RECOGSYS - Sistema de Controle de Acesso Utilizando Reconhecimento Facial

Pedro Henrique Ferreira da Costa Damião[1], Sandro Roberto Fernandes[2]

[1] IF Sudeste MG – Campus Juiz de Fora

[2]Núcleo de Informática – IF Sudeste MG – Campus Juiz de Fora

pedrohenriquefcd@gmail.com, sandro.fernandes@ifsudestemg.edu.br

Abstract. The *IoT has made the presence of technology increasing in our daily lives. However, some areas have shown such evolution in a slower way, as is the case of physical access control, and this deficit is even greater when we restrict searches to works that use facial recognition. Despite being a constant topic of research, you do not see many works or large-scale commercial products using this technology. Thus, this article aims to present the process of developing a physical access control application using facial recognition, called RECOGSYS. The system was developed based on the architecture of microservices and programming tools based on Javascript, from the application development to the facial recognition library. At the end of the research, a functional and scalable system of easy and intuitive use was obtained.*

Resumo. *A IoT tem deixado a presença da tecnologia cada vez maior em nosso cotidiano. Entretanto algumas áreas têm apresentado tal evolução de modo mais lento, como é o caso do controle de acesso físico, e esse déficit é maior ainda quando nos restringimos a trabalhos que utilizem reconhecimento facial. Apesar de ser um tema constante de pesquisa, não se vêm muitos trabalhos ou produtos comerciais em larga escala utilizando esta tecnologia. Assim, o presente artigo tem como objetivo apresentar o processo de desenvolvimento de uma aplicação de controle de acesso físico utilizando reconhecimento facial, denominada RECOGSYS. O sistema foi desenvolvido baseado na arquitetura de microsserviços e ferramentas de programação baseadas em Javascript, desde o desenvolvimento da aplicação à biblioteca de reconhecimento facial. Ao final da pesquisa obteve-se um sistema funcional e escalável de fácil e intuitiva utilização.*

1. Introdução

Com o advento da IoT[1], suas aplicações estão cada vez mais envolvendo a rotina das pessoas, preenchendo-a com inovação até mesmo nas tarefas mais triviais. Uma destas tarefas é o controle de acesso físico, para o qual contamos atualmente com diversas alternativas inteligentes [Malik et al. 2020].

[1] IoT, *Internet of Things* - Internet das Coisas: é a interconexão de objetos cotidianos com a internet.

A implementação de um controle de acesso físico visa regular o acesso de um determinado sujeito a um ambiente, mantendo os cinco principais pilares da segurança da informação em relação ao recinto: confidencialidade, disponibilidade, integridade autenticidade e não repúdio. Este processo inclui uma etapa de identificação, seguida pela autenticação e, por fim, a verificação da autorização do indivíduo para o acesso pretendido [Malik et al. 2020].

Os primeiros problemas no controle de acesso surgem com aqueles métodos onde nem sempre as etapas são executadas imediatamente uma após a outra como, por exemplo, o uso de chaves convencionais ou *tags* NFC[2]: apesar destas serem entregues a uma pessoa de confiança, podem posteriormente serem compartilhadas, quebrando assim o princípio da confidencialidade do ambiente a ser protegido e perdendo o atributo de não repúdio. Porém, estes problemas de fraude não são exclusivos de métodos mais simples: Conforme exposto por KRAKENFX (2021), é possível fraudar impressões digitais em sistemas mais modernos, utilizando materiais simples e baratos e com etapas que podem ser reproduzidas facilmente. Assim, conclui-se que "impressões digitais não devem ser consideradas como uma alternativa segura a senhas fortes" [KRAKENFX 2021].

Diante destes problemas, e da crescente disponibilidade de tecnologia, o reconhecimento facial tem ganhado muita atenção nos últimos anos. Apesar de algumas dificuldades, tal como a adaptação da tecnologia a ambientes não controlados, esta estratégia já adquiriu satisfatório grau de maturidade e apresenta bons resultados. Estes resultados são ainda mais satisfatórios quando utilizado aprendizado de máquina, sendo uma boa alternativa para a autenticação em sistemas de controle de acesso, principalmente porque é uma tecnologia que não precisa de contato, evitando assim a proliferação de vírus e bactérias [Adjabi et al. 2020].

2. Fundamentos Teóricos

A seguir são abordados e revisados os principais fundamentos teóricos essenciais para o pleno entendimento do trabalho realizado, das metodologias adotadas e dos trabalhos relacionados.

2.1. PACS

Um PACS, *Physical Access Control System* - Sistema de Controle de Acesso Físico, pode ser definido como uma aplicação computacional que gerencia e envolve todas as características de uma aplicação de controle de acesso físico que, por sua vez, de acordo com Ferreira (2003), pode ser definida como "toda e qualquer aplicação de procedimento ou uso de equipamentos com o objetivo de proteger ambientes, equipamentos ou informações cujo acesso deve ser restritos". Além disso, complementando a visão de PACS, Ferreira (2003), ressalta que essas aplicações podem utilizar chaves, crachás, vídeos, *smartcards*, biometria, dentre outros, e que a adoção de cada tecnologia, ou

[2] Tags NFC, *Near Field Communication* – Comunicação de Campo Próxima: são objetos com objetivo de transmitir dados gravados previamente para os dispositivos com a mesma tecnologia ao se aproximarem.

conjunto de tecnologias, está intrinsecamente ligada à importância do que está sendo protegido.

2.2. Identificação

A identificação é a etapa do processo de controle de acesso na qual é respondida à pergunta "Quem é você?". Ou seja, é nesta etapa que o usuário fornece sua credencial de acesso, tal como: nome, e-mail ou qualquer outra informação que possa identificá-lo. Entretanto, vale ressaltar que este processo não garante que o usuário seja realmente quem está dizendo ser [Zviran and Erlich 2006].

2.3. Autenticação

Já na etapa de autenticação, busca-se realizar a devida conferência da identificação do usuário. Este processo pode ser realizado utilizando uma senha pré-cadastrada, uma OTP[3], características biométricas, dentre outras abordagens [Zviran and Erlich 2006].

2.4. Autorização

Por fim, na etapa de autorização, são definidas as operações que o usuário pode desempenhar no sistema, uma vez que sua identidade já foi verificada [Zviran and Erlich 2006].

2.5. Microsserviços

De acordo com Lewis e Fowler (2014), "A arquitetura de microsserviços é uma abordagem para desenvolver uma aplicação na forma de diversos pequenos serviços, cada um rodando seu próprio processo e se comunicando entre si através de mecanismos leves". Esta arquitetura surge visando resolver um dos maiores problemas dos sistemas monolíticos[4]: manter a performance com o aumento dos dados a serem processados, ou seja, manter a escalabilidade. Entretanto, é importante ressaltar que a implementação desta arquitetura eleva consideravelmente a complexidade da aplicação e o nível de dificuldade para elaboração de testes funcionais[5] [Tapia et al. 2020].

3. Revisão Sistemática

O escopo da revisão sistemática baseia-se em encontrar artigos acadêmicos referentes à sistemas de controle de acesso que, preferencialmente, utilizem biometria facial para autenticação e que geraram conclusões relevantes ou sistemas funcionais. Desta forma, espera-se compreender o estado da arte deste tópico de pesquisa, além de solidificar bases para o desenvolvimento de uma nova ferramenta com este propósito.

---

[3] OTP, *One-time password* - Senha descartável: é uma senha gerada automaticamente destinada a ser utilizada em apenas uma autenticação

[4] Arquitetura de sistemas onde todos os componentes são executados em conjunto agrupados em um único processo

[5] Metodologia de teste de software baseada nos requisitos funcionais do sistema, testando seu funcionamento global

Para alcançar tais objetivos, primeiramente foram definidas as seguintes perguntas a serem respondidas:

1. Quais as técnicas de reconhecimento facial mais eficientes e seguras?
2. Quais as principais funcionalidades de um sistema de controle de acesso físico?
3. Quais os padrões de projeto mais adequados para garantir o desempenho, a escalabilidade e a segurança do sistema?

