

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS  
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO  
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO



**VERIFICAÇÃO AUTOMÁTICA GEORREFERENCIADA COM  
RECONHECIMENTO FACIAL**

ISABEL CRISTINA ARAÚJO RUBIM ALVES

GOIÂNIA  
2021

ISABEL CRISTINA ARAÚJO RUBIM ALVES

**VERIFICAÇÃO AUTOMÁTICA GEORREFERENCIADA COM  
RECONHECIMENTO FACIAL**

Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a):

Marcelo Antonio Adad de Araújo, MEE.

GOIÂNIA  
2021

ISABEL CRISTINA ARAÚJO RUBIM ALVES

**VERIFICAÇÃO AUTOMÁTICA GEORREFERENCIADA COM  
RECONHECIMENTO FACIAL**

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, para a obtenção do título de Bacharel em Engenharia de Computação, em 08 de junho de 2021.

---

Profa. Ma. Ludmilla Reis Pinheiro dos Santos  
Coordenador(a) de Trabalho de Conclusão de Curso

Banca examinadora:

---

Orientador: Prof. Marcelo Antonio Adad de Araújo,  
M.E.E.

---

Membro 1: Prof. Carlos Alexandre Ferreira de Lima,  
M.E.E.

---

Membro 2: Prof. Pedro Araújo Valle, M.E.E.

GOIÂNIA  
2021

À minha mãe Maria Helena.

## **AGRADECIMENTOS**

Primeiramente, agradeço a minha mãe Maria Helena, por sempre estar do meu lado e me oferecendo apoio durante o curso mesmo diante de dificuldades não deixou de acreditar em mim.

Agradeço imensamente ao professor Marcelo Antonio Adad de Araújo por ter me dado a oportunidade de ser sua orientanda e ter me ajudado a elaborar o presente trabalho. Agradeço pela sua disponibilidade de sempre para tirar dúvidas, pela paciência, pela dedicação e por compartilhar seu amplo conhecimento em computação contribuindo para minha formação acadêmica.

Aos colegas de curso que muitas vezes dedicaram um tempo para me ajudar durante a graduação, especialmente ao Maychon, que nunca se absteve em me ajudar, esteve sempre do meu lado durante todo esse longo percurso, com nossas semelhanças nos tornamos amigos.

## RESUMO

O presente trabalho apresenta um estudo e aplicação sobre o uso da verificação automática georreferenciada com reconhecimento facial, e uso de um sistema microcontrolado, aplicações desktop e mobile, uso do *Progressive Web App*, desenvolvidas em linguagens HTML, CSS, *JavaScript* e C para o sistema microcontrolado, com o propósito de conceder acesso automático a um local com base em uma foto cadastrada anteriormente em conjunto com o georreferenciamento. As informações necessárias para a verificação serão atingidas através do protótipo em *hardware* e *software*, utilizando os serviços cognitivo do Microsoft Azure de Detecção Facial para a verificação e validação do reconhecimento facial e APIs de georreferenciamento em conjunto com o ESP32. O armazenamento dos dados é realizado no banco de dados em nuvem utilizando a plataforma Google *Firebase*, para fazer a comunicação das informações entre o sistema microcontrolado, aplicação desktop e mobile. Através da aplicação o usuário poderá usar das decisões para o destravamento de um local de forma automática com sua identificação.

***Palavras-Chave:*** *Reconhecimento Facial, Georreferenciamento, Segurança, Automação, ESP32.*

## ABSTRACT

This work presents a study and application on the use of automatic georeferenced verification with facial recognition, and the use of a microcontrolled system, desktop and mobile applications, use of the Progressive Web App, developed in HTML, CSS, JavaScript and C languages for the system micro-controlled, with the purpose of granting automatic access to a location based on a previously registered photo in conjunction with geo-referencing. The information needed for verification will be achieved through prototype hardware and software using Microsoft Azure Cognitive Facial Detection services for verification and validation of facial recognition and georeferencing APIs in conjunction with ESP32. Data storage is performed in the cloud database using the Google Firebase platform, to communicate information between the microcontrolled system, desktop and mobile applications. Through the application, the user will be able to use the decisions to unlock a location automatically with its identification.

*Palavras-Chave: Facial Recognition, Geofencing, Security, Automation, ESP32.*

## LISTA DE FIGURAS

|  |    |
|--|----|
| Figura 1 - Processo da tarefa de verificar.....  | 23 |
| Figura 2 - Processo da tarefa de identificar.....  | 23 |
| Figura 3 - Atividades do processo para reconhecimento facial.....                            | 25 |
| Figura 4 - Árvore JSON exemplo.....  | 28 |
| Figura 5 - Diagrama do funcionamento de uma API.....   | 29 |
| Figura 6 - Adição da área demarcada.....   | 31 |
| Figura 7 - Figura ilustrativa do projeto em geral.....                                       | 32 |
| Figura 8 - Interface da IDE do Arduino.....  | 34 |
| Figura 9 - Gerenciador de placas, necessário para a utilização das bibliotecas do ESP32..... | 35 |
| Figura 10 - Tela de gerenciamento de bibliotecas.....  | 35 |
| Figura 11 - Tela para criar um projeto no <i>Firebase</i> .....                              | 36 |
| Figura 12 - Tela para definir o nome do projeto <i>Firebase</i> .....                        | 37 |
| Figura 13 - Tela do <i>Cloud Firestore</i> do <i>Firebase</i> .....                          | 37 |
| Figura 14 - Tela <i>Realtime Database</i> do <i>Firebase</i> .....                           | 38 |
| Figura 15 - Tela de cadastro do usuário.....   | 39 |
| Figura 16 - Dados do usuário cadastrado.....   | 39 |
| Figura 17 - Tela de login.....   | 40 |
| Figura 18 - <i>Class Auth.js</i> .....   | 41 |
| Figura 19 - Banco de dados <i>Realtime</i> com os dados salvos da área demarcada.....        | 42 |
| Figura 20 - <i>Class RealtimeDb.js</i> .....   | 43 |
| Figura 21 - Tela inicial do portal do Azure para iniciar a avaliação gratuita.....           | 44 |
| Figura 22 - <i>Marketplace</i> dos serviços cognitivos encontrados no portal do Azure.....   | 44 |
| Figura 23 - Tela para criar o serviço cognitivo Detecção Facial no portal do Azure.....      | 45 |
| Figura 24 - Tela de acesso as chaves e ponto de extremidade no portal do Azure.....          | 45 |
| Figura 25 - Detecção Facial com características relacionadas do rosto.....                   | 46 |
| Figura 26 - Local trancado para o usuário.....   | 47 |
| Figura 27 - Câmera frontal ligada.....   | 48 |
| Figura 28 - Foto tirada para a verificação da face.....                                      | 49 |
| Figura 29 - <i>Class Face.js</i> .....   | 50 |



|   |    |
|---|----|
| Figura 30 - Representação da funcionalidade do <i>webpack</i> .....                                 | 51 |
| Figura 31 - Ponto de entradas para a configuração do <i>webpack</i> .....                           | 52 |
| Figura 32 - Saída para a configuração do <i>webpack</i> .....                                       | 52 |
| Figura 33 - Implementação dos <i>loaders</i> no arquivo de configuração do <i>webpack</i> .....     | 53 |
| Figura 34 - Plugins implementados no arquivo de configuração do <i>webpack</i> .....                | 54 |
| Figura 35 - Implementação do compilador babel no arquivo de configuração <i>.babelrc</i> .....      | 55 |
| Figura 36 - Arquivo <i>package.json</i> .....   | 56 |
| Figura 37 - Criação da conta dos Mapas do Azure.....  | 57 |
| Figura 38 - Adição da área demarcada desejada do usuário.....                                       | 58 |
| Figura 39 - Pop-up indicando que o usuário está perto da cerca virtual por metros de distância..... | 59 |
| Figura 40 - Pop-up indicando que o usuário está há tantos quilômetros da cerca virtual.....         | 59 |
| Figura 41 - Plugin <i>WorkboxWebpackPlugin</i> .....  | 60 |
| Figura 42 - Plugin <i>WebpackPwaManifest</i> .....  | 61 |
| Figura 43 - Tela para adicionar <i>Geo Face</i> a tela inicial do smartphone.....                   | 62 |
| Figura 44 - Tela para a confirmação da instalação do aplicativo.....                                | 62 |
| Figura 45 - Tela com o aplicativo adicionado a tela inicial do smartphone.....                      | 63 |
| Figura 46 - Tela inicial ao clicar no aplicativo.....   | 64 |
| Figura 47 - Aplicativo aberto no smartphone.....  | 65 |

## LISTA DE ABREVIATURAS

|          |                               |
|----------|-------------------------------|
| Me(a).   | Mestre(a)                     |
| Prof(a). | Professor(a)                  |
| M.E.E.   | Mestre em Engenharia Elétrica |
| MSc.     | Mestre em Ciências            |

## LISTA DE SIGLAS

|       |  |
|-------|--|
| Wi-Fi | <i>Wireless Fidelity</i>                   |
| IoT   | <i>Internet of Things</i>                  |
| RFID  | <i>Radio Frequency Identification</i>      |
| NFC   | <i>Near Field Communication</i>            |
| IP    | <i>Internet Protocol</i>                   |
| FPGAs | <i>Field Programmable Gate Array</i>       |
| RDF   | <i>Resource Description Framework</i>      |
| OWL   | <i>Web Ontology Language</i>               |
| EXI   | <i>Efficient XML Interchange</i>           |
| JS    | <i>JavaScript</i>                          |
| API   | <i>Application Programming Interface</i>   |
| GPS   | <i>Global Positioning System</i>           |
| WPS   | <i>Wi-Fi Protected Setup (WPS)</i>         |
| PWA   | <i>Progressive Web App</i>                 |
| HTML  | <i>Hyper Text Markup Language</i>          |
| MCU   | <i>Microcontroller Unit</i>                |
| μC    | Unidade de Controle                        |
| SOC   | <i>System On a Chip</i>                    |
| CPU   | <i>Central Processing Unit</i>             |
| POI   | Ponto de Interesse                         |
| NoSQL | <i>Not only Structured Query Language</i>  |
| JSON  | <i>JavaScript Object Notation</i>          |
| API   | <i>Application Programming Interface</i>   |
| CEP   | Código de Endereçamento Postal             |
| HTTPS | <i>Hyper Text Transfer Protocol Secure</i> |
| HTTP  | <i>Hyper Text Transfer Protocol</i>        |
| REST  | <i>Representational State Transfer</i>     |
| SDK   | <i>Software Development Kit</i>            |
| WAN   | <i>World Area Network</i>                  |

|      |                                 |
|------|---------------------------------|
| IA   | Inteligência Artificial         |
| ID   | Identificação                   |
| URL  | <i>Uniform Resource Locator</i> |
| BLOB | <i>Binary Large Object</i>      |
| CLI  | <i>Command-Line Interface</i>   |

## LISTA DE EQUAÇÕES

|                |    |
|----------------|----|
| Equação 1..... | 60 |
|----------------|----|

## SUMÁRIO

|   |           |
|---|-----------|
| <b>1 INTRODUÇÃO .....</b>                                   | <b>15</b> |
| 1.1 Objetivos.....  | 16        |
| 1.1.1 Objetivo Geral .....                                  | 16        |
| 1.1.2 Objetivos Específicos .....                           | 16        |
| 1.2 Justificativa.....                                      | 17        |
| 1.3 Métodos .....   | 17        |
| 1.4 Resultados esperados .....                              | 17        |
| <b>2 REFERENCIAL TEÓRICO.....</b>                           | <b>18</b> |
| 2.1 <i>Internet of Things</i> .....                         | 18        |
| 2.2 Microcontroladores.....                                 | 19        |
| 2.3 <i>JavaScript</i> .....                                 | 19        |
| 2.3.1 <i>Progressive Web App</i> .....                      | 20        |
| 2.4 Biometria .....   | 21        |
| 2.5 Reconhecimento facial .....                             | 22        |
| 2.6 <i>Geofencing</i> .....                                 | 25        |
| 2.7 <i>Firebase</i> .....                                   | 27        |
| 2.8 Interface de Programação de Aplicativos.....            | 28        |
| 2.9 Azure .....   | 29        |
| 2.10 Trabalho relacionado .....                             | 30        |
| 2.10.1 <i>Verificação Automática Georreferenciada</i> ..... | 30        |
| <b>3 PROPOSTA DE SOLUÇÃO .....</b>                          | <b>32</b> |
| 3.1 Componentes de hardware.....                            | 33        |
| 3.1.1 Microcontrolador ESP32 .....                          | 33        |
| 3.2 Componentes de software.....                            | 33        |
| 3.2.1 Visual Studio Code.....                               | 33        |

|   |           |
|---|-----------|
| 3.2.2 Arduino IDE .....                         | 33        |
| 3.2.3 Google <i>Firebase</i> .....              | 36        |
| 3.2.4 <i>Cloud Firestore</i> .....              | 38        |
| 3.2.5 <i>Realtime Database</i> .....            | 41        |
| 3.2.6 Portal Azure.....                         | 43        |
| 3.2.7 Detecção Facial do Azure.....             | 46        |
| 3.2.8 <i>Webpack</i> .....                      | 51        |
| 3.2.9 Babel.....                                | 54        |
| 3.2.10 npm .....                                | 55        |
| 3.2.11 Azure Mapas.....                         | 56        |
| 3.2.12 <i>Progressive Web App</i> .....         | 60        |
| <b>4 CONSIDERAÇÕES FINAIS E CONCLUSÃO .....</b> | <b>66</b> |
| <b>5 REFERÊNCIAS .....</b>                      | <b>68</b> |
| <b>APÊNDICE A .....</b>                         | <b>74</b> |

## 1 INTRODUÇÃO

O presente trabalho de conclusão de curso pretende abordar de forma construtiva os assuntos a seguir em seu capítulo inicial.

Com a evolução da ciência as aplicações na tecnologia, indústria e outros, é fundamental o uso de microcontroladores, por serem computadores de um único chip lançados para possibilitar o controle de dispositivos a receber informações de forma independente e automática. Assim, para desenvolver um sistema microcontrolado é necessário executar um conjunto de instruções que fiquem presentes em uma memória não volátil para que os dados sejam persistentes. Com isso, é possível o desenvolvimento dos sistemas microcontrolados de forma automática (PERIM; NASCIMENTO, 2017).

O termo internet das coisas que é uma conexão com vários dispositivos eletrônicos de modo que podem ser controlados remotamente por meio de um tipo de rede, através de algum outro dispositivo para ser controlado, salvo maior juízo, é a aplicação ideal para os microcontroladores (SANTOS *et al.*, 2016), apresentando-se em várias aplicações, atualmente visto na automação residencial, industrial e comercial.

Entretanto, o dispositivo a ser controlado pode ser uma aplicação para sistemas microcontrolados, desktop ou mobile. Na atualidade, o *Progressive Web App* (PWA) permite desenvolver uma aplicação tanto para mobile quanto para desktop, com as linguagens de programação HTML, CSS, *JavaScript*, sendo possível ter as funcionalidades de um aplicativo clássico, como notificações, câmera, geolocalização, entre outros (TRINDADE; AFFINI, 2019). Um sistema microcontrolado, pode ser desenvolvido em uma linguagem C, para que possa acionar objetos físicos e enviar informações.

A localização do usuário pode ser obtida através da geolocalização do próprio aparelho celular (ALVES, 2018). Desse modo, o georreferenciamento pode criar uma cerca geográfica virtual sendo composta pelas coordenadas coletadas com a geolocalização e possibilitando definir uma área de preferência do usuário e até notificar ao usuário, caso ele passe pela área demarcada anteriormente (WU, 2019). O georreferenciamento pode ser utilizado em várias aplicações que necessitam de rastreamento, além de proporcionar mais segurança (ZIN, *et. al.*, 2016).

O reconhecimento facial é uma forma de segurança mais criteriosa, nela é possível identificar uma pessoa de forma precisa, verificando e validando características únicas deste indivíduo, sendo muito utilizado, para conceder acesso em determinados ambientes, passando



por testes para verificar se o candidato analisado é realmente quem diz ser. A verificação é a comparação de uma imagem ou um vídeo com uma imagem cadastrada anteriormente armazenada em uma base de dados, assim verificando se a identidade é verdadeira ou falsa (FARGETUN, 2005).

No capítulo dois, é apresentado o histórico do estudo do problema abordado no presente trabalho.

No capítulo três, apresenta-se uma proposta de solução para o problema descrito anteriormente, ou seja, utilizando microcontroladores, georreferenciamento e reconhecimento facial para uma solução segura.

O trabalho procurará responder à pergunta se é realmente possível a utilização do georreferenciamento, microcontroladores e reconhecimento facial, para se obter um grau de segurança satisfatório para o usuário.

## **1.1 Objetivos**

Neste tópico será apresentado o objetivo geral e os objetivos específicos do presente trabalho.

