao uso de máscaras faciais.

O trabalho contribui para a comunidade de desenvolvedores, disponibilizando os códigos e a documentação da aplicação, bem como as melhorias e adequações propostas. O trabalho também corrobora a ideia de que o reconhecimento facial é uma tecnologia promissora para aplicações *web*, pois oferece mais segurança e praticidade no processo de autenticação, além de se beneficiar da evolução dos algoritmos de inteligência artificial e visão computacional.

Como trabalho futuro, pretendemos explorar o uso da FaceAPI no modo de chamada de identificação, que permite usar um conjunto de imagens da mesma pessoa para a associação, aumentando a precisão na identificação das características semelhantes. Também planejamos implementar um mecanismo de troca da imagem armazenada no banco pela imagem capturada no processo de autenticação, visando atualizar as expressões faciais dos usuários e reduzir a taxa de decrescimento da curva de confiabilidade.

8. Agradecimento

Agradecemos ao CEFET-MG pelo apoio durante o desenvolvimento deste trabalho.

Referências

Avast (2019). 95% dos brasileiros estão usando senhas fracas. https://press.avast.com/pt-br/95-dos-brasileiros-estao-usando-senhas-fracas. Online; acessado em 12-Dezembro-2021.

BabelORG (2015). Babel is a javascript compiler. https://babeljs.io. Online; acessado em 22-Junho-2022.

CanIUse (2022). Window navigator. https://caniuse.com/?search=navigator.mediaDevices. Online; acessado em 15-Abril-2022.

Consultoria, F. (2016). Boas práticas no desenvolvimento de software e sistema web. https://fluxoconsultoria.poli.ufrj.br/blog/boas-praticas-desenvolvimento-software/. Online; acessado em 20-Abril-2022.

DelfStack (2022). Exibir imagens base64 em html. https://www.delftstack.com/pt/howto/html/html-img-base64/. Online; acessado em 09-Março-2022.

Fajardo, F. T. (2019). Desenvolvimento de uma aplicação web com detecção e reconhecimento facial. Santa Maria, RS.

Matoso, D. (2020). Como acessar câmera com javascript (frontal e traseira). https://www.webdevdrops.com/como-acessar-camera-com-javascript/. Online; acessado em 28-Novembro-2021.

Monteiro, L. (2001). A internet como meio de comunicaÇÃo possibilidades e limitaÇÕes. pages 27–37. PUC-Rio.

Njah, F. (2022). Understanding the javascript dom manipulations. https://sweetcode.io/understanding-the-javascript-dom-manipulations/. Online; acessado em 19-Junho-2022.

OWASP (2021). Owasp top ten. https://owasp.org/www-project-top-ten/. Online; acessado em 17-Janeiro-2022.

Prashaan Pillay, S. V. (2019). Foresight: Real time facial detection and recognition using webassembly and localized deep neural networks.

Ruiwen Li, Songjie Cai, T. S. (2018). Login authentication with facial gesture recognition.

TEG6, P. (2021). Reconhecimento facial em smartphones: é seguro e deve ser usado? https://l1nq.com/pfRJE. Online; acessado em 08-Dezembro-2021.

TiLT (2019). 5 em cada 10 brasileiros usam a mesma senha em diferentes contas na web. https://l1nq.com/Ob8Iu. Online; acessado em 08-Dezembro-2021.

Zucher, V. (2020). O que é padrão mvc? entenda arquitetura de softwares! https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc. Online; acessado em 21-Abril-2022.