E, baseando-se nestas perguntas, foram desenvolvidas suas respectivas *strings* de busca:

1. "*efficient*" *AND* "*secure*" *AND* ("*technique*" *OR* "*method*") *AND* ("*face recognition*" *OR* "*facial recognition*")
2. "*physical*" *AND* "*access control system*"
3. "*design pattern*" *AND* "*scalability*" *AND* "*performance*" *AND* "*security*" *AND* "*system*"

Posteriormente, foram definidos os seguintes critérios para a busca de publicações:

- Publicações acadêmicas em formato de artigo
- Publicações em português, inglês ou espanhol
- Artigos publicados entre 2019 e 2021 (três anos anteriores à pesquisa)

Por fim, devido a sua acessibilidade às publicações completas e vastidão de conteúdo, o Portal de Periódicos CAPES foi definido como indexador de busca do trabalho. Utilizando as *strings* de busca definidas anteriormente, foram obtidos, respectivamente, 391, 152 e 60 artigos como resultado inicial. Destes, após leitura do título e das palavras-chave, foram selecionadas, respectivamente, 9, 12 e 18 publicações para análise do abstract. Passada esta etapa restaram, respectivamente, 7, 6 e 6 artigos para análise detalhada, que selecionou, respectivamente, 5, 4 e 3 artigos relevantes para este trabalho.

4. Trabalhos Relacionados

As pesquisas acerca de PACS e reconhecimento facial têm gerado muitos trabalhos ultimamente, seja apenas focado em uma das duas propostas, seja unindo ambas.

Um desses trabalhos é o de Petrakis et al. (2020), que desenvolve um PACS na arquitetura de microsserviços, com seus serviços rodando na nuvem privada das respectivas instituições usuárias, e enviando dados analíticos para a nuvem pública. No quesito de autenticação, os autores utilizaram a conexão *bluetooth*[6] de dispositivos móveis.

Também pode-se citar os trabalhos de Abozaid et al. (2019) e

[6] Protocolo de comunicação de curto alcance e baixo consumo de energia que permite dois dispositivos trocarem informações entre si sem cabos

Si et al. (2020), que adotam a fusão de diferentes técnicas de autenticação. Eles adotam, respectivamente, reconhecimento de voz com reconhecimento facial e reconhecimento de marcha com reconhecimento facial.

Além disso, analisando pesquisas focadas apenas em reconhecimento facial, pode-se destacar dois trabalhos: Kang (2019) que compara diversos modelos de redes neurais em reconhecimento facial para determinar sua acurácia. Enquanto isso, Sabharwal et al. (2019) apresenta uma análise quanto à capacidade de se reconhecer rostos após procedimentos cirúrgicos.

5. Metodologia

5.1. Detecção e Reconhecimento Facial

O estado da arte de Visão Computacional [Fard and Hashemi 2020] aponta para o uso de CNN, *Convolutional Neural Networks* - Redes Neurais Convolucionais, visto que esta abordagem comumente oferece melhores resultados aos principais desafios desta área de pesquisa: iluminação, variação de posição, envelhecimento, uso de acessórios como óculos e bonés, dentre outros.

Considerando as vantagens de uma CNN, e restringindo-se a soluções que utilizem a linguagem Javascript, optou-se por utilizar a Face-API.js, uma API*, Application Programming Interface* – Interface de Programação de Aplicações, em Javascript para reconhecimento facial implementada a partir do núcleo do Tensorflow.js, que conta com uma detecção leve e rápida de 68 pontos no rosto [Mühler 2019].

Para a detecção facial, foi utilizado o modelo *Tiny Face Detector*, de apenas 190Kb. Seu objetivo é se manter performático consumindo menos recursos computacionais, podendo assim ser utilizado em diversas plataformas [Mühler 2019].

Para a descrição do rosto em um vetor de 128 posições, a Face-API.js utiliza uma arquitetura similar à ResNet-34 para desenvolver uma rede neural semelhante à biblioteca dlib quando utilizada para reconhecimento facial. Esta implementação alcança 99.38% de acurácia no LFW, *Labeled Faces in the Wil*d – Rostos Rotulados em Ambientes Não-Controlados, renomado repositório para o estudo de reconhecimento facial [Mühler 2019].

Por fim, dados os descritores, utiliza-se o cálculo da Distância Euclidiana para obter-se o percentual de similaridade entre duas faces [Mühler 2019].

5.2. Tecnologias

Abaixo são descritas as principais tecnologias utilizadas no desenvolvimento do sistema, assim como algumas de suas vantagens e a plataforma a que se propõe trabalhar.

5.2.1. Node.js

O Node.js é um software interpretador de código Javascript assíncrono orientado a eventos, permitindo o desenvolvimento de aplicações escaláveis de rede [Node.js 2019].

Desenvolvido em 2009, foi o primeiro ambiente com o intuito de permitir a execução de código Javascript no lado do servidor. Além disso, ele também implementa a execução em uma única thread, diminuindo a alocação de CPU e memória RAM em relação a aplicações desenvolvidas em C#, Java e PHP, por exemplo [Lenon 2018].

5.2.2. React

O React é uma biblioteca [7] Javascript para construção de interfaces de usuário multiplataforma, utilizando HTML [8] e CSS [9] embutido no Javascript. Com auxílio de outras bibliotecas, pode ser renderizado para uso como aplicativo móvel e para realidade virtual, dentre outros, além do uso na web [Valle and Dieminger 2022]. Pode-se citar também que o React trabalha muito bem com outras bibliotecas e *frameworks* [10], pois utiliza uma lógica de separar a interface em diversos componentes, diferentemente de alguns frameworks, tal como o Vue.js, que utiliza a lógica de *templates* [11] [React 2017].

5.2.3. Electron

O Electron é um framework que permite a compilação de um único código para aplicações nativas. Utilizando apenas HTML, CSS e Javascript, o resultado é um programa nativo para os sistemas Windows, Linux e Mac OS.

5.2.4. Typescript

O Typescript é uma linguagem de programação e um superconjunto sintático estrito de Javascript, adicionando tipagem estática na linguagem. Isto significa que todo código Javascript pode ser executado como Typescript, desde que devidamente tipado [Nascimento 2021]. Ou seja, podemos desenvolver programas em Node.js, React e Electron utilizando Typescript sem empecilhos.

5.2.5. PostgreSQL

O PostgreSQL é o mais avançado Sistema Gerenciador de Banco de Dados objeto-relacional de código aberto do mundo [PostgreSQL 2021]. Seu sucesso se deve, dentre outras características, ao oferecimento de suporte à gatilhos, visões e linguagem procedural, além da facilidade de uso.

Se tratando especificamente das operações de reconhecimento facial, o PostgreSQL se destaca ainda mais devido à facilidade de se armazenar vetores, resultando em um trabalho mais fluido dos descritores de características a serem salvos.

---

[7] Biblioteca, em programação, é um conjunto de códigos voltados a resolver um tipo de problema

[8] HTML é uma linguagem de marcação utilizada para construir páginas na web

[9] CSS é um mecanismo para se adicionar estilo à estrutura criada com HTML

[10] Framework, em programação, é uma abstração que possui várias funcionalidades prontas

[11] Template, em programação web, é um modelo de interface a ser seguido

5.2.6. Redis

O REDIS é um banco de dados do tipo chave-valor onde as informações são armazenadas em memória, permitindo operações de escrita e leitura com performance superior a outros tipos de bancos de dados. Devido às suas características, suas principais aplicações são armazenamento de cachê, sessões e mensageria.

5.3. Boas Práticas de Desenvolvimento

Abaixo são definidas algumas ferramentas utilizadas durante o desenvolvimento do sistema visando garantir a produção de código com boa qualidade e padronizado.

5.3.1. Git

O Git é um sistema de controle de versões de arquivos. Seu principal objetivo é manter a organização e compatibilidade entre as alterações realizadas, mesmo com diversas pessoas trabalhando ao mesmo tempo, seja no mesmo arquivo, seja em arquivos diferentes [Schmitz 2015].

Apesar deste projeto ser desenvolvido por uma única pessoa, o uso do Git se mostra necessário para manter a organização e melhor visualização das tarefas realizadas. Visando maximizar este objetivo, foi utilizado o padrão TBD, *Trunk-Based Development* – Desenvolvimento Baseado em Tronco, um padrão de organização das ramificações do repositório de trabalho, que consiste na criação de uma *branch* principal, que é alimentada por *branches* auxiliares focadas em funcionalidades específicas, e pode originar release *branches* quando é necessário realizar uma publicação do sistema.