### **1.1.1 Objetivo Geral**

Realizar o reconhecimento facial de forma automática de uma foto já cadastrada anteriormente, em conjunto com georreferenciamento para entrada automática do local.

### **1.1.2 Objetivos Específicos**

Estão divididos em sete etapas para que se possa construir a aplicação de forma efetiva.

- Construir uma aplicação híbrida (Desktop, Mobile) sendo uma utilizada uma estrutura para desenvolver componentes em qualquer plataforma;
- Utilizar serviço em nuvem para autenticar o usuário e salvar dados, com segurança local;
- Utilizar o *Wireless Fidelity* (Wi-Fi) para conectar o microcontrolador;
- Detectar e verificar a foto facial do usuário em análise;
- Determinar uma cerca geográfica computacional e verificar sua validade;
- Validar o usuário através da foto facial no mobile;

- Destruar a tranca de uma porta através do reconhecimento facial com *WebView* e utilização de microcontrolador.

## 1.2 Justificativa

O presente trabalho por questões de segurança do usuário justifica o uso do georreferenciamento em conjunto com o reconhecimento facial apresentando mais segurança, através de uma aplicação desktop, mobile e microcontrolada para destravar um local por meio da identificação do indivíduo.

É sabido que a *Internet of Things* (IoT) permite a interação de *hardware* e *software* através da internet, sendo possível trocar dados entre eles. Com a tecnologia de georreferenciamento torna-se possível coletar a localização e demarcar uma área de preferência do usuário, além disso, o uso do serviço cognitivo de Detecção Facial do Microsoft Azure pode tornar realidade o reconhecimento facial de um indivíduo, portanto pode ser funcional para o usuário. A aplicação é desenvolvida para o usuário demarcar uma área em que ele queira entrar automaticamente através do reconhecimento facial.

## 1.3 Métodos

Primeiramente, é realizada uma análise do estado da arte dos assuntos relacionados com a utilização de periódicos, livros, artigos, revistas e outras fontes confiáveis. Em seguida, é desenvolvido um software para web e mobile que realize o georreferenciamento e o posterior reconhecimento facial, juntamente com o software que é desenvolvido para o microcontrolador ESP32. Logo depois, é realizada a montagem dos componentes de hardware para que se realize a integração com os softwares desenvolvidos, e por fim, realizar o destravamento automático de uma fechadura em um local da forma mais segura possível.

## 1.4 Resultados esperados

Deseja-se que os resultados esperados no presente trabalho sejam capazes de contribuir com os estudos de IoT, para mais segurança através da automação residencial, comercial ou industrial através do reconhecimento facial e georreferenciamento para conceder acesso a um local.

## 2 REFERENCIAL TEÓRICO

Esse capítulo mostra definições e os conceitos dos termos que já vêm sendo utilizados para desenvolvimento deste projeto, logo em seguida são apresentados os trabalhos relacionados.

### 2.1 *Internet of Things*

Internet das Coisas (*do inglês Internet of Things = IoT*) é a conexão entre objetos computacionais, do dia a dia, através da internet ou alguma rede, para controlá-los remotamente de forma que sejam acessados por um dispositivo a ser controlado (SANTOS *et al.*, 2016).

A internet das coisas pode ter várias aplicações, como cidades inteligentes, aplicações em saúde, casas inteligentes, seguranças, dentre outras. A IoT é um conjunto de tecnologias que são complementares para que a comunicação e integração dos objetos físicos e do mundo virtual possa se tornar viável. Apresentam-se, aqui, alguns blocos básicos para construção da IoT, são eles:

**Identificação:** para tecnologias como *Radio Frequency Identification (RFID)*, *Near Field Communication (NFC)* e endereçamento *Internet Protocol (IP)*.

**Sensores e atuadores:** são os que coletam informações dos objetos físicos para informações destinadas ao contexto, computacional por exemplo.

**Comunicação:** utilizando técnicas para a conexão dos objetos com algumas tecnologias como Wi-Fi, Bluetooth, IEEE 802.15.4 e RFID.

**Computação:** é responsável por compilar algoritmos nos objetos inteligentes incluindo microcontroladores, processadores e FPGAs (*Field Programmable Gate Array*).

**Serviços:** são utilizados para identificações de interesse do usuário para que sejam disponíveis de qualquer lugar onde seja necessário.

**Semântica:** extrai os dados existentes dos objetos, tendo como objetivo fornecer um serviço específico. São utilizadas diversas técnicas como *Resource Description Framework (RDF)*, *Web Ontology Language (OWL)* e *Efficient XML Interchange (EXI)* (SANTOS *et al.*, 2016).

## 2.2 Microcontroladores

Os microcontroladores (MCU ou  $\mu\text{C}$ ) podem estar nos mais diversos dispositivos eletrônicos que existem atualmente, como automóveis, aviões, TVs e brinquedos, entre outros. Ele pode ser programável, com isso há um sistema computacional completo dentro de um único chip que fez gerar o termo *System On a Chip* (SOC) (REBOUÇAS FILHO, 2014). O SOC representa que é possível ter um sistema completo em único chip, com isso acelerando as soluções de controle eletroeletrônico na indústria (PERIM; NASCIMENTO, 2017).

Sistemas de único chip são fruto da evolução da tecnologia, da ciência e da indústria para que se possa ter circuitos dedicados apenas em um chip. Assim surgiu o microcontrolador, capaz de controlar dispositivos que são capazes de enviar e receber informações de forma autônoma e automática, desse modo, tornando o desenvolvimento de sistemas automáticos possível (PERIM; NASCIMENTO, 2017). Estes dispositivos incluem no interior de um único chip *Central Processing Unit* (CPU), memória de dados e de programa, entrada/saída, relógio interno, temporizadores, entre outros. São dispositivos relativamente baratos e podem ter alta eficiência (REBOUÇAS FILHO, 2014).

Para obter um sistema microcontrolado programável capaz de executar um conjunto de instruções para controle de algum dispositivo, a CPU deve se ligar a outras partes através de barramentos para operar em conjunto. E o código deve estar presente em uma memória não volátil, geralmente do tipo *Flash*, o que significa que os dados são persistentes no microcontrolador e com isso controla as ações. Assim, o programa a ser construído realiza as trocas de dados de forma correta e comanda memórias e dispositivos internos (PERIM; NASCIMENTO, 2017).

## 2.3 JavaScript

*JavaScript* (JS) está presente em quase todos os navegadores da web sendo uma linguagem de script, e interpretada na qual a análise do código fonte ocorre ao mesmo tempo que a execução do código. Ela é baseada em uma linguagem de objetos e protótipos que vem do paradigma de orientação a objeto, mas com uma diferença, os objetos se clonam entre os preexistentes por isso o emprego do termo protótipo (SILVA; SOBRAL, 2017).

Com isso, ela permite desenvolver do lado do cliente, solicitando dados dos servidores e assim tornando as páginas web dinâmicas e retornando uma resposta às ações do usuário de forma interativa (SILVA; SOBRAL, 2017). Sendo uma linguagem interpretada, na prática os

navegadores compilam a linguagem antes da execução do código, em medidas de desempenho não é considerada uma simples adição (COPES, 2018).

Portanto, *JavaScript* é uma linguagem de programação possibilitando implementar páginas web complexas, com intervalo de tempo mínimo na atualização. Ela está composta no terceiro bloco que compõe programas da web, duas das outras são *Hyper Text Markup Language* (HTML) e *Cascading Style Sheets* (CSS), sendo o HTML utilizado para estruturar as páginas web na forma de como inserir parágrafos, tabelas, cabeçalhos, conteúdo, imagens e vídeos. O CSS define a estilização do conteúdo HTML sendo utilizado para estruturar, cores, posicionamento do conteúdo e fontes, assim a linguagem *JavaScript* permite ao conteúdo ser atualizado dinamicamente (COLABORADORES DA MOZILLA, 2020).

### 2.3.1 *Progressive Web App*

*Progressive Web App* (PWA) foi abordado pelo Google para se usar os blocos que compõem a web (HTML, CSS, *JavaScript*), apresentando um comportamento semelhante a um aplicativo mobile, podendo ainda ser instalado através da *Uniform Resource Locator* (URL) sem a necessidade de uma loja de aplicativos. Com isso, quando instalado, as funcionalidades que constituem um aplicativo clássico são ativadas como notificações, desempenho mais rápido, gerência de memória, geolocalização, câmera, uso offline, entre outros (TRINDADE; AFFINI, 2019).

De acordo com o Google, as seguintes características que um PWA possui são:

- Um design responsivo para ser compatível com qualquer formato como desktop, celular e tablet;
- Conectividade para trabalhar offline, ou com baixa qualidade de rede. Sendo também similar a aplicativos;
- Atualizado e seguro via *Hyper Text Transfer Protocol Secure* (HTTPS);
- Identificado assim como aplicativos em mecanismos de buscas;
- Compatível com recursos mobile;
- Permite ser instalado sem a necessidade para acessar uma loja de aplicativos tal que pode ser acessado na tela inicial do dispositivo;
- Compartilhamento facilitado através da URL (TRINDADE e AFFINI, 2019).

Para a integração do PWA supre três requisitos necessários que são Conexão HTTPS, *Service Worker* e *Web App Manifest*. O Manifesto do Aplicativo Web é um *JavaScript Object*

*Notation* (JSON), que fornece as informações como ícone, nome, autor e descrição sobre o aplicativo fornecendo detalhes quando fica presente na tela inicial permitindo funcionar offline e receber notificações tipo push. O *Service Worker* é um script que trabalha em segundo plano o qual não necessita da ação do usuário, funcionando como um servidor proxy que comunica entre o navegador web e aplicação web e a rede quando estiver disponível para que possa fornecer experiências offline, permitindo adiar a conectividade até que o usuário tenha uma rede estável (TRINDADE; AFFINI, 2019).

Dessa forma, um PWA se comporta como um aplicativo nativo de um *smartphone*, tablet, laptop ou desktop. Através do acesso a um navegador web e com a interação do usuário, torna-se um aplicativo a partir da adição do usuário para a tela inicial do dispositivo em que ele se encontra, com isso passa a ter as funcionalidades que eram exclusivas de aplicativos nativos, como por exemplo: uso de câmera, offline, notificações, geolocalização, entre outros. No desenvolvimento da aplicação, usam-se os blocos que compõem a web (HTML, CSS, JS), permitindo a manutenção com baixo custo e a portabilidade entre navegadores e plataformas, garantidos pelo design responsivo (TRINDADE; AFFINI, 2019).

## 2.4 Biometria

O termo biometria é derivado do grego *bios* (vida) e *metron* (medida). A biometria é uma forma de avaliação de características físicas únicas ou comportamentais exclusivas de cada pessoa. O conjunto entre o uso da tecnologia e da biometria é utilizado principalmente para autenticar ou identificar a identidade dos indivíduos. A autenticação funciona para permitir ou não permitir a identificação do indivíduo, enquanto a identificação estabelece a identidade a ser conhecida de um indivíduo (MAGALHÃES; SANTOS, 2013).

Existem duas categorias de medidas levadas em consideração na biometria, as medidas fisiológicas que podem ser morfológicas ou biológicas. Os identificadores morfológicos são impressões digitais, formatos da mão, dedo, padrão de veia, rosto e olhos, que incluem íris e retina.

Para as identificações biológicas, tem-se o DNA, sangue, saliva, urina, em suma os diversos fluidos corporais que são altamente utilizados na medicina e polícias forenses (THALES GROUP, 2020).

As medidas comportamentais são reconhecimento de voz, pressionamento de tecla, a forma que um indivíduo usa tal objeto, gestos, som de passos, assinatura com características que levam em consideração a velocidade do movimento da caneta, pressão exercida,

acelerações e inclinação. No entanto, os vários conjuntos de medição variam em grau de confiabilidade. As medições fisiológicas oferecem mais confiança por se manter constante ao longo da vida de um indivíduo por não estarem expostas aos efeitos do estresse, como nas medidas comportamentais (THALES GROUP, 2020).

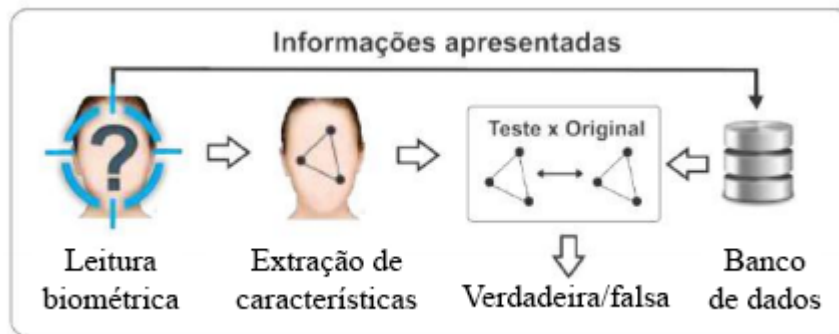
## **2.5 Reconhecimento facial**

O reconhecimento facial tem como função identificar padrões faciais com o formato da face, boca, nariz, distância dos olhos, entre outros (PRADO, 2018). Assim é feita sem a necessidade da interação direta com o indivíduo que está prestes a ser avaliado, necessitando apenas de uma câmera para captura da imagem e um sistema que dê seguimento ao reconhecimento e ocorra a identificação (FAGERTUN, 2015).

As etapas do reconhecimento facial visam analisar as características únicas da face capturada de cada indivíduo que possam estar presentes em uma base de dados. O processo para o reconhecimento de faces pode variar para cada sistema dependendo do seu objetivo, sendo que cada processo pode ser feito de forma individualizada, assim o processo pode ser realizado em três tarefas específicas (FAGERTUN, 2005), são elas: verificar, identificar e reconhecer (COSTA, 2019).

A forma para se verificar se um indivíduo é ele mesmo, é o processo de reconhecer e identificar como uma tarefa, para por exemplo se obter acesso a um determinado local, o indivíduo deve passar por um teste para verificar se ele é realmente quem diz ser. O teste de verificação começa separando os indivíduos em dois grupos, sendo um deles os usuários que querem obter acesso com sua própria identidade e o outro que querem obter acesso através de, por exemplo, uma identidade falsa. A verificação compara a imagem facial com as imagens que existem em um banco de dados do sistema, para que o objetivo seja encontrar apenas uma imagem correspondente. A Figura 1 demonstra o processo para a verificação facial com o objetivo de verificar a identidade verdadeira ou falsa de um usuário (FARGETUN, 2005).

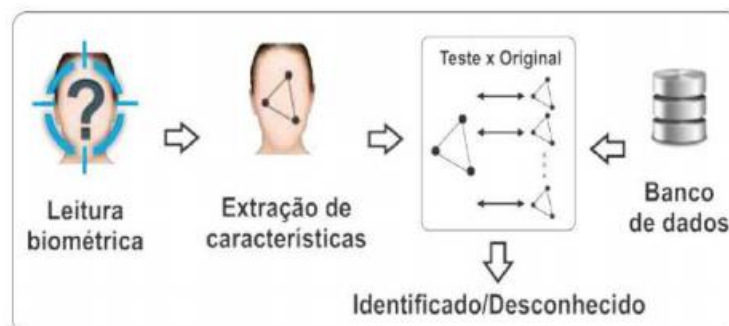
Figura 1 - Processo da tarefa de verificar.



Fonte: adaptada de Silva, 2015.

A tarefa de identificar consiste em um dos requisitos mais complexos nos sistemas de reconhecimento facial, onde a vigilância diária verifica vários indivíduos com suas características faciais já conhecidas e armazenadas no banco de dados (PRADO, 2018). Essa tarefa não apresenta necessidade do contato direto com o indivíduo, porque captura de forma automática a face dos indivíduos que está presente na câmera de monitoramento do local. Há um esforço computacional maior nessa atividade porque é sem o conhecimento do indivíduo que a face dele está sendo capturada, portanto a comparação da imagem é feita através do banco de imagens já existentes, verificando se a imagem capturada é conhecida ou desconhecida. Para que essa tarefa seja bem-sucedida o banco de imagens deve conter um número alto de imagens faciais e não só de características faciais específicas dos indivíduos, para que a identificação seja possível para qualquer indivíduo (FAGERTUN, 2005). A Figura 2 mostra o processo da tarefa de identificar.

Figura 2 - Processo da tarefa de identificar.



Fonte: Silva, 2015.

Na tarefa de identificar, o teste facial parte da ideia inicial de que todas as faces dos indivíduos já são conhecidas. Com isso, é comum que o banco de dados contenha imagens que



vêm de diversas bases espalhadas em diferentes ambientes em que o sistema de reconhecimento facial se encontra. Portanto, uma maior quantidade de imagens diversas deve compor o banco de dados, sendo maior a probabilidade de identificar um indivíduo como reconhecido em ambientes abertos com o sistema em operação (COSTA, 2019).

Por fim, a tarefa de reconhecimento é uma réplica da tarefa de identificação, sendo a diferença no momento presente do estudo a possibilidade de considerar indivíduos desconhecidos. No reconhecimento se tem a capacidade de montar um banco de dados contendo as faces dos indivíduos desconhecidos em um ambiente proposto. Desse modo, os indivíduos que vão aparecendo nesse ambiente são verificados para descobrir se há alguma relação facial com os cadastrados no banco (COSTA, 2019).

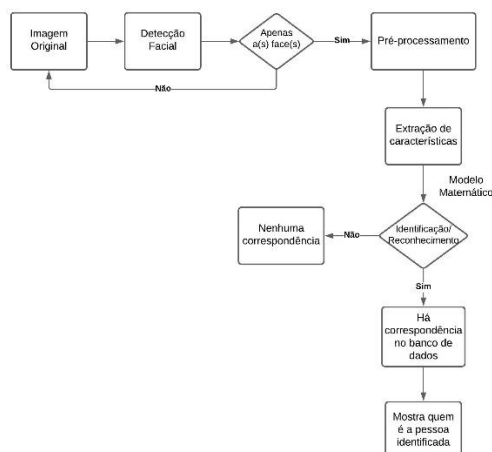
As prováveis tarefas do reconhecimento facial são uma atividade complexa que necessita de processos que compõem diversas etapas em uma ordem específica. A ordem é necessária para a construção do sistema de reconhecimento facial que inicialmente é composto por imagens que serão submetidas a algoritmos computacionais de reconhecimento (COSTA, 2019).

A detecção facial é a primeira atividade do processo de reconhecimento facial, essa irá determinar as características presentes nas faces dos indivíduos, nas imagens do banco de dados. Para melhorar a qualidade das imagens é aplicado um pré-processamento e com isso melhorar o sistema de reconhecimento (GALTON, 1888).

A segunda atividade é relacionada as extrações de características faciais mais relevantes presentes nas imagens pré-processadas armazenadas.

Por último, é desempenhada a classificação e correspondência de faces que foram capturadas e têm as identidades verificadas e validadas como verdadeiras ou não, com o auxílio de algoritmos de classificação de padrões. A Figura 3 mostra diagrama de fluxo do processo de reconhecimento facial.

Figura 3 - Atividades do processo para reconhecimento facial.



Fonte: Elaborado pela Autora.

Como mostrado na Figura 3 a primeira atividade é a detecção facial com o objetivo de detectar faces de humanos para que as imagens presentes no banco de dados sejam usadas nas próximas atividades do reconhecimento facial. Com isso, elimina-se o que não seja face em uma imagem, facilitando as próximas etapas que não precisam de detalhes supérfluos que possam estar presentes em uma imagem ajudando no processo de reconhecimento facial (COSTA, 2019).

## 2.6 Geofencing

*Geofencing* são cercas geográficas digitais predefinidas em um determinado local pelo usuário. Os locais podem ser cidades, bairros, parques ou qualquer local que pode ser identificado no mapa (NAMIOT; SNEPS-SNEPPE, 2013). O *geofencing* pode ser tanto uma ferramenta quanto um software que faz o uso do *Global Positioning System* (GPS), com isso o GPS fornece a localização de qualquer lugar através de satélites espalhados na órbita da Terra que enviam sinais para os receptores GPS, assim as informações recebidas serão usadas nos receptores GPS, para definir os dados e manifestar a localização precisa como a de um indivíduo, desta forma a *geofencing* pode criar uma cerca geográfica virtual em qualquer local no planeta Terra (ZIN *et al.*, 2016).

Entretanto, o GPS não é a única fonte de localização com coordenadas, estas podem ser obtidas também através de *Wi-Fi Protected Setup* (WPS) ou pontos de acesso *Wi-Fi* (NAMIOT; SNEPS-SNEPPE, 2013). Desse modo, a demarcação geográfica compromete um meio para captar um serviço estabelecido dentro de uma zona digital, cercada através das tecnologias GPS,

WPS ou pontos de acesso *Wi-Fi*. Os locais cercados podem ser tanto grandes quanto pequenos dependendo da necessidade do usuário (MAGEE, 2020).

Alguns conceitos para cercas geográficas podem ajudar no seu entendimento, como coordenadas são pontos que têm como base dois números que podem ser negativos e positivos, e são conhecidos como longitude e latitude, esses números representam um ponto específico no mapa, assim uma cerca geográfica é composta de coordenadas que ajudam a definir os limites específicos de uma área. Existem também os Pontos de Interesse (POI) que é uma área específica do mapa conhecida por um público, por exemplo, restaurantes, estádios e hotéis. Pode ser útil por não depender das coordenadas e sim de um ponto específico centralizado em um POI, dessa maneira um aplicativo que tem cercas geográfica centralizadas em um POI para sugestão de restaurante, podem notificar um usuário assim que passar por um restaurante de seu apreço (WU, 2019).

Assim, o uso da *geofencing* pode ser útil em vários casos de uso, alguns exemplos como:

- Empresas que trabalham com entregas de mercadorias, recebe uma notificação quando o entregador sai da rota de entrega.
- Em empresas que contém áreas proibidas existe a necessidade do monitoramento completo. Se algum funcionário não autorizado tentar entrar em alguma área proibida, a identificação do funcionário aciona um alerta para a segurança do local.
- Na publicidade, quando algum cliente entrar em uma área de *geofencing*, uma notificação pode ser acionada para o cliente, caso tenha alguma promoção ou algo especial que esteja contido na loja.
- Em supermercados ou lojas, caso um item saia da área de *geofencing* a etiqueta RFID do item pode enviar um alerta.
- Uma pessoa em prisão domiciliar ao tentar sair do local preestabelecido, a tornozeleira alerta as autoridades.

Tendo em vista as aplicações, as vantagens da *geofencing* são significativas para os negócios, controle nas estradas através do rastreamento de veículo, em vendas para sugerir ofertas e brindes promocionais, é um ótimo recurso utilizando localização em tempo real e dados para melhorar a segurança em qualquer local ou cenário. No entanto, há algumas desvantagens, de maneira que o usuário deve sempre interagir ou aprovar o uso da *geofencing*, o consumo de bateria em dispositivos é relativamente alto pelo uso do GPS e por último, por ter uma limitação de área fixa (ZIN *et al.*, 2016).

No entanto, a *geofencing* pode ser definida por dois grupos, a pessoa desenvolvedora do software que vai ser usado para delimitar uma área virtual ou o cliente final que irá utilizar o aplicativo que será habilitado para responder algum serviço específico. Usuários finais devem ligar os serviços de localização disponíveis em seus dispositivos para que a *geofencing* funcione perfeitamente. Assim, a configuração do desenvolvedor ou do usuário final no aplicativo, com a área já demarcada, pode emitir um alerta ou uma notificação quando se entra na área *geofencing*, esse cenário é conhecido no inglês como “*if this, then that*” (se for isso, faça aquilo), no qual uma ação é baseada em outra ação para ser acionada (REDAÇÃO da CIO/EUA, 2018). Portanto, para utilizar a *geofencing* é necessário o uso de um *Application Programming Interface* (API) que disponibiliza os perímetros necessários para marcar uma área de interesse para que o aplicativo em execução receba as notificações quando cruzar a cerca demarcada. O Google fornece a *Geofencing* API que usa de forma inteligente os sensores disponíveis em um dispositivo para identificar com exatidão a localização do dispositivo sem consumir muito da bateria (GOOGLE DEVELOPER GROUPS, 2020).

## 2.7 *Firestore*

*Firestore* é um serviço proprietário do Google que fornece um comportamento simplificado no desenvolvimento web e móvel, assim pode permitir pessoa desenvolvedora focar em partes menos abstratas e mais visuais, possibilitando mais produtividade a nível técnico. O serviço *Firestore* vem com banco de dados visual, hospedagem, armazenamento de arquivos, processamento do lado do servidor para que a aplicação seja mais protegida dos serviços que eles disponibilizam como autenticação (DECHALERT, 2019).

Desse modo, o *Firestore* é mais conhecido como sendo um *Realtime Database* (Banco de Dados em Tempo real), o que significa que pode ser possível criar aplicações avançadas ao permitir o acesso do banco de dados executando ao lado do código do cliente. Os dados salvos no banco de dados são disponibilizados offline e localmente. Mesmo os dados sendo mantidos assim, os eventos em tempo real podem continuar fazendo requisições, conduzindo uma melhor experiência para o cliente final. Quando a conexão for estabelecida no servidor, o banco de dados em tempo real faz a sincronização das alterações feitas quando estava off-line, e recupera os dados on-line, permitindo as atualizações remotas com êxito (FIREBASE, 2020).

O banco de dados em tempo real é um banco *Not only Structured Query Language* (NoSQL) (FIREBASE, 2020). Sendo assim, NoSQL são bancos de dados não tabulares (que

não se apresenta em forma de tabela), contendo uma variedade de modelos de dados dependendo do tipo chave-valor e possibilitando apenas uma estrutura de dados (SCHAEFER, 2020). O banco de dados *Firebase* tem a forma de um *JavaScript Object Notation* (JSON) o que significa que tem uma forma de objeto conforme apresentado na Figura 4.

Figura 4 - Árvore JSON exemplo.

```
{
  "users": {
    "alovelace": {
      "name": "Ada Lovelace",
      "contacts": { "ghopper": true },
    },
    "ghopper": { ... },
    "eclarke": { ... }
  }
}
```

Fonte: Firebase, 2020.

Dessa forma, os dados são estruturados em um documento com formato JSON, do modo que podem ser lidos ou gravados. Por fim, o serviço *Firebase Authentication*, permite definir o acesso de quais dados se fazem necessários, e como eles são capazes de ser acessados (FIREBASE, 2020).

## 2.8 Interface de Programação de Aplicativos

*Application Programming Interface* (API) pode ser composta em um conjunto de instruções para executar uma determinada tarefa, pode ser considerada uma ligação para dois softwares se comunicarem sem uma ação humana. Dessa forma, uma API funciona como um ponto de entrada para uma determinada parte de um software (MERCIER, 2019).

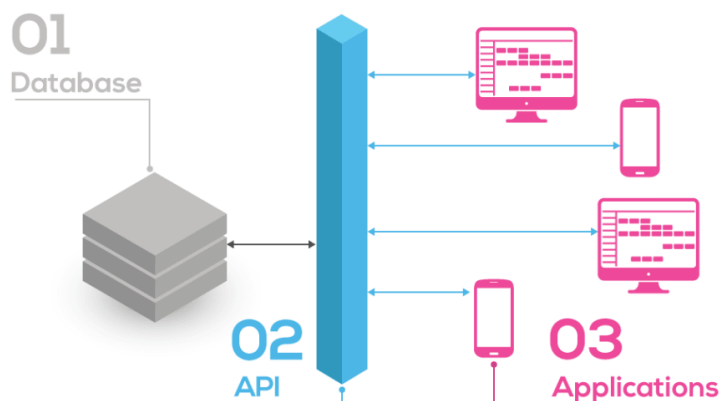
*Application* – a aplicação define qual serviço deve ser usado para interagir com o software, podendo ser fluxo de dados meteorológicos, dados abertos de um portal ou qualquer outra aplicação que disponibiliza compartilhamento de dados.

*Programming* – programação é um conjunto de instruções expressadas através de códigos feitos por uma pessoa desenvolvedora que pode se comunicar com a aplicação, por exemplo, um programa pode solicitar apenas o Código de Endereçamento Postal (CEP) e por meio deste obtém o endereço mais completo.

Interface – uma interface é um meio para o usuário final entrar com os dados e solicitar um serviço, a interface é o que disponibiliza a interação com os recursos disponíveis no serviço que está sendo solicitado (MERCIER, 2019).

A Figura 5 representa a ligação da API entre um uma aplicação e uma base de dados.

Figura 5 - Diagrama do funcionamento de uma API.



Fonte: Mercier, 2019.

Com isso, ao usar uma API pode-se acrescentar funcionalidades específicas e melhorar a fluidez do desenvolvimento. As APIs podem ser uma alternativa na hora do desenvolvimento de um software, evitando criar algo já existente. No geral, as APIs são um conjunto de tarefas específicas que tem como propósito fazer a comunicação entre aplicativos (RAPIDAPI, 2020).

## 2.9 Azure

A Microsoft oferece uma plataforma de serviços na nuvem conhecida como Microsoft Azure, a qual inclui serviços de computação, análise, armazenamento e rede. Além do mais, há várias formas de serviços disponíveis por meio da computação na nuvem, por exemplo, Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS), Software como Serviço (SaaS) sem uso do servidor. Diante disso, ela disponibiliza seus serviços por meio da contratação sendo o preço cobrado com base na utilização dos serviços (ROUSE, 2020).

A plataforma responsável por esses serviços é um console que está disponível na web, mas também pode se utilizar *command line* (linha de comando) para fazer o uso das ferramentas disponíveis online, com a possibilidade de monitoramento e gerenciamento dos recursos solicitados de forma contínua. Além disso, é capaz de gerenciar os serviços de forma programada por meio de modelos, APIs e serviços específicos (MICROSOFT AZURE, 2019).

São incluídos os serviços cognitivos baseados em nuvem e disponibilizados em forma de APIs *Representational State Transfer* (REST) que disponibiliza os dados através de uma requisição *Hyper Text Transfer Protocol* (HTTP) e *Software Development Kit* (SDKs) de bibliotecas que fornece ferramentas, documentação e exemplos de códigos, são serviços que a pessoa desenvolvedora não há necessidade de conhecimento técnico sobre Inteligência Artificial (IA) ou de ciência de dados. Com isso, os serviços cognitivos fazem o entendimento de vários outros serviços de IA e assim permitem o desenvolvedor criar soluções cognitivas, como ouvir, falar, ver, tomar decisões e entender (MICROSOFT AZURE, 2020).

## **2.10 Trabalho relacionado**

Esse tópico apresenta trabalho relacionado ao presente trabalho e um estudo.

### ***2.10.1 Verificação Automática Georreferenciada***

O trabalho de verificação automática georreferenciada teve como objetivo desenvolver um aplicativo móvel para verificar e validar a geolocalização do usuário através de APIs para que realize o acionamento automático de um local com microcontrolador, mostrando em um aplicativo a localização do usuário através do Google Maps sendo possível a demarcação da área desejada para o usuário como apresentado na Figura 6, e a remoção dela (ALVES, 2018).

Figura 6 - Adição da área demarcada.



Fonte: Alves, 2018.

Com isso, ao inicializar o aplicativo, são coletadas as informações da atual localização do usuário com o serviço de localização do próprio aparelho celular, para isso as coordenadas são validadas por meio da API de *Geofencing* verificando se o usuário está na área demarcada ou não. As coordenadas são gravadas no banco de dados *realtime* do *Google Firebase* como valores inteiros para informar o atual status do usuário em associação a área georreferenciada anteriormente (ALVES, 2018).

Para o destravamento da tranca foi utilizado o microcontrolador NodeMCU ESP8266-12E com o desenvolvimento de um software para embarcados capaz de ler os dados gravados no banco de dados *realtime* sendo utilizado a biblioteca “*FirebaseArduino*”, com isso o aplicativo é capaz de ler e decidir se o usuário está na cerca demarcada ou não para ocorrer o destravamento automático do local (ALVES, 2018).



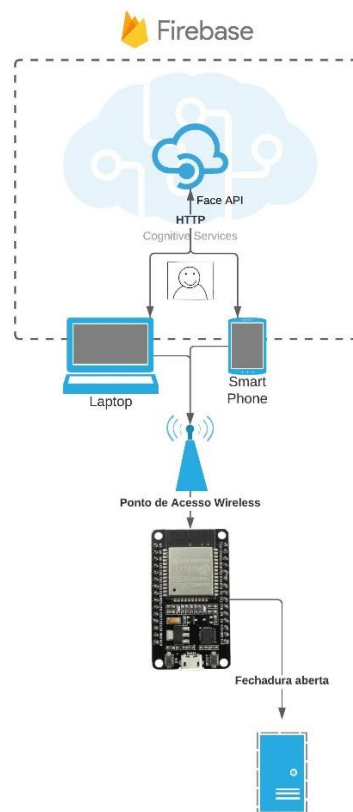
### 3 PROPOSTA DE SOLUÇÃO

Nesse capítulo são apresentadas as ferramentas e tecnologias que estão sendo utilizadas no presente trabalho com base no estado da arte.

O objetivo deste trabalho é desenvolver um software híbrido e um software para hardware. Os dois terão uma comunicação entre si capaz de fazer o controle de um local automaticamente através de uma interface, caso esteja em uma área georreferenciada de determinado indivíduo que está registrado no software híbrido.