5.3.2. ESLint

O ESLint é uma ferramenta que realiza análise de código Javascript, baseando-se em regras previamente definidas, visando procurar e corrigir possíveis padrões problemáticos tal como variáveis e importações não utilizadas e falta de padronização na nomenclatura de variáveis [Corrêa 2020]. Consequentemente, o uso do ESLint também auxilia no trabalho em grupo, garantindo que os padrões de desenvolvimento sejam mantidos por todos os envolvidos, assim como facilitando as revisitas dos desenvolvedores ao seu próprio código.

5.3.3. Prettier

Tal como o ESLint, o objetivo do Prettier é manter a padronização do código produzido e facilitar o trabalho em grupo. Entretanto, o Prettier atua como um formatador de código, garantindo que, por exemplo, o código seja indentado apenas com espaços, não seja ultrapassado um certo limite de caracteres em cada linha, o estilo de aspas, dentre outros.

6. Desenvolvimento

Através da utilização das tecnologias e metodologias apresentadas, foi possível o desenvolvimento de um sistema que atende às necessidades previstas inicialmente. A seguir são apresentadas as premissas do sistema e detalhes do processo de planejamento e desenvolvimento.

6.1. Apresentação do sistema

O nome do sistema surgiu da junção das palavras inglesas "*Recognition*" e "*System*", que traduzidas para o português brasileiro significam, respectivamente, "Reconhecimento" e "Sistema", realizando alusão ao objetivo do projeto: desenvolver um sistema de controle de acesso com reconhecimento facial.

Além da criação do nome, foi desenvolvido um logotipo para o sistema, apresentado na Figura 1. As cores utilizadas foram embasadas na teoria da psicologia das cores, visando complementar o propósito do sistema. De acordo com esta teoria, o preto, utilizado no fundo, remete a poder e formalidade, enquanto o Zomp, um tom de verde utilizado nas letras, remete à segurança [Printi 2021].

Figura 1. Logotipo do Sistema. Fonte: O Autor

6.2. Modelagem

Após a definição do escopo do sistema e seu propósito, iniciou-se a etapa de modelagem do sistema. Esta etapa é importante para se ratificar a arquitetura a ser utilizada, assim como findar bases para o processo de desenvolvimento propriamente dito.

6.2.1. Diagrama de Casos de Uso

A primeira etapa na modelagem foi o desenvolvimento do diagrama de casos de uso, apresentado na Figura 2. Ele é o responsável por gerenciar as expectativas dos usuários em relação às funcionalidades do sistema, atuando como referência em todo o planejamento.

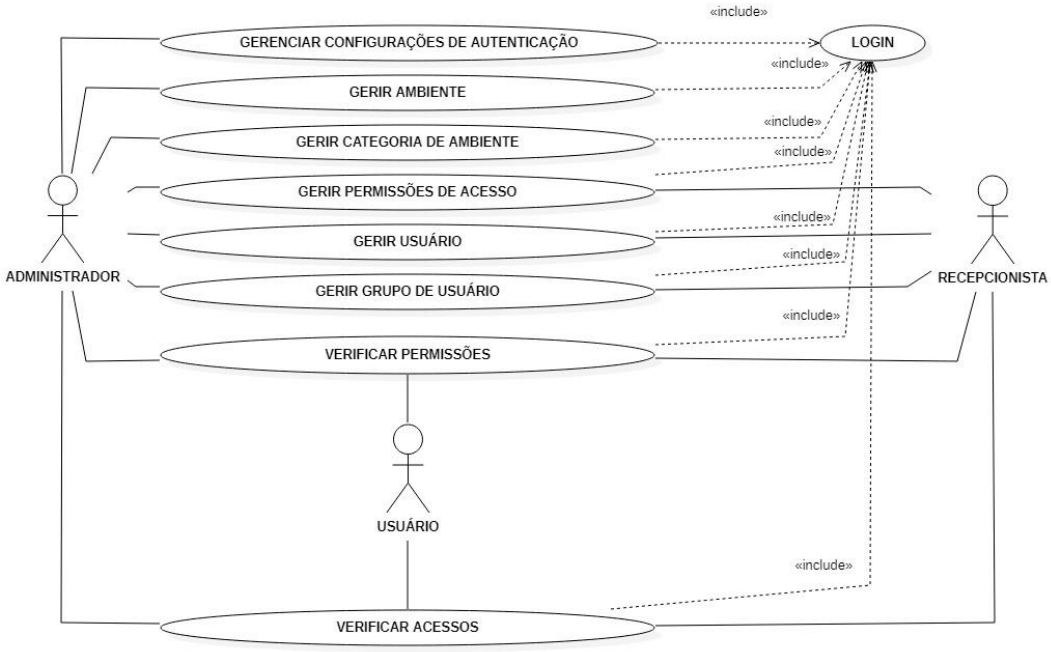

Figura 2. Diagrama de caso de uso. Fonte: O Autor.

6.2.2. Diagrama de Fluxo

Já o segundo passo dentro da etapa de modelagem foi o desenvolvimento do fluxo principal do sistema, apresentado na Figura 3. Ele é a base para entender como os dados trafegam dentro do sistema, assim como ratificar a divisão e a função de cada módulo que compõe o sistema.

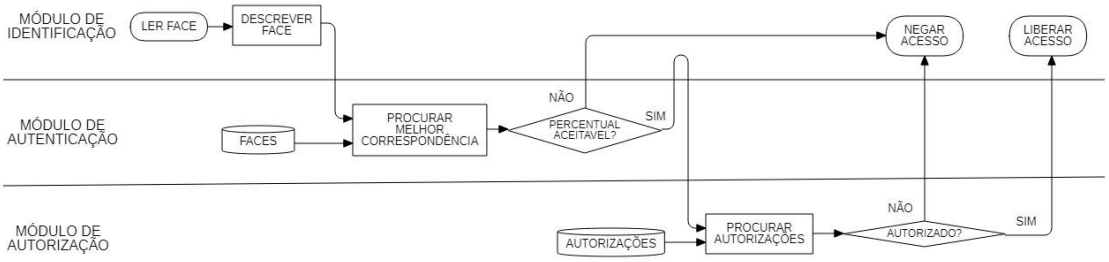

Figura 3**. Diagrama de fluxo. Fonte: O Autor.**

Visando reforçar a arquitetura de microsserviços a ser adotada no desenvolvimento do sistema, vale ressaltar os seguintes pontos neste diagrama:

1. A separação de responsabilidades de cada módulo, visando independência entre os serviços
2. A separação das bases de dados, também visando independência entre os serviços
3. O retorno do resultado da autenticação, ao módulo de identificação antes de ser enviada ao módulo de autorização, ressaltando a separação de responsabilidade dos serviços

6.2.3. Diagrama Entidade Relacionamento

Por fim, considerando a arquitetura adotada e as funcionalidades a serem implementadas, foi desenvolvido o diagrama Entidade Relacionamento de ambos os bancos de dados do sistema. Na Figura 4 podemos conferir o diagrama do banco de dados do módulo de autenticação, enquanto na Figura 5 podemos conferir o diagrama referente ao módulo de Autorização.

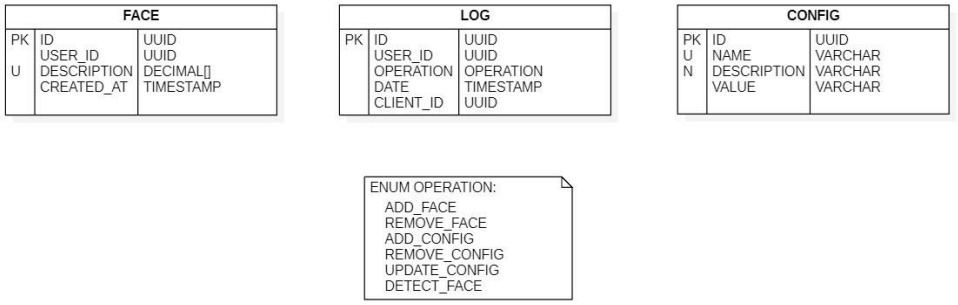

Figura 5. Diagrama Entidade Relacionamento do Módulo de Autenticação. Fonte: O Autor.

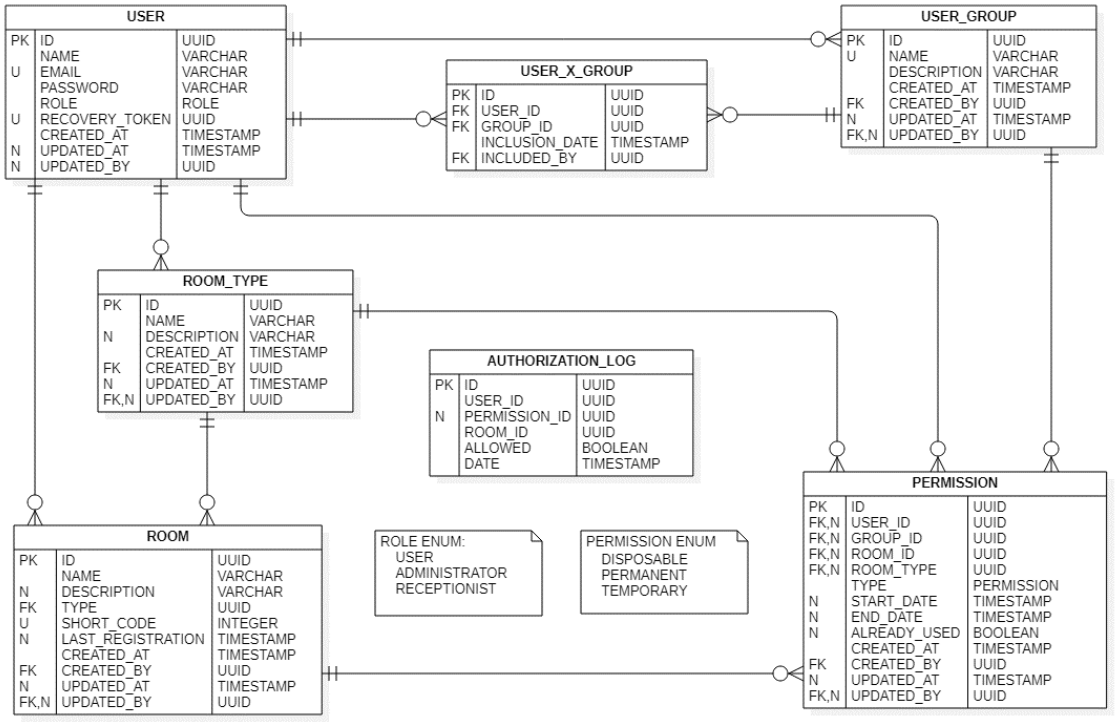

Figura 4. Diagrama Entidade Relacionamento do Módulo de Autorização. Fonte: O Autor.

Vale ressaltar que a ausência da implementação das chaves estrangeiras nas tabelas LOG, no módulo de autenticação, e AUTHORIZATION_LOG, no módulo de autorização, foram intencionais, uma vez que, devido ao contexto do dado e para fins históricos, a intenção é que ele não seja deletado em cascata no caso da deleção de uma face, usuário e afins.

6.3. *Back-end*

O sistema conta com dois serviços no back-end: um para o módulo de autenticação e outro para o módulo de autorização. Ambos são API's RESTful[12], utilizando o protocolo HTTPS, *Hyper Text Transfer Protocol Secure* – Protocolo de Transferência de Hipertexto Seguro para trafegar dados em padrão JSON, desenvolvidas em Node.js e utilizando banco de dados PostgreSQL como Sistema Gerenciador de Banco de Dados. Além das funcionalidades, podemos citar duas diferenças entre elas:

1. Autorização de acesso: A API de autenticação utiliza um token pré-definido enviado no cabeçalho das requisições para autorizar, ou não, o acesso. Enquanto isso, a API de autorização utiliza cookies de sessão enviados via cabeçalho da requisição, e definidos pelo servidor
2. Estado: A API de autenticação é *stateless*[13], enquanto a API de autorização é *stateful*[14], armazenando os dados de sessão em um Redis, conforme pode ser visualizado na Figura 6, que apresenta a função de gerenciamento de sessões na API de autorização.

```
function SessionMiddleware(
  request: Request,
  response: Response,
  next: NextFunction
): Promise<void>{
  session({
    store: new RedisStore({ client: RedisClient }),
    secret: process.env.SESSION_SECRET || 'local-session-secret',
    saveUninitialized: false,
    resave: false,
    cookie: {
      secure: false,
      httpOnly: false,
      maxAge: 30 * 60 * 1000,
      sameSite: 'lax',
    },
  });
}
```

Figura 6. Função de Gerenciamento das sessões no Módulo de Autorização. Fonte: O Autor

[12] Aplicação que implementa a arquitetura REST, *Representational State Transfer – Transferência Representacional de Estado*
[13] Tipo de aplicação no qual não são armazenadas informações referentes às transações anteriores
[14] Tipo de aplicação na qual são armazenadas informações referentes às transações anteriores

Abaixo podemos conferir as informações de todos os *endpoints* de ambas API's. As informações da Tabela 1 são referentes à API de autenticação, enquanto na Tabela 2 são referentes à API de autorização.

Tabela 1. *Endpoints* do serviço de autenticação. Fonte: O Autor.

Método	Rota	Descrição	Restrita?
GET	/faces	Lista dados de todas as faces cadastradas	X
POST	/faces	Cria um registro de face	X
GET	/faces/{user_id}	Retorna dados da face indicada	X
DELETE	/faces/{user_id}	Exclui todas as faces do usuário indicado	X
GET	/configs	Lista todas as configurações	X
POST	/configs	Cria uma configuração	X
GET	/configs/{config_name}	Retorna a configuração indicada	X
UPDATE	/configs/{config_id}	Atualiza a configuração Indicada	X
DELETE	/configs/{config_id}	Exclui a configuração indicada	X
POST	/detect	Retorna a melhor combinação de face	X
GET	/logs	Lista todos os logs salvos	X

Tabela 2. *Endpoints* do serviço de autorização. Fonte: O Autor.

Método	Rota	Descrição	Restrita?
GET	/login	Realiza o login do usuário	
GET	/logout	Realiza o logout do usuário	X
POST	/forgotpassword	Recebe o e-mail do usuário e envia um link para cadastro de nova senha	
POST	/resetpassword	Salva a nova senha do usuário	
GET	/permissions	Lista as permissões de acordo com o filtro	X
GET	/permissions/{id}	Lista a permissão especificada	X
POST	/permissions	Cria uma permissão	X
UPDATE	/permissions/{id}	Edita uma permissão existente	X
DELETE	/permissions/{id}	Exclui uma permissão existente	X
GET	/accessattempts	Lista as tentativas de acesso de acordo com o filtro	X
GET	/users	Lista todos os usuários	X
GET	/users/{id}	Lista o usuário especificado	X
POST	/users	Cria um usuário	X
UPDATE	/users/{id}	Atualiza o usuário especificado	X
DELETE	/users/{id}	Exclui o usuário especificado	X
GET	/users/{id}/faces	Lista informações de todas as faces do usuário especificado	X
POST	/users/{id}/faces	Cria uma face para o usuário especificado	X
DELETE	/users/{id}/faces	Exclui as faces do usuário especificado	X
GET	/usergroups	Lista todos os grupos de usuários	X
GET	/usergroups/{id}	Lista o grupo de usuários especificado	X

POST	/usergroups	Cria um grupo de usuários	X
UPDATE	/usergroups/{id}	Atualiza o grupo de usuários especificado	X
DELETE	/usergroups/{id}	Exclui o grupo de usuários especificado	X
GET	/usergroups/{id}/users	Lista todos os usuários do grupo de usuários indicado	X
POST	/usergroups/{id}/users	Adiciona um usuário a um grupo de usuários	X
DELETE	/usergroups/{id}/users/{id}	Exclui o usuário especificado do grupo de usuários especificado	X
GET	/roomtypes	Lista todas as categorias de ambientes	X
GET	/usergroups/{id}	Lista a categoria de ambiente especificada	X
POST	/roomtypes	Cria uma categoria de ambiente	X
UPDATE	/roomtypes/{id}	Atualiza a categoria de ambiente especificada	X
DELETE	/roomtypes/{id}	Exclui a categoria de ambiente especificada	X
GET	/rooms	Lista todos os ambientes	X
GET	/rooms/{id}	Lista o ambiente especificado	X
POST	/rooms	Cria um ambiente	X
UPDATE	/rooms/{id}	Atualiza o ambiente especificado	X
DELETE	/rooms/{id}	Exclui o ambiente especificado	X
GET	/configs	Lista as configurações do módulo de autenticação	X
GET	/configs/{name}	Lista a configuração especificada	X
POST	/configs	Cria uma configuração no módulo de autenticação	X
UPDATE	/configs/{id}	Atualiza a configuração especificada no módulo de autenticação	X
DELETE	/configs/{id}	Exclui a configuração especificada no módulo de autenticação	X
POST	/isauthorized	Verifica autorização do usuário para acessar ambiente	
POST	/configureclient	Configura um módulo de identificação cliente	