A Figura 7 ilustra o projeto onde o smartphone ou laptop irá coletar as informações como o nome, foto do indivíduo, localização demarcada que serão salvas no banco de dados *real time* do Google *Firebase* e irá acionar o mesmo quando estiver perto da localização georreferenciada para confirmar sua identidade. Assim, o microcontrolador ESP32 irá ler o status se a identidade é verdadeira ou falsa e tomará a ação de destravar ou não uma fechadura automaticamente.

Figura 7 - Figura ilustrativa do projeto em geral.



Fonte: Elaborada pela Autora.

### 3.1 Componentes de hardware

Nesse tópico serão apresentados os componentes físicos que irão compor este projeto.

#### 3.1.1 Microcontrolador ESP32

Será utilizado o microcontrolador ESP32 que será desenvolvido um software capaz de ler o status da identificação do indivíduo que está salva no banco de dados *real time* do Google *Firebase*, para isso será utilizada a biblioteca *Firebase ESP32 Client*. Diante disso, será capaz de tomar decisões conforme o status do indivíduo e validar automaticamente pelo software híbrido.

Para efetuar o acionamento automático do local demarcado será feito o uso da biblioteca WiFi.h que permite definir o ponto de acesso ou modo estação será utilizado o modo estação que permite enviar para o ESP32 os dados da rede Wi-Fi que ele irá fazer a conexão, logo depois que estiver configurado através de um portal no navegador. Portanto, o ESP32 funciona como um dispositivo sem fio se conectando a um roteador wireless (NASCIMENTO, 2020).

### 3.2 Componentes de software

Nesse tópico serão apresentados os softwares que irão compor este projeto.

#### 3.2.1 Visual Studio Code

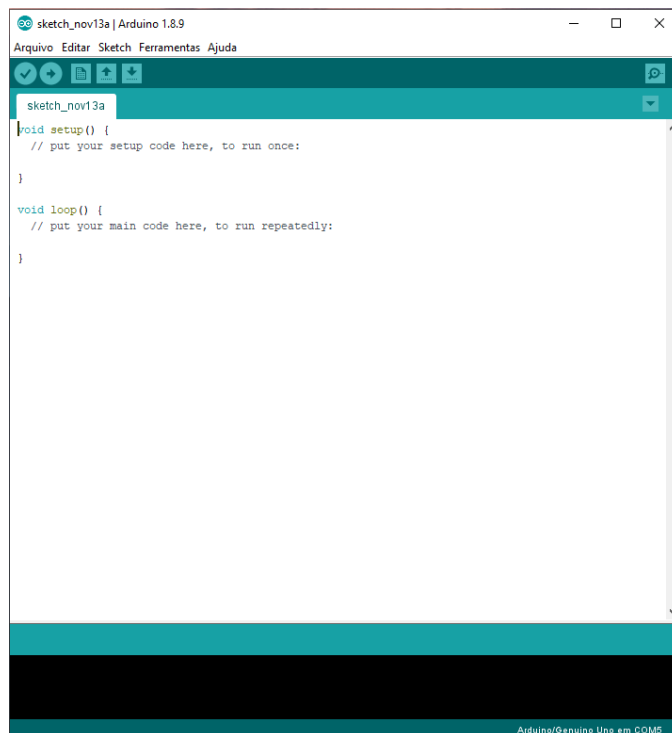
Visual Studio Code é um editor de código-fonte leve e intuitivo para desenvolvimento, estando disponível para os sistemas operacionais macOS, Linux e Windows. Com suporte já integrado para a linguagem *JavaScript* (VISAL STUDIO CODE, 2020).

Ele será utilizado para desenvolvimento do software, sendo utilizado hibridamente com as linguagens HTML, CSS e JS.

#### 3.2.2 Arduino IDE

O Arduino utiliza uma interface *Integrated Development Environment* (IDE) de forma intuitiva e simplificada para melhorar a experiência do usuário, conforme ilustrado na Figura 8. A linguagem utilizada para decodificação foi desenvolvida em JAVA que é derivada de outras duas linguagens, a saber *Processing* e *Wiring* (MATOS AQUINO *et al.*, 2017).

Figura 8 - Interface da IDE do Arduino.

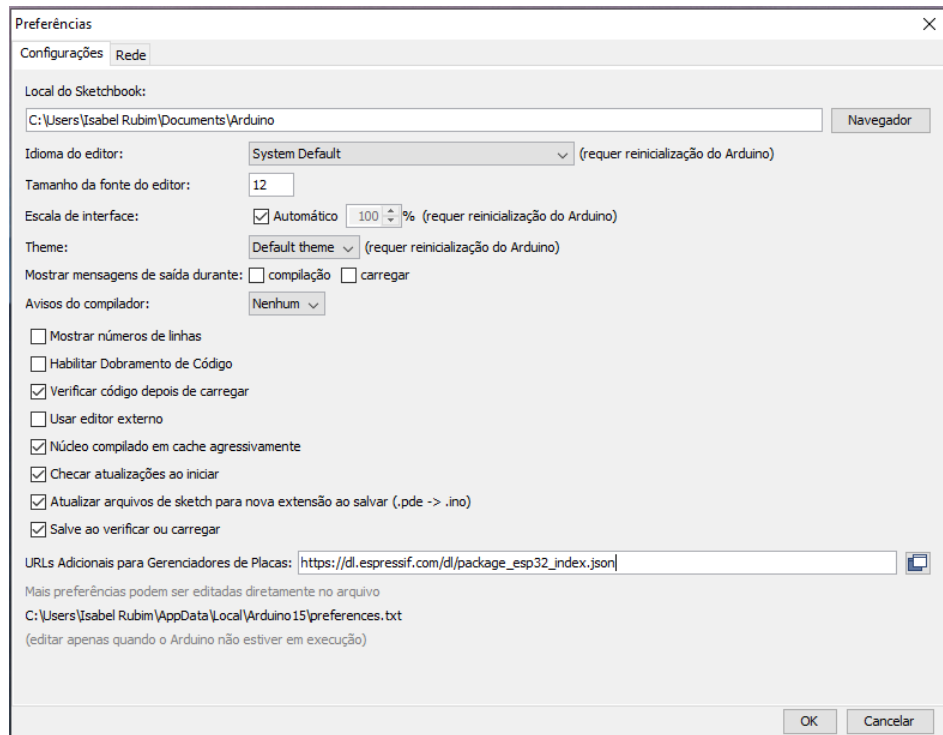


Fonte: Elaborado pela Autora.

A sintaxe de programação dentro da IDE é semelhante as linguagens C/C++, o desenvolvimento do código utiliza termos em inglês como as funções *void setup()* e *void loop()* que são obrigatórias. Para aprimoramento da aplicação, a IDE, utiliza bibliotecas assim como os outros softwares (MATOS AQUINO *et al.*, 2017).

Sendo um software considerado livre que permite a integração com o ESP32 juntamente com as importações da placa no menu de preferências da IDE apresentada na Figura 9.

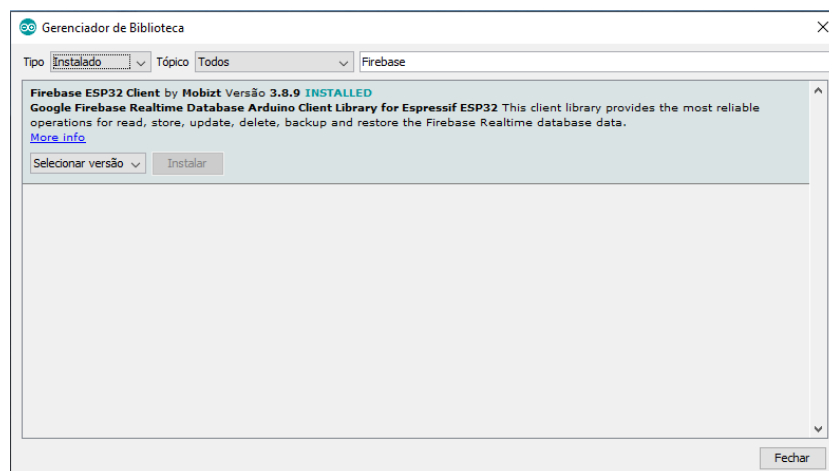
Figura 9 - Gerenciador de placas, necessário para a utilização das bibliotecas do ESP32.



Fonte: Elaborado pela Autora.

Será necessário também o uso da instalação da biblioteca *Firestore ESP32 Client*, para isso é necessário ser instalada no menu gerenciador de biblioteca na IDE, como é mostrado na Figura 10.

Figura 10 - Tela de gerenciamento de bibliotecas.



Fonte: Elaborado pela Autora.

A partir dessas configurações, o projeto pode ser elaborado com a devida programação para o ESP32 conectar-se ao Wi-Fi que já vem integrado para se conectar à rede *World Area Network* (WAN) e ao *Firebase* destinado a gravar ou ler os dados do banco de dados *realtime*.

### 3.2.3 Google *Firebase*

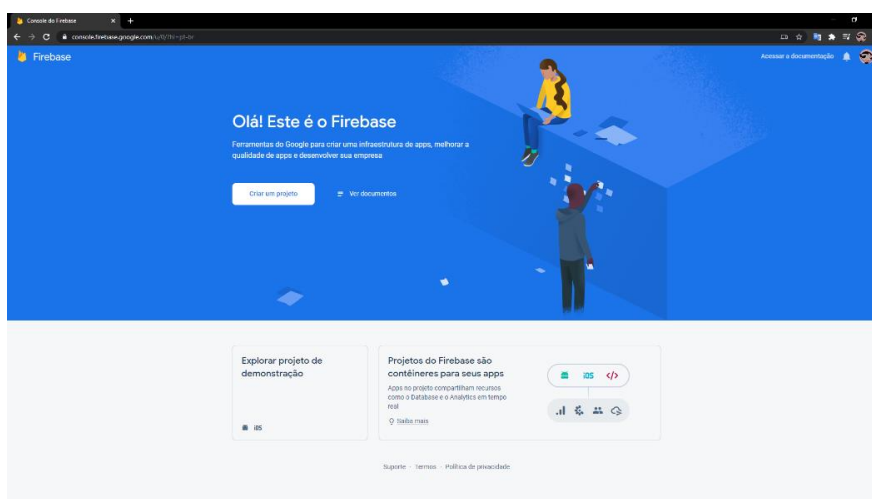
O *Firebase* será utilizado como plataforma de armazenamento de dados em nuvem, o banco da Google *realtime* se apresenta flexível para softwares híbridos. A conectividade vai consistir entre dispositivos físicos em rede e interface via web, formando uma ponte entre a interface via web e os dispositivos para o funcionamento correto do sistema.

Além disso, o presente banco de dados, ao ser usado em uma aplicação suporta seis linguagens de programação que são *Swift*, *Objective-C*, *Java*, *JavaScript*, *C++* e *Unity* (FIREBASE, 2020). O presente trabalho usa a linguagem de programação *JavaScript* para a integração do banco de dados *realtime Firebase*.

As funcionalidades do *Firebase* que serão usadas para este projeto, são:

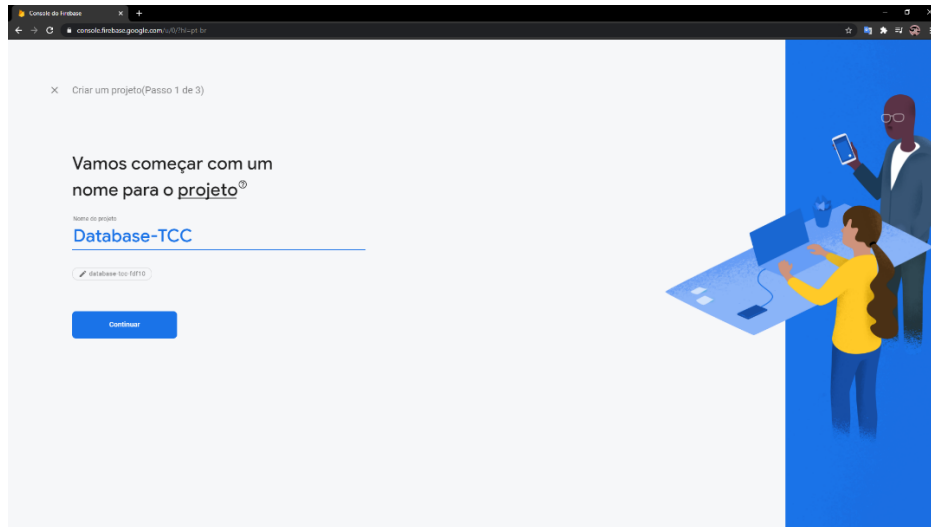
- Console – plataforma via web que permite inicialmente criar um projeto conforme mostra a Figura 11 e Figura 12, além disso, mostra o funcionamento do projeto e os dados armazenados;
- *Cloud Firestore* – banco de dados *NoSQL* que faz o armazenamento em forma do *JavaScript Object Notation* (JSON);
- *Realtime* – faz a gravação dos dados no banco em tempo real.

Figura 11 - Tela para criar um projeto no *Firebase*.



Fonte: Elaborado pela Autora.

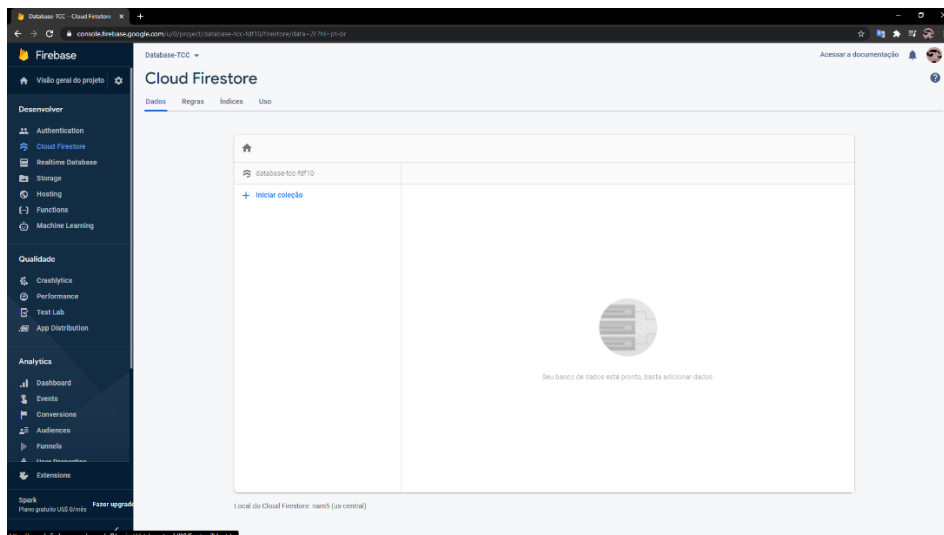
Figura 12 - Tela para definir o nome do projeto *Firebase*.



Fonte: Elaborado pela Autora.

Para o uso do *Cloud Firestore* é necessário a inicialização dele na página inicial depois da criação do projeto como mostra Figura 13 o *Cloud Firestore* está inicialmente sem qualquer coleção criada, será criada posteriormente no presente trabalho, para salvar os dados dos usuários como nome, e-mail, URL da foto do seu rosto e uma identificação gerada pelo *Cloud Firestore* na coleção *users* (UID).

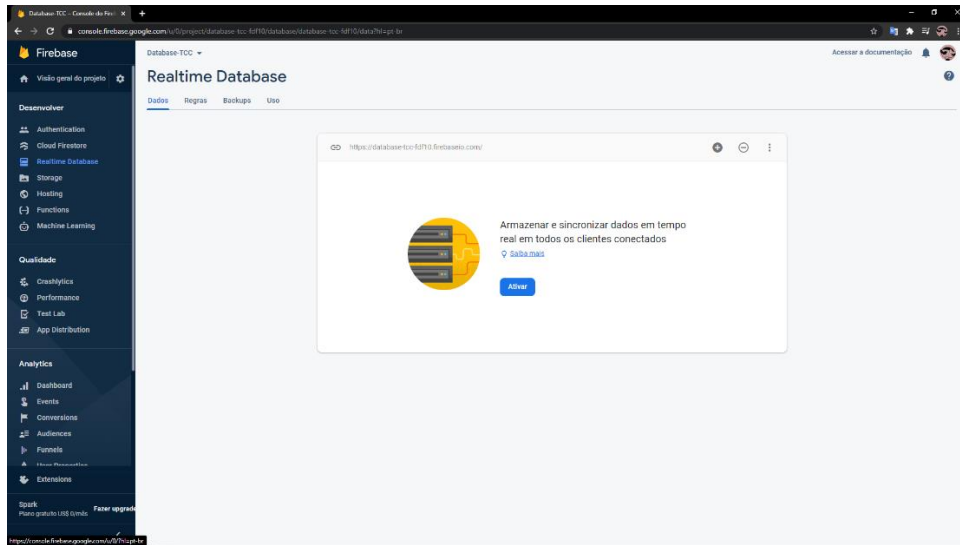
Figura 13 - Tela do *Cloud Firestore* do *Firebase*.



Fonte: Elaborado pela Autora.

O *Realtime Database*, apresentado na Figura 14, é necessário para ativar antes de iniciar as gravações em tempo real.

Figura 14 - Tela *Realtime Database* do *Firebase*.



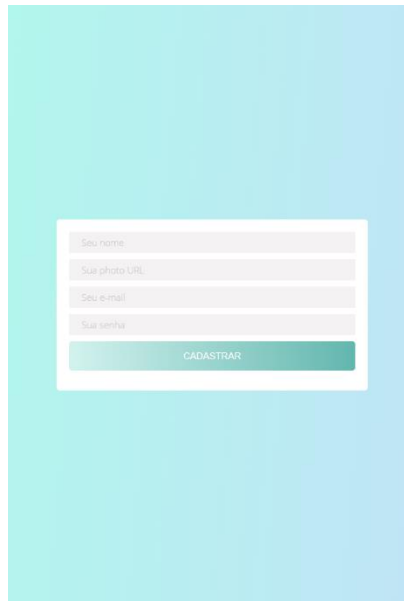
Fonte: Elaborado pela Autora.

O *Realtime Database* é utilizado para coletar os dados da área demarcada pelo usuário, como latitude, longitude, raio e uma *flag* indicando quando o usuário estiver dentro da área demarcada, para que indique assim que estiver no ponto georreferenciado salvo anteriormente.

### 3.2.4 *Cloud Firestore*

Na aplicação desenvolvida, o uso do *Cloud Firestore* é utilizado para o usuário se cadastrar na aplicação e consequentemente realizar um login com seus dados cadastrados, na tela de cadastro apresentada na Figura 15 é solicitado o nome, e-mail, senha, e uma URL da sua foto que contenha seu rosto para posteriormente ser verificada com sua foto tirada dentro da aplicação para realizar o destrancamento do local.

Figura 15 - Tela de cadastro do usuário.

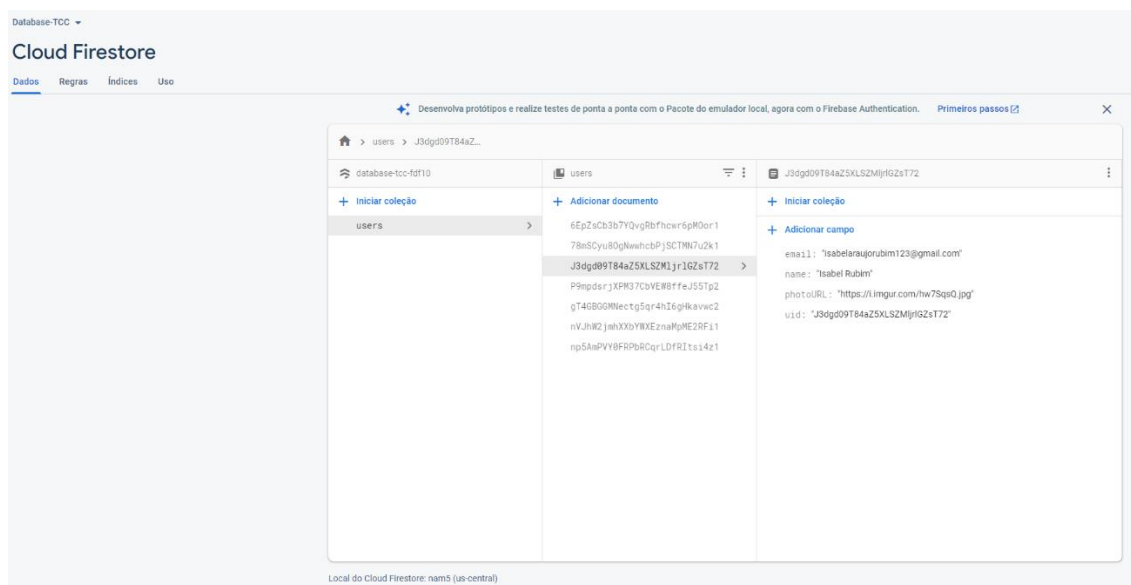


A tela de cadastro do usuário apresenta um formulário centralizado sobre um fundo com gradiente de verde para azul. O formulário contém quatro campos de entrada de texto empilhados verticalmente, rotulados como 'Seu nome', 'Sua photo URL', 'Seu e-mail' e 'Sua senha'. Abaixo dos campos, há um botão de ação verde com o texto 'CADASTRAR' em letras maiúsculas brancas.

Fonte: Elaborado pela Autora.

Após o usuário realizar o cadastro, os dados são salvos no documento com um ID gerado para coleção *users* criada para salvar os usuários cadastrados, como pode ser visto na Figura 16.

Figura 16 - Dados do usuário cadastrado.



Fonte: Elaborado pela Autora.



Com os dados salvos, o usuário poderá realizar o login com seu e-mail e senha cadastrados, a tela de login é apresentada na Figura 17.

Figura 17 - Tela de login.



A imagem mostra uma interface de login centralizada em uma tela com fundo degradado de verde-água para azul. O formulário de login contém dois campos de entrada: o primeiro para o e-mail, com o texto "isabelarajerubim123@gmail.com" preenchido, e o segundo para a senha, com pontos cinza representando caracteres ocultos. Abaixo dos campos há um botão verde com o texto "ENTRAR" em branco. Na base do formulário, há um link que diz "Não tem cadastro? Cadastre-se".

Fonte: Elaborado pela Autora.

O código desenvolvido para realização do cadastro e login foi feito na *class* Auth.js apresentada na Figura 18.

Figura 18 - Class Auth.js.

```

import firebase from 'firebase/app';
import 'firebase/firestore';
import 'firebase/auth';

export default class Auth {
  constructor() {
    // Referências para o banco de dados e para os métodos disponibilizados para o auth
    this.db = firebase.firestore();
    this.auth = firebase.auth();

    // Pega a coleção users criada no Cloud Firestore
    this.users = this.db.collection('users');

    // Disponibiliza os dados em cache
    firebase.firestore().enablePersistence();
  }

  /**
   * @param {string} email - email@example.com
   * @param {string} password - *****
   * @param {string} customData - photoURL, name, uid
   */
  async signUp(email, password, customData) {
    try {
      // Cadastro do usuário com email, senha e dados customizados
      const credential = await this.auth.createUserWithEmailAndPassword(email, password);
      if (credential.user) {
        this.setUserData(credential.user, customData);
      }
    } catch (error) {
      console.log('error to credential', error);
    }
  }

  /**
   * @param {string} email
   * @param {string} password
   * @returns {email, name, photoURL, uid.}
   */
  async signIn(email, password) {
    try {
      // Faz o login do usuário
      const credential = await this.auth.signInWithEmailAndPassword(email, password);
      if (credential.user) {
        this.user = await this.getUser(credential.user);
        return this.user;
      }
    } catch (error) {
      console.log('error in signIn', error);
    }
  }

  // Obtém as propriedades do usuário salva no banco de dados cloud firestore
  async getUser({uid}) {
    const user = await this.users.doc(uid).get();
    if (user.exists) {
      return user.data();
    }
  }

  // Desloga o usuário
  async signOut() {
    try {
      await this.auth.signOut();
    } catch (error) {
      console.log('error in logout');
    }
  }

  //
  setUserData(user, customData) {
    // Adiciona propriedades do usuário no banco de dados cloud firestore
    const userRef = this.users.doc(user.uid);
    const data = {
      uid: user.uid,
      email: user.email,
      name: customData.name,
      photoURL: customData.photoURL
    };
    userRef.set(data, { merge: true });
  }
}

```

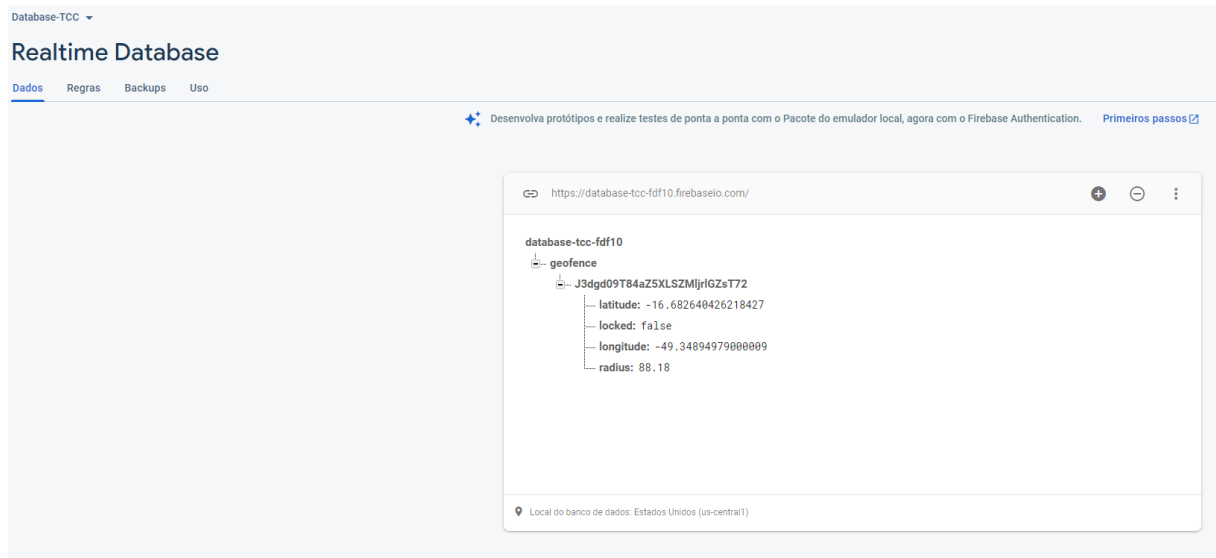
Fonte: Elaborado pela Autora.

As duas funções principais *signOut* e *signIn* têm como função realizar o cadastro do usuário e enviar os dados do usuário para o *Cloud Firestore*, conseqüentemente, realizar o login com e-mail e senha.

### 3.2.5 Realtime Database

Na aplicação desenvolvida, o *Realtime Database* foi utilizado para salvar os dados da área demarcada pelo usuário, *geofence*. Os dados salvos da área demarcada são latitude, longitude, raio e uma *flag* booleana indicando se o local está trancado ou destrancado como é apresentado na Figura 19.

Figura 19 - Banco de dados *Realtime* com os dados salvos da área demarcada.



Fonte: Elaborado pela Autora.

Os dados são salvos como uma árvore JSON, no primeiro nó da árvore *geofence*, tem-se o UID para a identificação do usuário, indicando dos dados da sua área georreferenciada e a *flag locked* que será usada posteriormente em conjunto com o ESP32 para destrancar o local.

O código desenvolvido para a integração do banco de dados *Realtime* é mostrado na Figura 20.

Figura 20 - Class RealtimeDb.js.

```

import firebase from 'firebase/app';
import 'firebase/database';

export default class RealTimeDb {
  constructor() {
    // Referência para o banco de dados Realtime
    this.dbRealTime = firebase.database();
  }

  /**
   * @param {number} latitude
   * @param {number} longitude
   * @param {number} raio
   * @param {string} uid
   * @param {boolean} locked
   */
  async saveGeoFence({ latitude, longitude, radius }, uid, locked) {
    try {
      // Salva os dados no banco de dados Realtime, como referência o uid do usuário
      await this.dbRealTime.ref('geofence/${uid}').set({
        latitude,
        longitude,
        radius,
        locked
      });
    } catch (error) {
      console.log('error in save geofence', error);
    }
  }

  async getGeoFence(uid) {
    try {
      // Obtém os dados do usuário salvo no banco de dados Realtime
      await this.dbRealTime.ref('geofence/${uid}').once('value').then(snapshot => {
        this.markedArea = snapshot.val();
      });
    } catch (error) {
      console.log('error in get geofence', error);
    }
  }
}

```

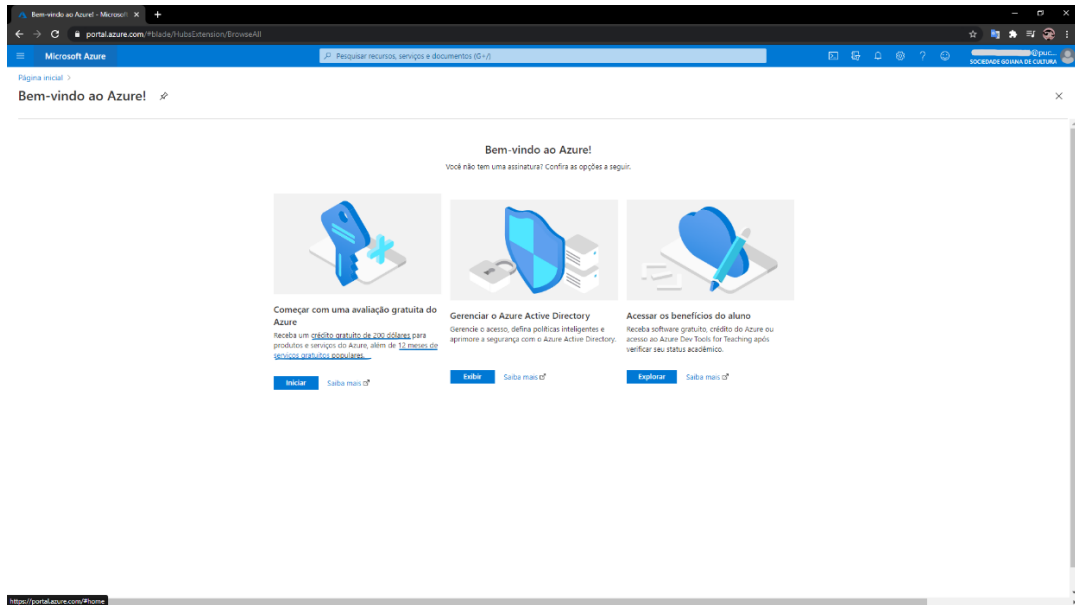
Fonte: Elaborado pela Autora.

A função *saveGeoFence* salva os dados da área georreferenciada referenciando o UID do usuário para em seguida a função *getGeoFence* obter os dados salvos a partir do UID do usuário.

### 3.2.6 Portal Azure

O portal do Azure é um console que apresenta todos os serviços que são disponibilizados pela Microsoft. Ao iniciar a conta na Microsoft Azure, é disponibilizado um crédito de 200 dólares para os serviços e produtos do Azure, além disso os serviços gratuitos populares ficam disponíveis durante 12 meses conforme é apresentado na Figura 21 a qual apresenta a tela inicial do Azure.

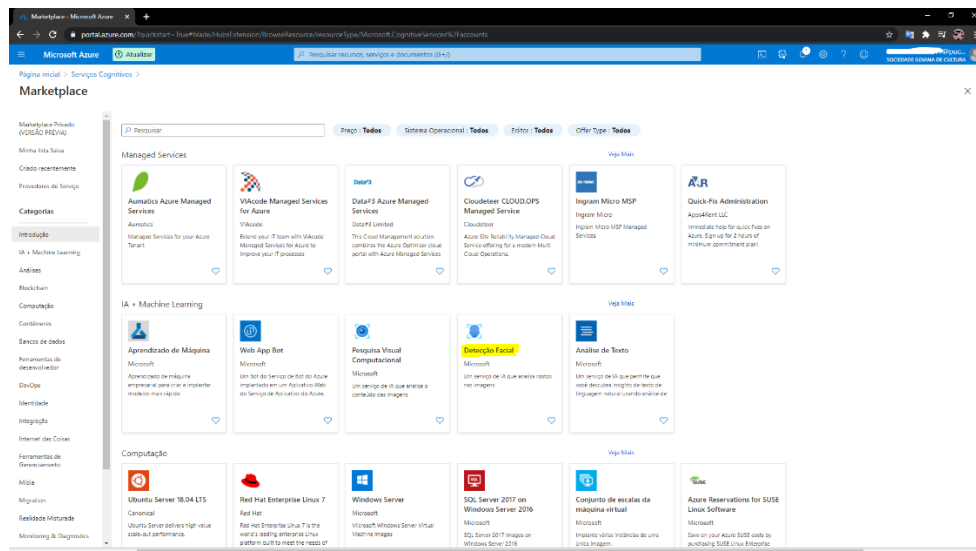
Figura 21 - Tela inicial do portal do Azure para iniciar a avaliação gratuita.



Fonte: Elaborado pela Autora.

No presente trabalho, utiliza a avaliação gratuita que inclui o serviço cognitivo Detecção Facial que pode ser encontrado no *Marketplace* dos serviços cognitivos como apresentado na Figura 22.