Além disso, é importante citar que ambas as API's utilizam os *design patterns Controller + Service + Repository*. Ou seja, temos a camada de *Controllers*, responsáveis pela conexão do Front-end com o Back-end, a camada de *Services*, responsáveis pela aplicação das regras de negócio, que são implementadas também a nível de banco de dados e, por fim, a camada de *Repositories*, responsáveis pela conexão com o banco de dados. A utilização destes *design patterns* permite maior desacoplamento entre as partes do sistema, facilitando manutenções e evoluções futuras. Esta implementação pode ser conferida na Figura 7, que apresenta a função de cadastro de face da API de autenticação nas três camadas.

```
// Controller
async store(request: Request, response: Response): Promise<void> {
  isAuth(['AUTHORIZATION_SERVER'], request);
  const { userId, description, clientId } = request.body;
  const face = await FaceService.create({
    userId,
    description,
    clientId,
  });
  response.json(face);
}

// Service
async create(
  {userId, description, clientId}: { userId: string; description: number[]; clientId: string;
}): Promise<Face> {
  if (!userId)  throw new AppError('User ID is required', 400);
  if (!description) throw new AppError('Description is required', 400);
  if (description.length =/= 128) throw new AppError('Description must be 128 positions', 400);
  const face = await FaceRepository.findByDescription(description);
  if (face) throw new AppError('Face already exists', 400);
  Matcher.updateFaces();
  const newFace = FaceRepository.create({ userId, description});
  LogRepository.create('ADD_FACE', userId, clientId);
  return newFace;
}

// Repository
async create({ userId, description }: { userId: string; description: number[] }): Promise<Face> {
  const { rows } = await client.query( `
  INSERT INTO public.face(user_id, description)
  VALUES ($1, $2)
  RETURNING id, user_id;`,
    [userId, description]
  );
  const [result] = rows;

  return result as Face;
}
```

Figura 7. Código da função de criação de face na API de autenticação. Fonte: O Autor

6.4. *Front-end*

Já na questão de interação direta com o usuário, o sistema conta com dois serviços: um para o módulo de identificação e outro para o módulo de autorização, que serão melhor descritos abaixo.

6.4.1. Módulo de Identificação

Este módulo foi desenvolvido utilizando a tecnologia Electron, e otimizado para ser executado em um Raspberry Pi, visando utilizar a GPIO do mesmo para acionar uma fechadura elétrica, conforme esquema apresentado no modelo da Figura 8 e no protótipo desenvolvido apresentado na Figura 9.

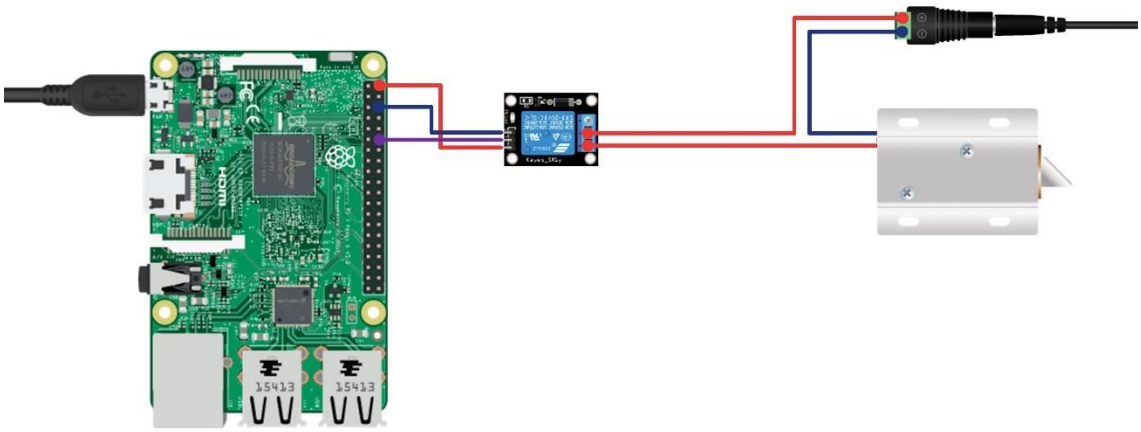

Figura 8. Esquematização da conexão do Raspberry Pi com a fechadura elétrica. Fonte: O Autor.

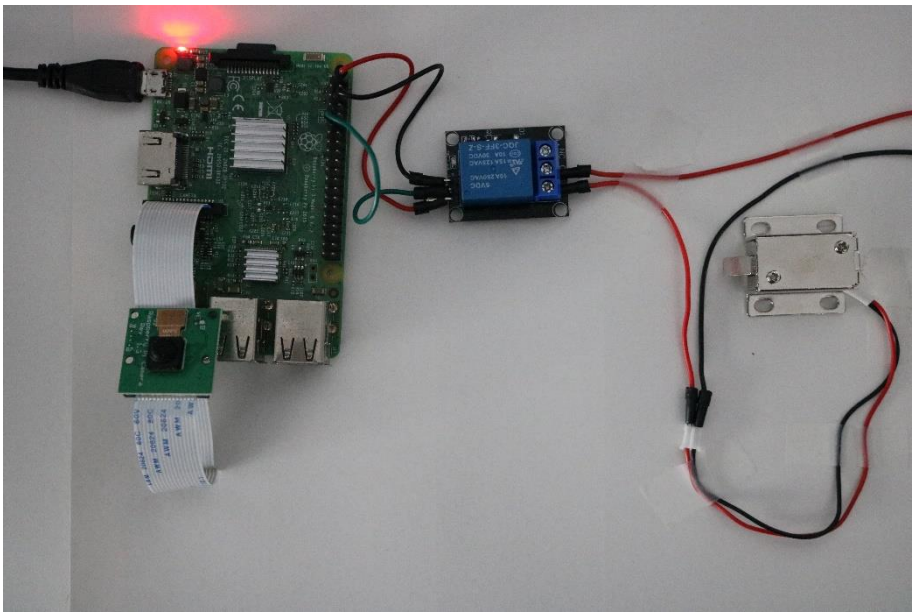

Figura 9. Protótipo do Módulo de Identificação

No aspecto visual, este módulo é consideravelmente simples, dispondo apenas de duas telas: uma para configuração inicial, apresentada na Figura 10, e outra que é exibida na maior parte do tempo, que identifica os rostos posicionados em frente à câmera e exibe o resultado da solicitação de acesso, conforme a Figura 11.

Figura 10. Tela de configuração inicial do Módulo de Identificação. Fonte: O Autor.

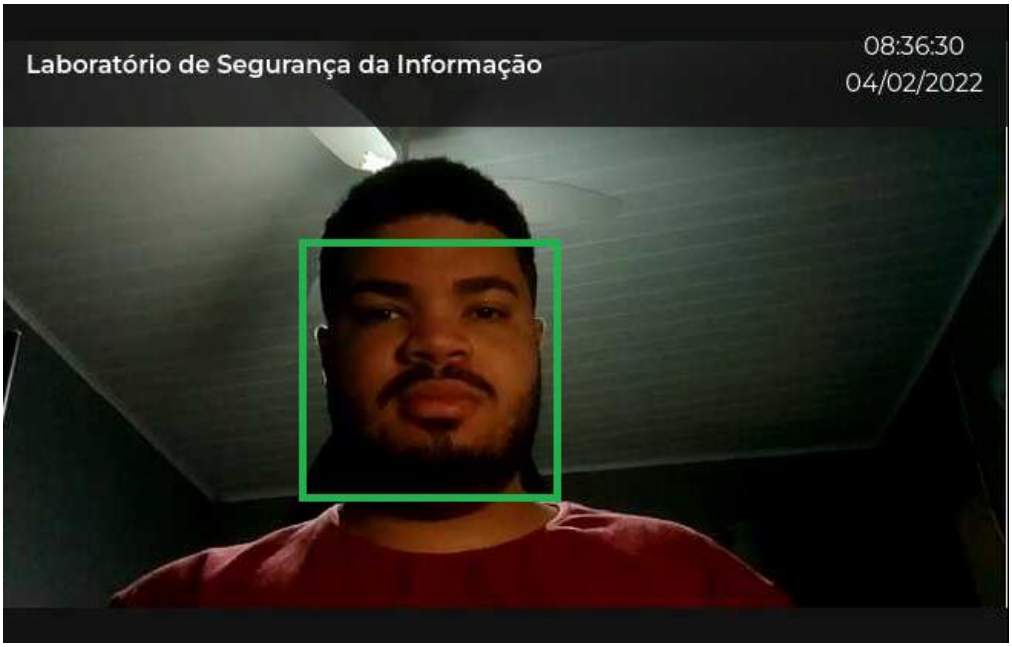

Figura 11. Tela principal do Módulo de Identificação. Fonte: O Autor.