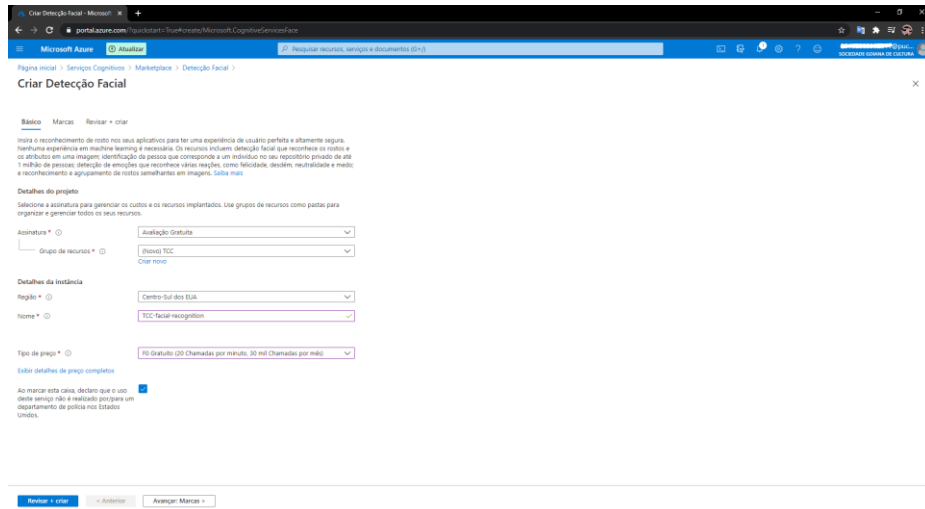
Figura 22 - *Marketplace* dos serviços cognitivos encontrados no portal do Azure.



Fonte: Elaborado pela Autora.

Para ser utilizada a detecção facial será necessário criar o serviço como está apresentado na Figura 23.

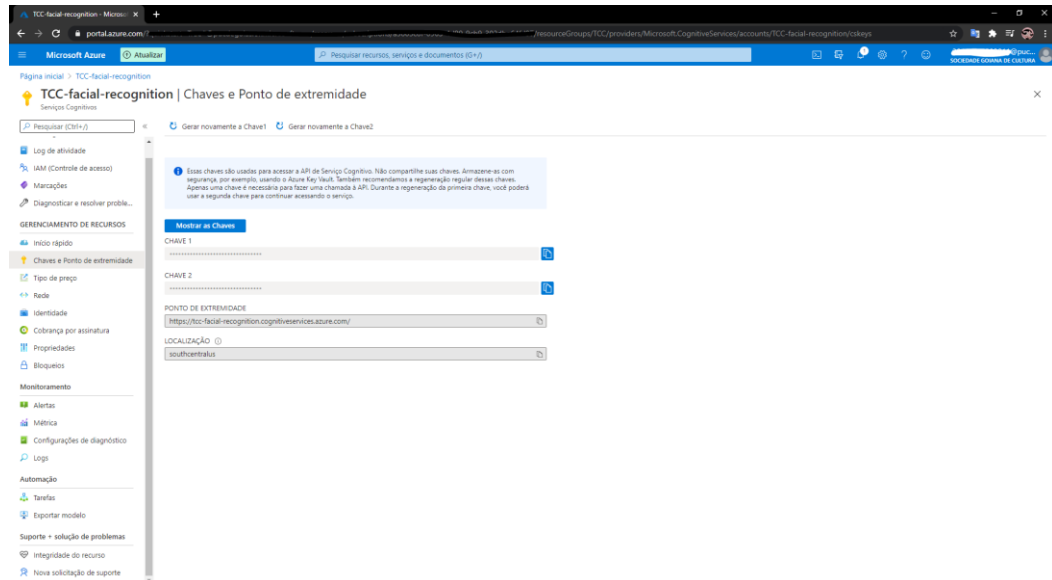
Figura 23 - Tela para criar o serviço cognitivo Detecção Facial no portal do Azure.



Fonte: Elaborado pela Autora.

Depois que o serviço detecção facial é criado, o Azure disponibiliza duas chaves para a integração com a aplicação desenvolvida, isso é apresentado na Figura 24.

Figura 24 - Tela de acesso as chaves e ponto de extremidade no portal do Azure.



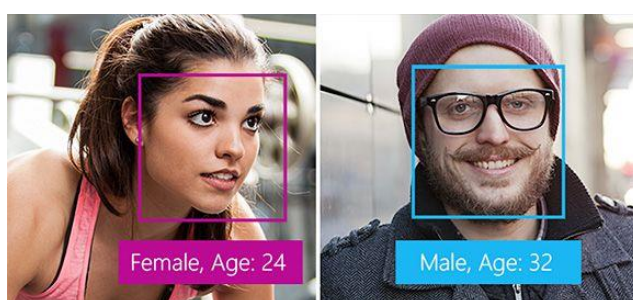
Fonte: Elaborado pela Autora.

As chaves disponíveis podem ser utilizadas, tanto a primeira quanto a segunda, mas apenas uma pode ser utilizada em conjunto com o ponto de extremidade (*endpoint*). A partir dessas configurações o projeto pode ser desenvolvido usando o serviço cognitivo de detecção facial disponibilizado pelo Azure.

### 3.2.7 Detecção Facial do Azure

O serviço de Detecção Facial do Azure foi elaborado com algoritmos de IA para realizar a detecção, reconhecimento e análise de rostos humanos nas imagens. Quanto a detecção facial essa detecta os rostos presentes em uma imagem e retorna as coordenadas onde ocorreu a detecção na imagem. Além disso, após detectar, o serviço retorna características relacionadas da pessoa, como mostra na Figura 25 (MICROSOFT AZURE, 2020).

Figura 25 - Detecção Facial com características relacionadas do rosto



Fonte: Azure, 2020.

Entretanto, há alguns pontos para se levar em consideração ao detectar um rosto, são eles:

- A imagem a ser detectada deve estar em alguns desses formatos: PNG, JPEG, GIF para o primeiro quadro e BMP;
- O tamanho máximo da imagem deve ser 6 MB;
- A variação do tamanho dos rostos devem estar entre 36 x 36 a 4096 x 4096;
- Se o rosto estiver muito inclinado ou sendo obstruído com óculos ou com as mãos, não será possível a detecção, sendo melhor vistas de rostos frontais para que se obtenha melhores resultados;
- Os vídeos devem estar sem o efeito de suavização;
- A velocidade do obturador recomendada é 1/60 segundo ou mais rápido;
- Ângulo do obturador deve preferivelmente estar focado na parte inferior do rosto, resultando em vídeos mais claros.

Para cada face detectada é gerado um identificador *hash* único, Face ID, de tal forma que pode ser solicitado em uma chamada na *Face* API (MICROSOFT AZURE, 2019).

A verificação facial é executada logo depois de detectar um rosto, ela é executada para verificar dois rostos ou um rosto associado a um indivíduo. Consequentemente, a verificação facial realiza a avaliação se os dois rostos pertencem ao mesmo indivíduo (MICROSOFT AZURE, 2020). Para a verificação é usado o mesmo *Face ID* que foi gerado ao detectar (MICROSOFT AZURE, 2019).

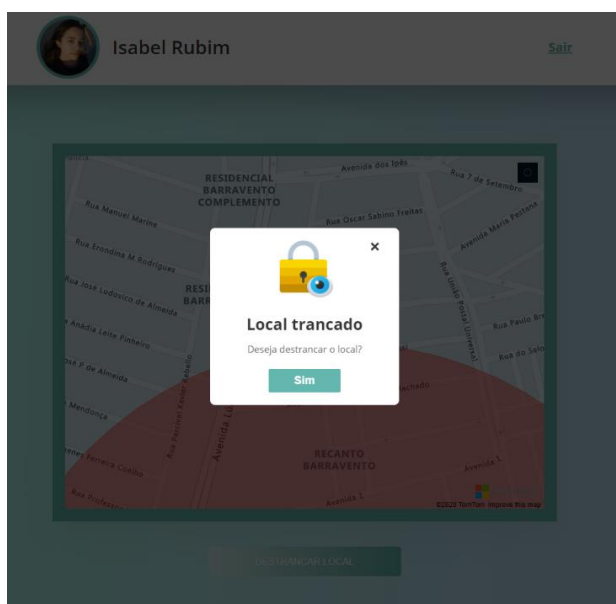
Para uma verificação mais eficaz, alguns pontos devem ser considerados além dos pontos de detecção facial, como:

- Não pode ter baixa iluminação nas imagens;
- Não pode ter diferenças nos tipos de cabelo ou cabelo no rosto;
- Não pode ter rostos diferentes devido a alteração de idade;
- Não pode ter expressões faciais que modifique o rosto (MICROSOFT AZURE, 2019).

O Reconhecimento facial se faz em conjunto com a detecção e a verificação facial. No presente trabalho será usado a *Face API* que contém a documentação de todas as chamadas necessárias para fazer o reconhecimento facial, os dados gravados ao fazer o reconhecimento facial serão gravados na nuvem e podem ser referenciados com seus respectivos ID.

Na aplicação do presente trabalho, o usuário terá a opção de destrancar o local que foi desejado e está dentro da cerca virtual, inicialmente, o local vai iniciar trancado como mostra na Figura 26.

Figura 26 - Local trancado para o usuário.

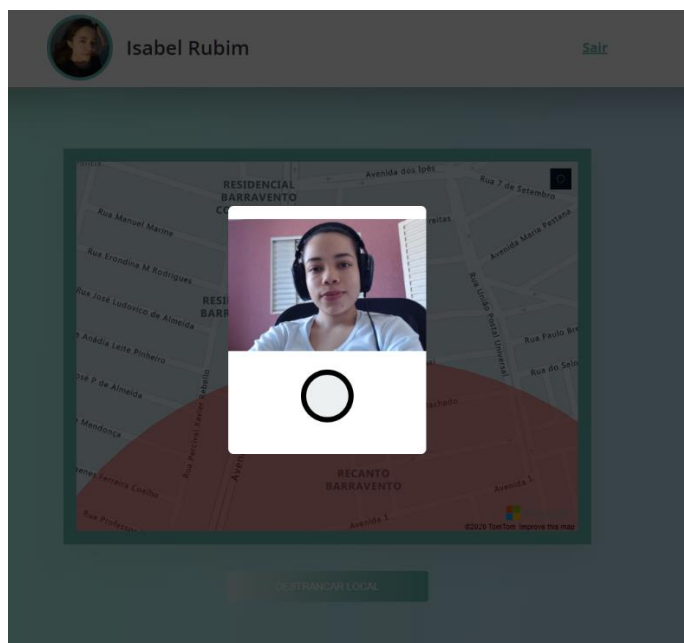


Fonte: Elaborado pela Autora.



Assim que o usuário decidir destrancar o local, clicando no sim, a câmera frontal será aberta para o usuário tirar sua foto sendo apresentado na Figura 27.

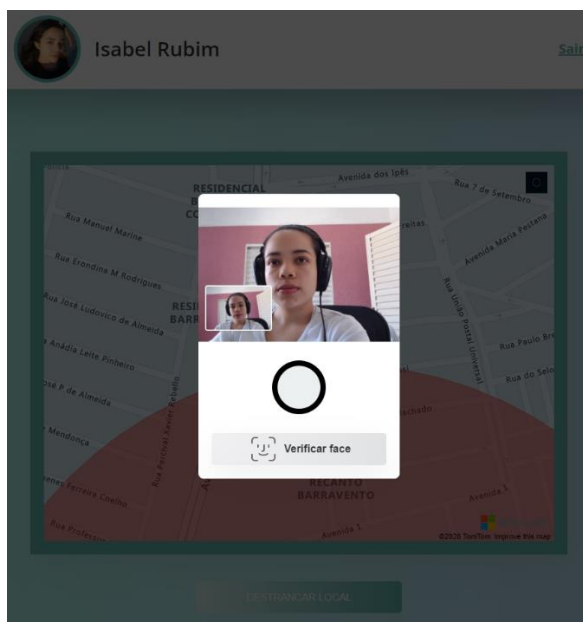
Figura 27 - Câmera frontal ligada.



Fonte: Elaborado pela Autora.

Ao tirar a foto, será exibido um botão, se o usuário deseja destrancar o local sendo apresentado na Figura 28, caso seja clicado no botão de verificar face será feita a verificação da foto tirada para gerar seu ID e comparar com o ID da verificação da foto cadastrada e salva inicialmente no banco de dados *Firebase* como uma URL.

Figura 28 - Foto tirada para a verificação da face.



Fonte: Elaborado pela Autora.

Caso as verificações das fotos sejam correspondidas com o valor do *isIdentical* igual a *true* da API da detecção facial e o usuário dentro da cerca virtual demarcada anteriormente, o local será destrancado com sucesso.

O código para a verificação facial foi desenvolvido na *class* chamada *Face.js*, apresentada na Figura 29 com seus respectivos comentários.

Figura 29 - Class Face.js.

```

import { detectUrl, verifyUrl, subscriptionKeyFace } from '../config';
export default class Face {
  /**
   * @param {string} url - URL salva no banco de dados Firebase
   */
  async detectFaceUrl(url) {
    // Faz a detecção do rosto presente na foto salva
    try {
      /**
       * Chamada na Face API com o endpoint para detecção dos rostos nas fotos,
       * com o cabeçalho na chamada para representação do dado do tipo JSON,
       * recebendo a URL.
       */
      const faceDetect = await fetch(detectUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Ocp-Apim-Subscription-Key': subscriptionKeyFace,
        },
        body: JSON.stringify({ url }),
      });
      const data = await faceDetect.json();
      if (data) {
        this.faceIdURL = data[0].faceId;
      }
    } catch (error) {
      console.log('error in detect URL', error);
    }
  }
  /**
   * @param {Blob} blob - objeto do tipo arquivo, com dados brutos (mutáveis).
   */
  async detectFace(blob) {
    // Faz a detecção do rosto presente na foto tirada
    try {
      /**
       * Chamada na Face API com o endpoint para detecção dos rostos nas fotos,
       * com o cabeçalho na chamada para representação do dado tipo binário,
       * recebendo o blob.
       */
      const faceDetect = await fetch(detectUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/octet-stream',
          'Ocp-Apim-Subscription-Key': subscriptionKeyFace,
        },
        body: blob,
      });
      const data = await faceDetect.json();
      if (data) {
        this.faceId = data[0].faceId;
      }
    } catch (error) {
      console.log('error in detect', error);
    }
  }
  async identifyFace() {
    // Faz a identificação do rosto presente na foto tirada e salva
    try {
      const paramsFaceIds = {
        faceId: this.faceId,
        faceId2: this.faceIdURL,
      };
      /**
       * Chamada na Face API com o endpoint para verificação das fotos,
       * com o cabeçalho na chamada para representação do dado tipo JSON,
       * recebendo os ids da detecção facial.
       */
      const faceVerify = await fetch(verifyUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Ocp-Apim-Subscription-Key': subscriptionKeyFace,
        },
        body: JSON.stringify(paramsFaceIds)
      });
      // Resultado se as fotos são correspondentes representado no valor isIdentical.
      this.faceVerifyResult = await faceVerify.json();
    } catch (error) {
      console.log('error in identify face', error)
    }
  }
  /**
   * @param {string} dataURL - URL da foto tirada
   * @returns arquivo do tipo Blob
   */
  async makeBlob(dataURL) {
    try {
      // Transforma a URL da foto tirada em um tipo Blob.
      const response = await fetch(dataURL);
      const responseBlob = await response.blob();
      return responseBlob;
    } catch (error) {
      console.log('error to make blob', error);
    }
  }
}

```

Fonte: Elaborado pela Autora.

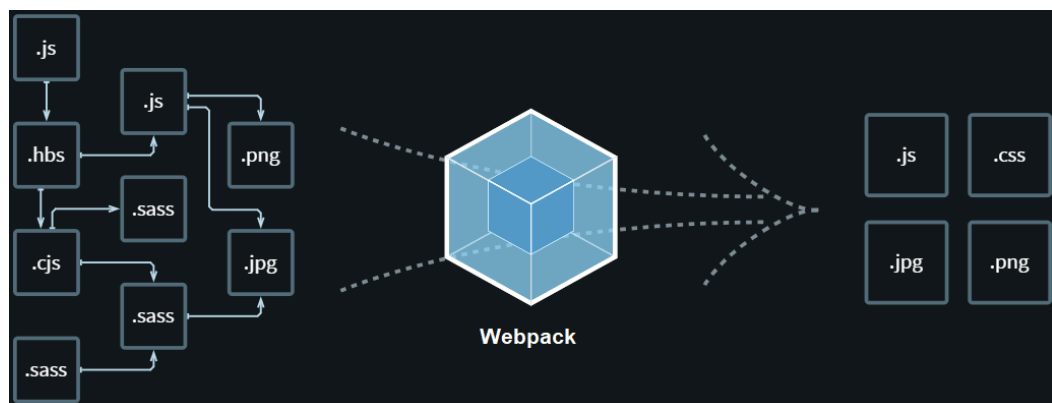
As funções *detectFaceUrl* e *detectFace* fazem as detecções dos rostos nas fotos, quando o usuário clicar no botão verificar face elas serão chamadas, a *detectFace* detecta o rosto da foto que foi tirada no momento que o usuário gostaria de destrancar o local e assim chamando a função *makeBlob* para que a foto tirada seja convertida em um arquivo do tipo *Binary Large Object* (BLOB) para que a Face API aceite e gere um ID da foto tirada e assim ser chamada a

função *identifyFace* que recebe os dois IDs um da foto salva obtida na *detectFaceUrl* e outro da *detectFace*. Realizando a comparação dos IDs a função *identifyFace* retorna um objeto JSON com dois valores representado como *isIdentical* e *confidence*, se o valor da *isIdentical* for igual *true* significa que os rostos das duas fotos são da mesma pessoa, caso contrário será *false*, para o valor da *confidence* representa-se o nível de confiança que pode variar entre 0 até 1, sendo a metade da unidade faz com que *isIdentical* seja *true*.