Entretanto, apesar da simplicidade das telas, este módulo apresenta duas tarefas importantes e relativamente complexas. A primeira delas é a detecção de rostos: após configurado, o sistema estará constantemente buscando por faces no vídeo de entrada, conforme demonstrado no fragmento de código da Figura 12. A segunda tarefa é a descrição do rosto detectado: o sistema cria um vetor de descrição de 128 posições, o

envia para o módulo de autenticação e, caso seja retornada uma identificação, esta é enviada ao módulo de autorização para verificar se o indivíduo está autorizado a acessar o ambiente, para então liberar a fechadura.

```
// Analisa o frame de entrada em busca da principal face
const detections = await faceapi.detectSingleFace(InputVideo, new TinyFaceDetectorOptions())
    .withFaceLandmarks()
    .withFaceDescriptor()
    ;


if (detections) {
    // Caso encontre a face, ajusta o canvas para exibir a face
    const resizedDetections = faceapi.resizeResults(detections, displaySize)
    // Define os atributos do retângulo que será exibido
    const drawOptions = {
        boxColor: 'Green',
        lineWidth: 5
    }
    // Organiza os dados da localização da face
    const box = {
        x: resizedDetections.detection.box.x,
        y: resizedDetections.detection.box.y,
        width: resizedDetections.detection.box.width,
        height: resizedDetections.detection.box.height,
    }
    // Desenha um retângulo em volta da face
    const drawBox = new faceapi.draw.DrawBox(box, drawOptions)
    canvas.getContext('2d').clearRect(0, 0, canvas.width, canvas.height)
    drawBox.draw(canvas, resizedDetections)
}
```

Figura 12. Fragmento de código do processo de detecção facial

Vale ressaltar também que, para este módulo conseguir interagir com o módulo de autorização e, consequentemente, funcionar apropriadamente, existe uma cerimônia inicial, onde deve ser feita uma requisição a partir do cliente para a rota /configureclient do segundo módulo, enviando uma identificação única criada no momento de cadastro de ambiente, obtendo-se assim um identificador que deverá ser utilizado nas requisições futuras e demais dados necessários ao funcionamento do mesmo, como o *link* do Módulo de Autorização e o nome do ambiente.

6.4.2. Módulo de Autorização

No contexto de *front-end*, o módulo de autorização conta com uma aplicação desenvolvida em React, que pode ser acessada tanto de computadores quanto de celulares, pois conta com design responsivo. Visto que o intuito deste módulo é o gerenciamento de permissões, ele implementa diversas funcionalidades necessárias a esta gestão, que serão abordadas abaixo.

Toda interação neste módulo se inicia pela realização do login, apresentado na Figura 13 em sua versão para computador e para celular, com exceção da funcionalidade de recuperar a senha. Esta restrição garante a confidencialidade tanto dos ambientes gerenciados quanto dos dados dos usuários do sistema, assim como pratica o princípio do não-repúdio.

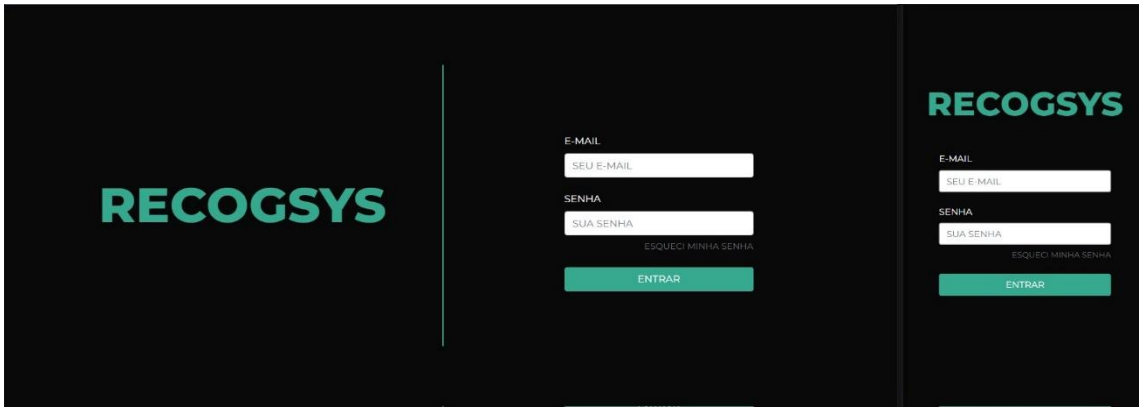

Figura 13. Tela de login do Módulo de Autorização na versão de computador à esquerda e na versão para celular à direita. Fonte: O Autor.

Após sua autenticação, os usuários têm acesso às páginas de acordo com seu papel, conforme especificado abaixo:

- Usuário: Podem verificar suas permissões e tentativas de acesso
- Recepcionista: Podem verificar permissões e tentativas de acesso de todos os usuários, cadastrar novos usuários e suas faces, conforme a Figura 14, cadastrar grupos de usuário e atribuir permissões descartáveis ou temporárias a qualquer usuário ou grupo de usuários
- Administrador: Podem verificar permissões e tentativas de acesso de todos os usuários, atribuir todos os tipos de permissões de acesso a qualquer usuário, e gerir os usuários e suas faces cadastradas, grupos de usuários, categorias de ambiente, ambientes e as configurações do módulo de autenticação

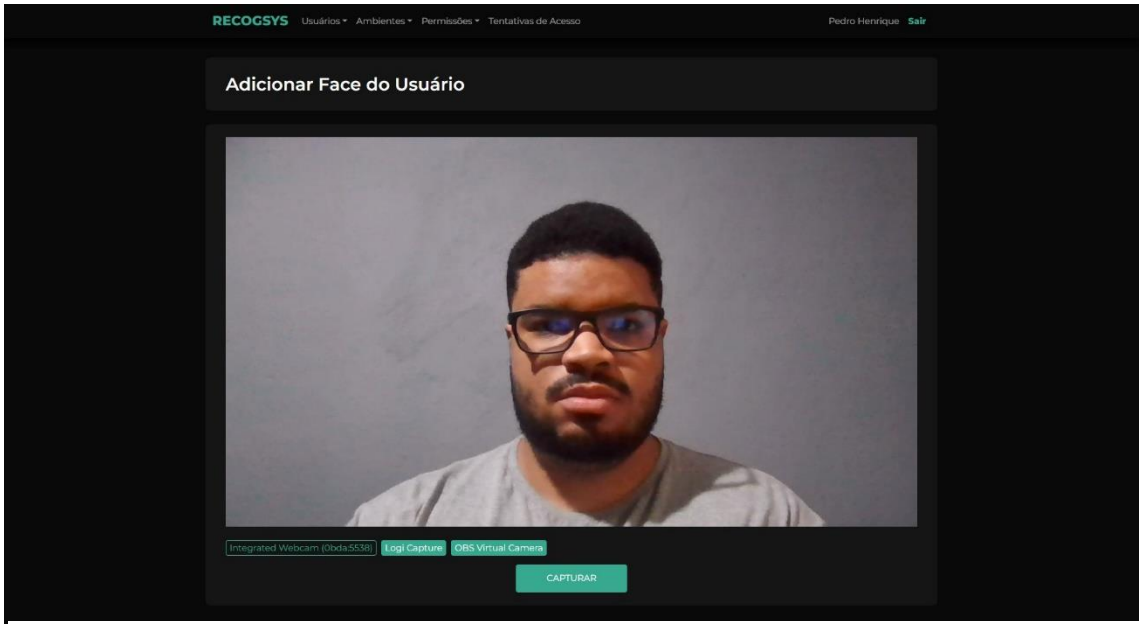

Figura 14. Tela de cadastro de faces do usuário no Módulo de Autorização. Fonte: O Autor.

Tal como seu logotipo, a interface do sistema foi desenvolvida de maneira minimalista, objetivando uma experiência de uso fluida e uma interface de usuário objetiva e de fácil entendimento e adaptação. Estes aspectos são bem representados pela Figura 15, que apresenta a tela de cadastro de usuários: nota-se que ela é objetiva e intuitiva. Outro exemplo relevante é a listagem de tentativas de acesso, exibida na Figura 16, que apresenta os acessos liberados em verde e os bloqueados em vermelho, permitindo a rápida associação do usuário.

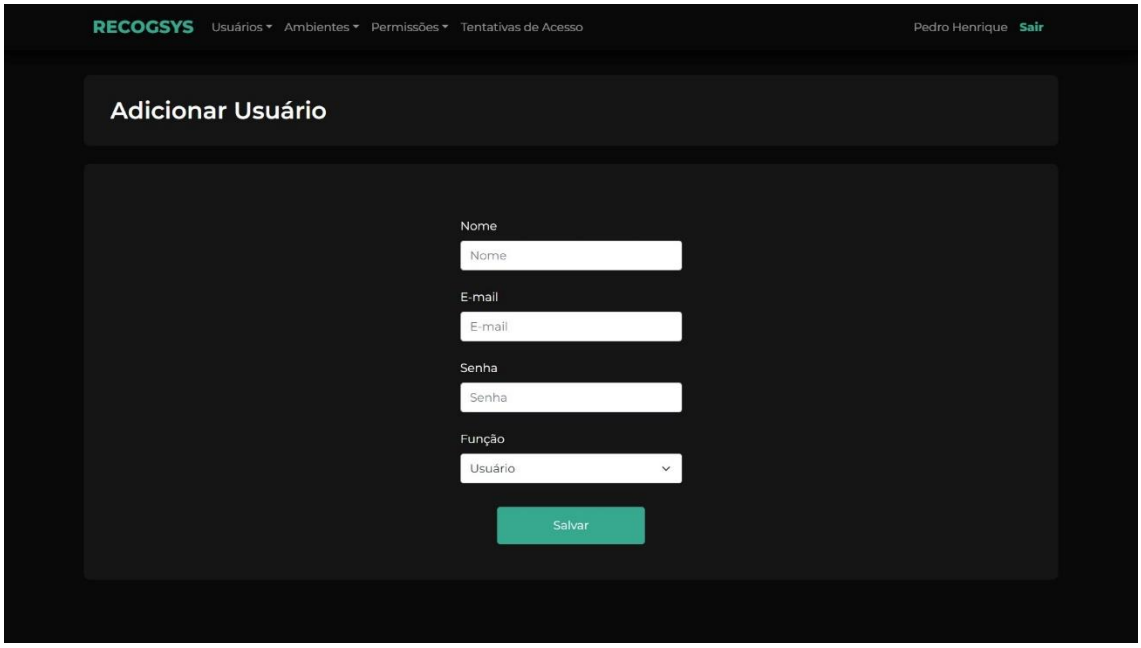

Figura 15. Tela de cadastro de usuário no Módulo de Autorização . Fonte: O Autor

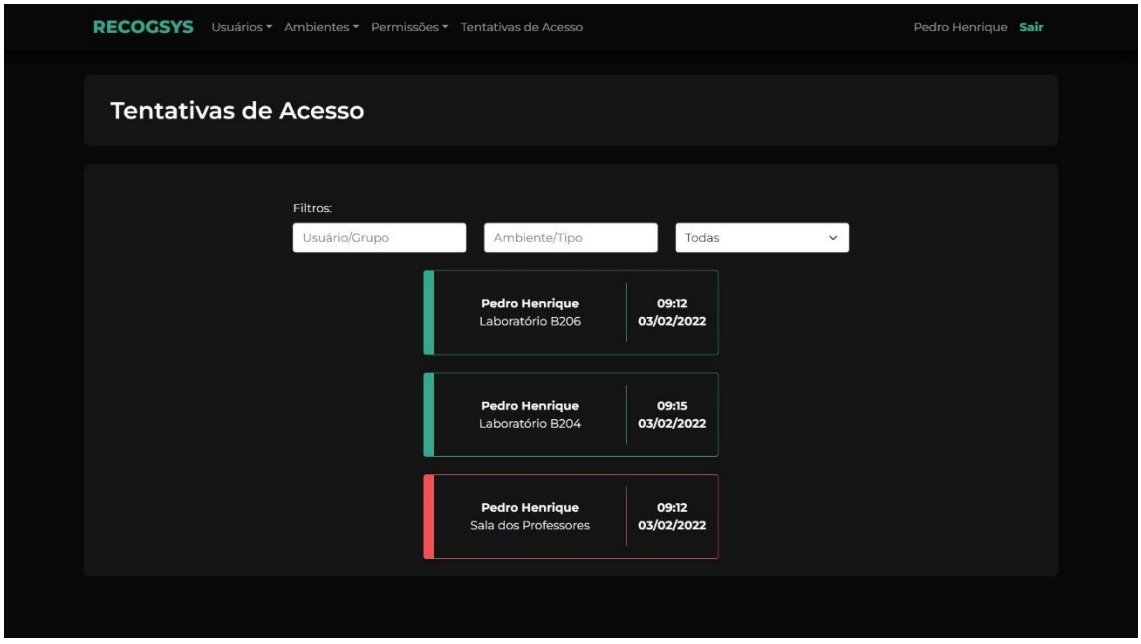

Figura 16. Tela de verificação de tentativas de acesso no Módulo de Autorização. Fonte: O Autor.

Ressalta-se também que, quando possível, as regras de negócio também são implementadas no *front-end*, completando três níveis garantidores de integridade para a maioria das regras de negócio.

Finalizando seu acesso, o usuário pode realizar logout a partir de qualquer página restrita do sistema, através do botão no canto superior direito na Figura 17, que apresenta a tela de verificação de permissões de acesso do usuário.

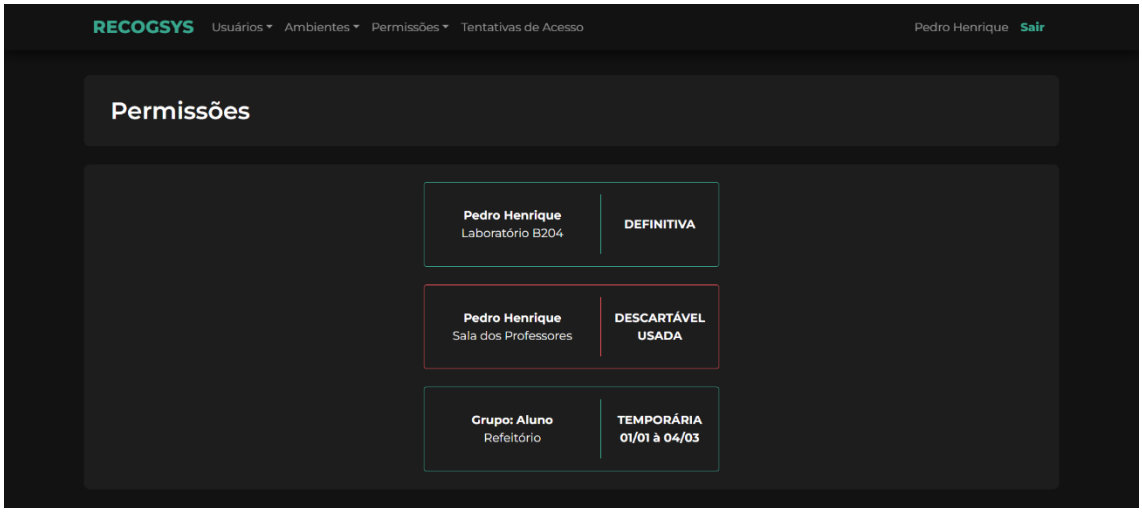

Figura 17. Tela de verificação de permissões no Módulo de Autorização. Fonte: O Autor.