### 3.2.8 Webpack

A funcionalidade do *webpack* é tornar todos os arquivos que estão em módulos em um único pacote ou mais em aplicações *JavaScript*. Na Figura 30, mostra-se a representação dos pacotes utilizando o *webpack* (THELARKLNN, 2021). Os módulos são divisões claras do que determinado código faz do aplicativo em geral. Para o *webpack*, os módulos podem representar suas dependências em arquivos *JavaScript* usando uma declaração *import* que cria uma dependência dentro do arquivo do próprio código criado. Em arquivos de estilos (CSS, SASS, LESS), por exemplo, quando se trabalhar com estilos deve se usar o *@import* dentro do arquivo (THELARKLNN, 2021).

Figura 30 - Representação da funcionalidade do *webpack*.



Fonte: *Webpack*.

Para o *webpack* ser usado em uma aplicação, é necessário um arquivo de configuração chamado *webpack.config.js*, com as determinadas configurações. No aplicativo desenvolvido neste trabalho, as seguintes configurações foram usadas de acordo com a própria documentação do *webpack*, os principais conceitos serão apresentados nessa seção em sequência.

Primeiramente, foi usado o ponto de entrada ou *entry* que indica em qual módulo o *webpack* deve iniciar a construção do gráfico empacotando os módulos dependentes (THELARKLNN, 2021). Na Figura 31, representam-se o código do aplicativo que há dois pontos de entradas, um `@babel/polyfill` que será discutido na próxima seção e um arquivo `index.js`.

Figura 31 - Ponto de entradas para a configuração do *webpack*.

```
module.exports = (_, { mode }) => ({
  entry: [
    '@babel/polyfill',
    './src/js/index.js'
  ],
```

Fonte: Elaborado pela Autora.

Em seguida, o output cria a pasta `dist` e encapsula, ou seja, faz *webpack*, para saber onde os pacotes serão criados e como serão emitidos dentro desta pasta `dist` gerada (THELARKLNN, 2021). Na aplicação o nome do arquivo empacotado terá o nome *bundle* que vai estar dentro de `./dist/js/bundle.js`.

Figura 32 - Saída para a configuração do *webpack*.

```
output: {
  path: path.resolve(__dirname, 'dist'),
  filename: 'js/bundle.js'
},
```

Fonte: Elaborado pela Autora.

Os *loaders* têm como função permitir que o *webpack* processe outros tipos de arquivos, como SASS/CSS, PNG/JPG/SVG e JSON, e assim torná-los módulos que podem ser usados na aplicação, já que ele processa apenas arquivos *JavaScript*, para que possam ser utilizados no fluxo de dependência apresentado genericamente na Figura 30. A nível de código os *loaders* possuem duas propriedades na configuração do *webpack*: a *test*, que diz para o *webpack* quando encontrar o caminho do arquivo que está em um *import* ou *require*; e a propriedade *use*, que utiliza-se do uso das declarações na chave *use*, para modificar o contexto antes de adicionar ao

pacote do *webpack* (THELARKLNN, 2021). Para a aplicação foi usado os *loaders* que estão na Figura 33, observar os comentários nessa figura.

Figura 33 - Implementação dos *loaders* no arquivo de configuração do *webpack*.

```

module: {
  rules: [
    {
      // Este pacote permite transpilar arquivos JavaScript
      test: /\.m?js$/,
      exclude: /node_modules/,
      use: {
        loader: 'babel-loader',
        options: {
          presets: ['@babel/preset-env'],
        },
      },
    },
    {
      /**
       * - style-loader é usado para adicionar a tag <style>
       * ao arquivo html na tag <head>.
       * - css-loader lê em um arquivo css como uma string
       * - sass-loader carrega um arquivo SASS/SCSS e o compila em CSS
       */
      test: /\.s[ac]ss$/i,
      use: [
        mode !== 'production'
          ? 'style-loader'
          : MiniCssExtractPlugin.loader,
        'css-loader',
        'sass-loader',
      ],
    },
    {
      /**
       * - file-loader resolve import/require()
       * em um arquivo em uma url e emite o arquivo no diretório de saída
       */
      test: /\.(png|jpg|svg)$/i,
      use: [
        {
          loader: 'file-loader',
          options: {
            name: '[path][name].[ext]',
            context: path.resolve(__dirname, "src/"),
            outputPath: '/',
            publicPath: '../',
            useRelativePaths: true
          }
        }
      ]
    }
  ]
}

```

Fonte: Elaborado pela Autora.

Os plugins no *webpack* têm como finalidade executar diversas tarefas, e até tarefas que os *loaders* não poderiam realizar (THELARKLNN, 2021). Na aplicação desenvolvida foram utilizados vários plugins, cada um destes determina uma função no aplicativo, na Figura 34 mostra-se os plugins utilizados e seus comentários.

Figura 34 - Plugins implementados no arquivo de configuração do *webpack*.

```

plugins: [
  /**
   * O plugin irá gerar um arquivo HTML5 para que
   * inclua todos os pacotes webpack no corpo usando script tags.
   */
  new HtmlWebpackPlugin({
    filename: 'index.html',
    template: './src/index.html'
  }),
  /**
   * Este plugin extrai CSS em arquivos separados.
   * Ele cria um arquivo CSS por arquivo JS que contém CSS.
   */
  new MiniCssExtractPlugin({
    filename: 'css/[name].css',
    chunkFilename: 'css/[id].css',
  }),
  /**
   * O plug-in criará um arquivo de service worker chamado sw
   * e o adicionará ao pipeline de ativos do webpack.
   */
  new WorkboxPlugin.GenerateSW({
    swDest: 'sw.js',
    clientsClaim: true,
    skipWaiting: true,
    maximumFileSizeToCacheInBytes: 5000000,
    runtimeCaching: [{
      urlPattern: new RegExp('^https://fonts\\.googleapis\\.com/'),
      handler: 'StaleWhileRevalidate',
      options: {
        cacheName: 'google-fonts',
        cacheableResponse: {
          statuses: [0, 200],
        }
      }
    }
  ]
  }),
  /**
   * Plugin que gera um 'manifest.json' para o Progressive Web Application,
   * com redimensionamento automático de ícones.
   */
  new WebpackPwaManifestPlugin({
    short_name: 'GF',
    name: 'Geo Face',
    description: 'Geo Face App',
    start_url: '/',
    display: 'standalone',
    background_color: '#abf0e9',
    theme_color: '#abf0e9',
    orientation: 'portrait-primary',
    icons: [
      {
        src: path.resolve('src/img/icons/manifest-icon-512.png'),
        sizes: [96, 128, 180, 192, 256, 512],
        type: 'image/png',
        purpose: 'maskable any'
      }
    ]
  })
],

```

Fonte: Elaborado pela Autora.

### 3.2.9 Babel

Babel é um compilador *JavaScript*. Ele é usado especialmente para converter o código *JavaScript*, a partir do lançamento do *ECMAScript* 2015+, para códigos compatíveis com versões mais antigas em navegadores ou ambientes atuais ou mais antigos. Além disso, pode-se utilizar uma sintaxe moderna para escrever o código, sem esperar o suporte necessário do navegador do usuário, com recursos de *polyfill* para linguagens modernas serem suportadas em navegadores antigos, ocasionando uma transformação de sintaxe e transformações de código (BABEL, 2021).

Para a configuração do compilador Babel no presente trabalho, foi necessário a criação de um arquivo `.babelrc` em que são implementadas as configurações mostradas na Figura 35.

Figura 35 - Implementação do compilador babel no arquivo de configuração `.babelrc`.

```
{
  "presets": [
    /*
     * Permite usar código JavaScript mais recente sem a necessidade
     * de microgerenciar quais transformações de sintaxe/polyfills
     * são necessárias para o ambiente de destino.
     */
    ["@babel/preset-env", {
      "targets": {
        /*
         * Suporta as últimas 5 versões de cada navegador e certifica de que
         * todas as versões do IE (a partir da 8) sejam compatíveis.
         */
        "browsers": [
          "last 5 versions",
          "ie >= 8"
        ]
      }
    }]
  ]
}
```

Fonte: Elaborado pela Autora.

### 3.2.10 npm

A organização npm é um repositório com vários registros de software que utiliza a linguagem *JavaScript*, onde é possível compartilhar e adquirir pacotes emprestados de código para serem usados nas aplicações, sendo um gerenciador de pacotes para a plataforma *Node JavaScript* e consiste em 3 pontos principais, que são:

- O site permite conhecer e gerenciar os pacotes;
- É executada através do terminal como *command-line interface* (CLI);
- O registro é um grande banco de dados público de software *JavaScript* (NPM, 2021).

O gerenciador de pacotes tem diversas funcionalidades, mas para o presente trabalho será usado para incorporar os pacotes disponibilizados no site dentro da aplicação. Para isso, é necessário um arquivo chamado `package.json`, usado para gerenciar os pacotes e a instalação futuramente em outros ambientes (NPM, 2021). O arquivo `package.json` lista todos os pacotes no qual a aplicação é dependente e especifica as versões de cada pacote utilizado (NPM, 2021).

Na Figura 36, apresenta-se o arquivo `package.json` utilizado na aplicação contendo todos os pacotes utilizados e as determinadas configurações iniciais como nome do autor, descrição, versão, scripts que são utilizados para rodar a aplicação tanto local quanto em



produção, arquivo principal, licença, *devDependencies* que são pacotes para o desenvolvimento e teste local, *dependencies* pacotes que são utilizados em produção (NPM, 2021).

Figura 36 - Arquivo *package.json*.

```
{
  "name": "tcc",
  "version": "1.0.0",
  "description": "Trabalho de Conclusão de Curso II",
  "main": "index.js",
  "scripts": {
    "dev": "webpack --mode development",
    "build": "webpack --mode production",
    "start": "webpack-dev-server --mode development --open"
  },
  "author": "Isabel",
  "license": "ISC",
  "devDependencies": {
    "@babel/core": "^7.10.5",
    "@babel/preset-env": "^7.10.4",
    "babel-core": "^6.26.3",
    "babel-loader": "^8.1.0",
    "babel-preset-env": "^1.7.0",
    "css-loader": "^3.6.0",
    "html-webpack-plugin": "^4.3.0",
    "mini-css-extract-plugin": "^0.9.0",
    "node-sass": "^4.14.1",
    "postcss-load-config": "^2.1.0",
    "postcss-preset-env": "^6.7.0",
    "resolve-url-loader": "^3.1.1",
    "sass-loader": "^9.0.2",
    "style-loader": "^1.2.1",
    "webpack": "^4.43.0",
    "webpack-cli": "^3.3.12",
    "webpack-dev-server": "^3.11.0",
    "webpack-pwa-manifest": "^4.3.0",
    "workbox-webpack-plugin": "^6.1.2"
  },
  "dependencies": {
    "@babel/polyfill": "^7.10.4",
    "azure-maps-control": "^2.0.31",
    "azure-maps-drawing-tools": "^0.1.6",
    "file-loader": "^6.0.0",
    "firebase": "^8.2.9"
  }
}
```

Fonte: Elaborado pela Autora.

Para a instalação dos pacotes e suas dependências é necessário rodar o comando *npm install* no terminal, dentro da pasta do projeto, isso criará uma pasta chamada *node\_modules* na raiz do projeto com todos os pacotes e suas dependências que serão utilizadas na aplicação.

### 3.2.11 Azure Mapas

O Azure Mapas é um conjunto de serviços geoespaciais e *Software Development Kit* (SDK) que usa dados para fornecer um cenário geográfico que pode ser usado em aplicações web e mobile (ANASTASIA HARRIS, 2020).

Para o presente trabalho, foi necessário a criação de uma conta dos Mapas do Azure que será criada no portal do Azure, para utilizar os serviços do Azure Maps. Na Figura 37 representa-se a criação da conta.

Figura 37 - Criação da conta dos Mapas do Azure.

The screenshot shows the Azure portal interface for creating an Azure Maps account. The browser address bar shows 'https://portal.azure.com'. The page title is 'Criar Azure Mapas'. The left sidebar shows the 'Criar um recurso' button highlighted. The main content area is titled 'Criar Conta do Azure Mapas' and contains the following details:

- DETALHES DO PROJETO**
  - Subscription: Azure para Estudantes ✓
  - Grupo de recursos: TCCII ✓
- DETALHES DA CONTA**
  - Nome: geo-face ✓
  - Tipo de preço: Standard S1 ✓

At the bottom, there is a checkbox for 'Confirmo que li e concordo com a Política de Privacidade e de Licenças.' and a 'Criar' button.

Fonte: Adaptada pela Autora.

Após a criação da conta, é disponibilizado uma chave de assinatura para os serviços serem utilizados na aplicação. Na aplicação, foi utilizado o serviço disponibilizado de configurar um limite geográfico usando o Azure Maps, para validar a localização do usuário em tempo real e demarcar a cerca virtual do local desejado do mapa disponibilizado pelo Azure Maps. Após a demarcação da área, será salvo os dados longitude, latitude e raio no banco de dados *real time* do Google Firebase, em seguida o usuário terá uma mensagem de feedback se o a área foi salva com sucesso ou não, como é mostrado na Figura 38.





O ângulo central de Haversine pode ser calculado utilizando-se a equação 1 onde o raio da terra sendo  $r$  e o  $d$  como sendo a distância entre dois pontos na latitude, sendo eles denominados de  $\phi_1, \phi_2$ . Distância entre dois pontos  $\lambda_1, \lambda_2$  sendo a longitude (UPADHYAY, 2020).

$$\text{haversine} \left( \frac{d}{r} \right) = \text{haversine}(\phi_2 - \phi_1) + \cos(\phi_1) \cos(\phi_2) \text{haversine}(\lambda_2 - \lambda_1) \quad (1)$$

Ao iniciar a aplicação, será solicitado para permitir o acesso a localização do dispositivo do usuário, caso seja permitido o acesso, a posição atual do dispositivo será capturada por meio do método disponibilizado pela API da Web *Geolocation* em conjunto com Azure Mapas.

### 3.2.12 Progressive Web App

O software para a aplicação do presente trabalho, foi desenvolvido para ser um PWA, com isso pode ser usado em desktop, mobile e web. Para o desenvolvimento do PWA em conjunto com *webpack*, foram utilizados os plugins *WorkboxWebpackPlugin* e *WebpackPwaManifest*.

O *Workbox* é uma biblioteca mantida pela Google que fornece um conjunto de boas práticas recomendadas no desenvolvimento para o *Service Worker* (WORKBOX, 2021), além disso, fornece o plugin *WorkboxWebpackPlugin*, para ser usado no arquivo de configuração do *webpack* na matriz de plugins, para ser utilizado o método *GenerateSW* que tem como função criar um arquivo *sw.js* na aplicação pronto para uso como é apresentado na Figura 41.

Figura 41 – Plugin *WorkboxWebpackPlugin*.

```

/**
 * O plug-in criará um arquivo de service worker chamado sw
 * e o adicionará ao pipeline de ativos do webpack.
 */
new WorkboxPlugin.GenerateSW({
  // O nome do arquivo que será criado.
  swDest: 'sw.js',
  // Controla clientes existentes assim que for ativado.
  clientsClaim: true,
  // Força o SW em espera e torna o mesmo ativo.
  skipWaiting: true,
  // Determina o tamanho máximo dos arquivos que serão pré-armazenados.
  maximumFileSizeToCacheInBytes: 5000000,
  // Define as regras de armazenamento em cache do tempo de execução.
  runtimeCaching: [
    // Corresponde a qualquer solicitação que tem fonts.googleapis.
    {
      urlPattern: new RegExp("https://fonts.googleapis.com/"),
      /**
       * Aplica uma estratégia de cache em paralelo com a rede.
       * A estratégia responderá com a versão em cache, se disponível,
       * caso contrário, aguardará a resposta da rede.
       */
      handler: 'StaleWhileRevalidate',
      options: {
        // Nome do cache personalizado.
        cacheName: 'google-fonts',
        // Código de status para as respostas que podem ser considerados armazenáveis em cache.
        cacheableResponse: {
          statuses: [0, 200],
        }
      }
    }
  ]
})

```

Fonte: Elaborado pela Autora.