Dada a explicação de todos os módulos do sistema, por fim, podemos verificar o produto de todas essas implementações: a autorização de acesso. Abaixo são apresentados os dois possíveis resultados de uma solicitação: autorização ou negação de acesso; com o módulo de identificação sendo executado em um Raspberry Pi, que por sua vez se comunica com os demais módulos sendo executados em uma máquina diferente. Na Figura 18, é apresentada a tela cujo acesso foi autorizado, e podemos conferir que a fechadura está destravada.

Figura 18. Sistema em execução no protótipo liberando o acesso

Enquanto isto, na Figura 19, podemos observar a situação na qual o acesso foi negado e a fechadura permanece travada.

Figura 19. Sistema em execução no protótipo negando o acesso

7. Resultados e Discussões

O sistema foi desenvolvido de acordo com as expectativas iniciais, permitindo o gerenciamento de usuários e seus respectivos grupos, faces e permissões, ambientes e seus respectivos tipos e mantendo registro de todas as tentativas de acesso, utilizando reconhecimento facial para a autenticação dos indivíduos.

No contexto de escalabilidade, o sistema apresenta-se totalmente escalável tanto verticalmente quanto horizontalmente. Esta capacidade é garantida, no módulo de autorização, pelo uso do Redis para armazenamento de sessões. Assim, este serviço pode ser replicado sem a perda de informações dos usuários logados, ressaltando-se que o uso do Redis não implica na perda de performance, uma vez que este apresenta desempenho extremamente veloz. Já no módulo de autorização, a escalabilidade está intrínseca ao seu desenvolvimento: por se tratar de um serviço *stateless*, não é necessária nenhuma abordagem para evitar a perda de dados na escala horizontal.

Além disso, é importante ressaltar que o módulo de autenticação realiza cache de todas as faces salvas em memória, sem prejuízos para a performance do serviço. Conforme observação da memória utilizada pela aplicação, cada nova face cadastrada aumenta, em média, 192Kb a quantidade de memória RAM utilizada. Ou seja, desprezando outras operações realizadas pelo servidor, uma alocação básica de 512Mb suportaria aproximadamente 2600 faces cadastradas.

Todavia, devido às restrições de circulação no presente contexto, não houve a possibilidade de testagem do sistema com grande quantidade de usuários cadastrados, movendo assim os esforços deste projeto para a garantia de qualidade nos demais aspectos do sistema, e registrando esta necessidade como trabalho futuro.

8. Trabalhos futuros

O primeiro trabalho futuro a ser realizado é a adaptação do protótipo para que possa ser instalado em um ambiente, para então serem realizados de testes com grande quantidade de usuários reais, conforme exposto no tópico Resultados e Discussões.

Além disso, através da observação dos resultados alcançados e de trabalhos similares citados na Revisão Bibliográfica, pode-se dizer que o sistema desenvolvido apresenta potencial de ser evoluído tanto em uma vertente vertical quanto em uma vertente horizontal.

Na vertente vertical, onde o objetivo é incrementar os serviços já oferecidos, pode-se planejar o desenvolvimento de novos módulos de autenticação, como *QR Code*, biometria digital e NFC, utilizando a arquitetura de micro *front-ends* para gerenciamento das funcionalidades específicas destes módulos de autenticação. Com esta evolução, pode-se se implementar a multi-autenticação, permitindo maior proteção para ambientes críticos.

Já em uma vertente horizontal, imagina-se a reestruturação do sistema como um SaaS, *Software as a Service – Software* como Serviço, onde diversas instituições diferentes possam acessar um único sistema, sem a necessidade de alocação de servidores *on-premises*[15] ou da alocação exacerbada de recursos em nuvem apenas para atender uma instituição. Pode-se incluir o uso de inteligência artificial também no módulo de autorizações, visando identificar tentativas suspeitas de acesso. Além disso, também é possível desenvolver uma versão simplificada do sistema, acoplando todas as funcionalidades em um único módulo, visando a utilização residencial.

9. Conclusões

Este projeto buscou desenvolver um sistema de controle de acesso baseado reconhecimento facial, utilizando arquitetura de microsserviços e alguns *design patterns*, como o *Repository* e o *Service*.

Pode-se dizer que o projeto já apresenta grau de maturidade para utilização supervisionada em ambientes reais, visando aprimoramento do sistema, assim como identificação de possíveis novas funcionalidades.

Também é importante ressaltar que, principalmente no contexto institucional, o controle de acesso é a primeira e principal barreira de segurança, ressaltando a necessidade de atenção a esta área, assim como a relevância deste tópico de pesquisa.

[15] Modelo de implementação de infraestrutura onde os equipamentos são instalados na própria sede da empresa e mantido pela sua respectiva equipe

Referências

Abozaid, A., Haggag, A., Kasban, H., and Eltokhy, M. (2019). Multimodal biometric scheme for human authentication technique based on voice and face recognition fusion. Multimedia Tools and Applications, 78(12):16345–16361.

Adjabi, I., Ouahabi, A., Benzaoui, A., and Taleb-Ahmed, A. (2020). Past, present, and future of face recognition: A review. Electronics, 9(8):1188.

Corrêa, E. M. (2020). Tutorial: linting em typescript com eslint. [Online; acessado em 11 de janeiro de 2022].

Fard, S. M. H. and Hashemi, S. (2020). Proposing a sparse representational based face verification system to run in a shortage of memory. Multimedia Tools and Applications, 79(3):2965–2985.

Ferreira, F. N. F. (2003). Segurança da informação. Editora Ciência Moderna.

Kang, K. (2019). Comparison of face recognition and detection models: Using different convolution neural networks. Optical Memory and Neural Networks, 28(2):101–108.

KRAKENFX (2021). Your fingerprint can be hacked for $5. here's how. by krakenfx. [Online; acessado em 22 de janeiro de 2022].

Lenon (2018). Node.js - o que é, como funciona e quais as vantagens. [Online; acessado em 07 de janeiro de 2022].

Lewis, J. and Fowler, M. (2014). Microservices: a definition of this new architectural term. [Online; acessado em 03 de janeiro de 2022].

Malik, A. K., Emmanuel, N., Zafar, S., Khattak, H. A., Raza, B., Khan, S., AlBayatti, A. H., Alassafi, M. O., Alfakeeh, A. S., and Alqarni, M. A. (2020). From conventional to state-of-the-art iot access control models. Electronics, 9(10):1693.

Mühler, V. (2019). face-api. js: Javascript api for face detection and face recognition in the browser and nodejs with tensorflow. js. [Online; acessado em 29 de janeiro de 2022].

Nascimento, F. (2021). Javascript ou typescript? [Online; acessado em 08 de janeiro de 2022].

Petrakis, E. G., Antonopoulos, F., Sotiriadis, S., and Bessis, N. (2020). ipacs: a physical access control system as a service and mobile application. Journal of Ambient Intelligence and Humanized Computing, 11(3):929–943.

PostgreSQL (2021). Postgresql: The world's most advanced open source relational database. [Online; acessado em 08 de janeiro de 2022].

Printi (2021). Psicologia das cores. [Online; acessado em 21 de janeiro de 2022].

React (2017). React: Uma biblioteca javascript para criar interfaces de usuário. [Online; acessado em 07 de janeiro de 2022].

Sabharwal, T., Gupta, R., Son, L. H., Kumar, R., and Jha, S. (2019). Recognition of surgically altered face images: an empirical analysis on recent advances. Artificial Intelligence Review, 52(2):1009–1040.

Schmitz, D. (2015). Tudo que você queria saber sobre git e github, mas tinha vergonha de perguntar. [Online; acessado em 11 de janeiro de 2022].

Si, W., Zhang, J., Li, Y.-D., Tan, W., Shao, Y.-F., and Yang, G.-L. (2020). Remote identity verification using gait analysis and face recognition. Wireless Communications and Mobile Computing, 2020.

Node.js (2019). Sobre | node.js. [Online; acessado em 07 de janeiro de 2022].

Tapia, F., Mora, M. Á., Fuertes, W., Aules, H., Flores, E., and Toulkeridis, T. (2020). From monolithic systems to microservices: A comparative study of performance. Applied Sciences, 10(17):5797.

Valle, L. F. F. G. and Dieminger, F. (2022). Começando com react. [Online; acessado em 07 de janeiro de 2022].

Zviran, M. and Erlich, Z. (2006). Identification and authentication: technology and implementation issues. Communications of the Association for Information Systems, 17(1):4.