O *WebpackPwaManifest* é o plugin *webpack* que gera o arquivo *manifest.json* para a aplicação PWA, com isso, pode-se definir o que irá aparecer para que a aplicação se apresente como um aplicativo nativo, são elas nome, ícone, cor do fundo, cor do tema, orientação para telas dos smartphones sendo apresentada na Figura 42.

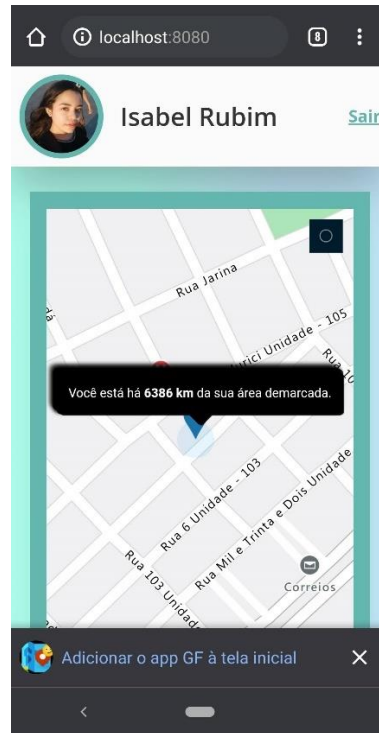
Figura 42 - Plugin *WebpackPwaManifest*.

```
/**
 * Plugin que gera um 'manifest.json' para o Progressive Web Application,
 * com redimensionamento automático de ícones.
 */
new WebpackPwaManifestPlugin({
  // Nome abreviado para o aplicativo
  short_name: 'GF',
  // Nome do aplicativo
  name: 'Geo Face',
  // Descrição do aplicativo
  description: 'Geo Face App',
  start_url: '/',
  // Modo de exibição, terá aparência de um aplicativo independente.
  display: 'standalone',
  // Cor de fundo
  background_color: '#abf0e9',
  // Cor do tema
  theme_color: '#abf0e9',
  // Modo retrato principal, posição vertical com os botões na parte inferior.
  orientation: 'portrait-primary',
  // Ícone com redimensionamento automático.
  icons: [
    {
      src: path.resolve('src/img/icons/icon-tcc.png'),
      sizes: [96, 128, 180, 192, 256, 512],
      type: 'image/png',
      purpose: 'maskable any'
    }
  ]
}),
```

Fonte: Elaborado pela Autora.

Para o usuário ter o aplicativo disponibilizado em seu smartphone, primeiro ele deve acessar a URL do aplicativo via navegador, em seguida irá apresentar a opção de download na parte de inferior da tela como pode ser visto na Figura 43.

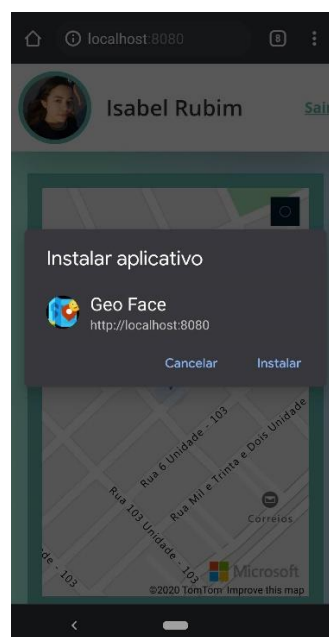
Figura 43 - Tela para adicionar *Geo Face* a tela inicial do smartphone.



Fonte: Elaborado pela Autora.

Ao clicar na opção, será apresentado o pop-up confirmando a instalação do aplicativo sendo apresentado na Figura 44.

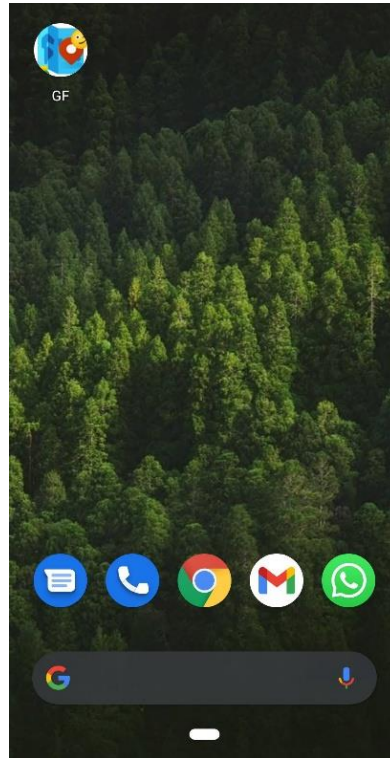
Figura 44 - Tela para a confirmação da instalação do aplicativo.



Fonte: Elaborado pela Autora.

Ao confirmar sim para a instalação, o aplicativo será adicionado na tela inicial do smartphone com seu respectivo ícone como é apresentado na Figura 45.

Figura 45 - Tela com o aplicativo adicionado a tela inicial do smartphone.



Fonte: Elaborado pela Autora.

Após instalado, o usuário pode usar a aplicação como um aplicativo, ao clicar no ícone será aberto o aplicativo e mostrará a tela apresentada na Figura 46.



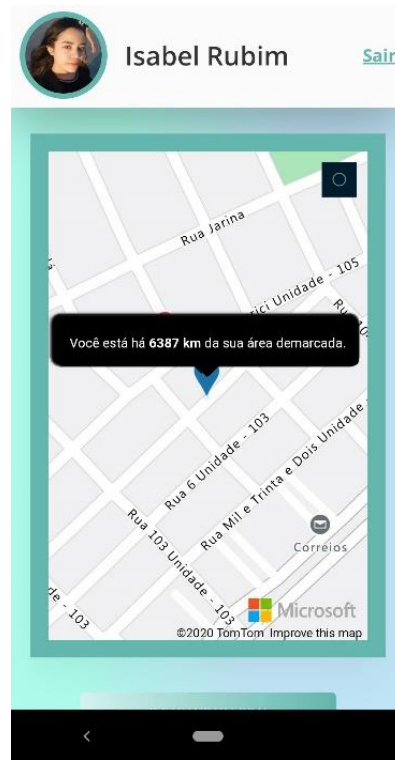
Figura 46 - Tela inicial ao clicar no aplicativo.



Fonte: Elaborado pela Autora.

Por fim, o usuário terá sua localização em tempo real e acesso a câmera do seu smartphone para o reconhecimento facial, além disso, a demarcação da área escolhida como mostra na Figura 47.

Figura 47 - Aplicativo aberto no smartphone.



Fonte: Elaborado pela Autora.

## 4 CONSIDERAÇÕES FINAIS E CONCLUSÃO

O presente trabalho teve como objetivo a integração de uma aplicação híbrida utilizando um microcontrolador com a finalidade de tomar ações baseadas no reconhecimento facial e na localização georreferenciada do usuário. Para o desenvolvimento da integração entre a aplicação híbrida e o microcontrolador foram utilizadas ferramentas do Google, com a plataforma Firebase, para possibilitar a comunicação em tempo real entre os sistemas, desse modo com a API de geolocalização da Microsoft Azure e navegador web em conjunto com a biblioteca utilizada no microcontrolador ESP32, para a comunicação direta com o banco de dados *realtime* do Firebase por meio da rede WiFi a qual se mostrou eficiente no funcionamento.

As informações disponibilizadas no banco de dados *realtime*, Mapas e Detecção Facial do Azure disponibilizaram confiabilidade, usabilidade, estruturabilidade e portabilidade para a comunicação entre os sistemas instalados nas diferentes plataformas. A biblioteca utilizada para ler e sobrescrever os dados do banco de dados *realtime* para o desenvolvimento do sistema no microcontrolador, mostrou-se eficaz para realizar a comunicação entre esses dispositivos, permitindo a leitura, gravação e remoção dos dados.

O Visual Studio Code forneceu toda a estrutura e ferramentas necessárias para o desenvolvimento da aplicação híbrida tornando a criação do software híbrido mais produtivo para navegadores e dispositivos móveis assim como nos desktops, podendo ser utilizado em várias plataformas que necessitam utilizar reconhecimento facial e geolocalização, o que se provou bastante útil em aplicações semelhantes.

Foram executados testes para validar e verificar a execução da aplicação e esta se apresentou com um resultado eficiente respondendo as questões de pesquisa, sendo possível o usuário destrancar um local por meio do georreferenciamento e reconhecimento facial obtido em qualquer dispositivo, utilizando banco de dados em *realtime* do Google Firebase para armazenamento do status e comunicação do microcontrolador.

O presente trabalho possibilitou vários conhecimentos na área de integração e comunicação em aplicações de hardware ou software com a utilização de APIs, automação em nuvem, aplicações híbridas, biométricas e georreferenciadas.

O propósito do projeto mostrou-se promissor para diversas aplicações onde a localização e o reconhecimento facial do usuário são um dado importante, sendo levado em consideração que a biometria é um passo importante para os sistemas de segurança e sistemas

de geolocalização está disponível em diversos dispositivos, dado que já contém essa tecnologia e uma câmera, faz-se possível o uso da aplicação e com baixo custo utilizar o reconhecimento facial e a geolocalização coletada por um dispositivo do usuário para executar ações em sistemas microcontrolados.

Para sugestões futuras todos os programas desenvolvidos encontram se listados no apêndice A.

## 5 REFERÊNCIAS

ALVES, Paulo Victor Alexandre. **Verificação automática georreferenciada**. 2018. 77 f. TCC (Graduação) - Curso de Engenharia de Computação, Escola de Ciências Exatas e da Computação, Pontifícia Universidade Católica de Goiás, Goiânia, 2018.

ANASTASIA HARRIS. Microsoft. **O que é o Azure Mapas?** 2020. Disponível em: <https://docs.microsoft.com/pt-br/azure/azure-maps/about-azure-maps>. Acesso em: 16 abr. 2021.

BABEL. **What is Babel?** Disponível em: <https://babeljs.io/docs/en/>. Acesso em: 14 abr. 2021.

COLABORADORES DA MOZILLA. **O que é JavaScript?** Disponível em: [https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First\\_steps/O\\_que\\_e\\_JavaScript](https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/O_que_e_JavaScript). Acesso em: 15 set. 2020.

COPEES, Flavio. **The Complete JavaScript Handbook**. 2018. Disponível em: <https://medium.com/free-code-camp/the-complete-javascript-handbook-f26b2c71719c>. Acesso em: 15 set. 2020.

COSTA, Vambaster José da. **Reconhecimento de padrões faciais: uma síntese**. 2019. 97 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná, Santa Helena, 2019. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/11953>. Acesso em: 25 set. 2020.

DECHALERT, Aphinya. **We need to talk about Firebase: where have you been all this time?. Where have you been all this time?.** 2019. Disponível em: <https://medium.com/madhash/we-need-to-talk-about-firebase-1ffeal180d75>. Acesso em: 18 out. 2020.

FAGERTUN, Jens. **Face Recognition**. 2005. 209 f. Dissertação (Mestrado) - Curso de Informatics And Mathematical Modelling, Technical University Of Denmark, Kongens

Lyngby, 2005. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/versions?doi=10.1.1.452.6538>. Acesso em: 27 set. 2020.

FIREBASE (org.). **Estruturar o banco de dados**. 2020. Disponível em: <https://firebase.google.com/docs/database/ios/structure-data>. Acesso em: 18 out. 2020.

FIREBASE (org.). **Firestore Realtime Database**. 2020. Disponível em: <https://firebase.google.com/docs/database/?hl=pt-br>. Acesso em: 18 out. 2020.

FIREBASE (org.). **O Firebase ajuda as equipes de aplicativos para dispositivos móveis e da Web a alcançar o sucesso**. Disponível em: [https://firebase.google.com/?gclid=Cj0KCQiAnb79BRDgARIsAOVbhRo-RaUapMDBdiyduERcMBFTSHC3UV-zN9aXSK-b09dwnSYKIHwGwR4aAg3CEALw\\_wcB](https://firebase.google.com/?gclid=Cj0KCQiAnb79BRDgARIsAOVbhRo-RaUapMDBdiyduERcMBFTSHC3UV-zN9aXSK-b09dwnSYKIHwGwR4aAg3CEALw_wcB). Acesso em: 14 nov. 2020.

GALTON, Francis. **Personal identification and description**. Nature, p. 173-201. jun. 1888. Disponível em: [http://galton.org/bib/JournalItem.aspx\\_action=view\\_id=168](http://galton.org/bib/JournalItem.aspx_action=view_id=168). Acesso em: 28 set. 2020.

GOOGLE DEVELOPER GROUPS (org.). **Provide contextual experiences when users enter or leave an area of interest**. Elaborado por Google Developer Groups. Disponível em: <https://developers.google.com/location-context/geofencing>. Acesso em: 12 out. 2020.

MAGALHÃES, Paulo Sérgio; SANTOS, Henrique Dinis. Biometria e autenticação. In: CONFERÊNCIA DA ASSOCIAÇÃO PORTUGUESA DE SISTEMAS DE INFORMAÇÃO, 4., 2003, Porto. **Anais [...]**. Porto: Apsi, 2003. p. 1-17. Disponível em: <http://hdl.handle.net/1822/2184>. Acesso em: 26 set. 2020.

MAGEE, Tamlin. **Geofencing: what is it?** 2020. Disponível em: <https://www.computerworld.com/article/3528795/geofencing-what-is-it.html>. Acesso em: 11 out. 2020.

MATOS AQUINO, Lygia et al. Proposta de um curso semipresencial de robótica educacional utilizando a plataforma Arduino. **Revista Principia - Divulgação Científica e Tecnológica do IFPB**, [S.I.], n. 34, p. 48-54, jun. 2017. ISSN 2447-9187. Disponível em: <https://periodicos.ifpb.edu.br/index.php/principia/article/view/1327>. Acesso em: 13 nov. 2020. doi: <http://dx.doi.org/10.18265/1517-03062015v1n34p48-54>.

MERCIER, Rémi. **What is an API?** 2019. Disponível em: [https://dev.to/mercier\\_remi/what-is-an-api-4ao9](https://dev.to/mercier_remi/what-is-an-api-4ao9). Acesso em: 19 out. 2020.

MICROSOFT AZURE (org.). **Azure portal overview**. 2019. Disponível em: <https://docs.microsoft.com/en-us/azure/azure-portal/azure-portal-overview>. Acesso em: 06 nov. 2020.

MICROSOFT AZURE (org.). **Conceitos de reconhecimento facial**. 2019. Disponível em: <https://docs.microsoft.com/pt-br/azure/cognitive-services/face/concepts/face-recognition>. Acesso em: 14 nov. 2020.

MICROSOFT AZURE (org.). **Detecção de face e atributos**. 2019. Disponível em: <https://docs.microsoft.com/pt-br/azure/cognitive-services/face/concepts/face-detection>. Acesso em: 14 nov. 2020.

MICROSOFT AZURE (org.). **O que é o serviço de Detecção Facial do Azure?** 2020. Disponível em: <https://docs.microsoft.com/pt-br/azure/cognitive-services/face/overview>. Acesso em: 14 nov. 2020.

MICROSOFT AZURE (org.). **O que são os Serviços Cognitivos do Azure?** 2020. Disponível em: <https://docs.microsoft.com/pt-br/azure/cognitive-services/what-are-cognitive-services>. Acesso em: 11 nov. 2020.

NAMIOT, Dmitry; SNEPS-SNEPPE, Manfred. Geofence and Network Proximity. **Internet Of Things, Smart Spaces, And Next Generation Networking**, [S.I.], v. 8121, p. 117-127, 2013.

Springer Berlin Heidelberg. [http://dx.doi.org/10.1007/978-3-642-40316-3\\_11](http://dx.doi.org/10.1007/978-3-642-40316-3_11). Disponível em: <https://arxiv.org/abs/1303.5943>. Acesso em: 10 out. 2020.

NASCIMENTO, Felipe Santos do. **Aprenda a configurar a rede WiFi do ESP32 pelo smartphone**. 2020. Disponível em: <https://www.filipeflop.com/blog/aprenda-a-configurar-a-rede-wifi-do-esp32-pelo-smartphone/>. Acesso em: 12 nov. 2020.

NPM. **About npm**. Disponível em: <https://docs.npmjs.com/about-npm>. Acesso em: 15 abr. 2021.

NPM. **Creating a package.json file**. Disponível em: <https://docs.npmjs.com/creating-a-package-json-file>. Acesso em: 15 abr. 2021.

NPM. **Specifying dependencies and devDependencies in a package.json file**. Disponível em: <https://docs.npmjs.com/specifying-dependencies-and-devdependencies-in-a-package-json-file>. Acesso em: 15 abr. 2021.

PERIM, Victor Gonçalves de Carvalho Feitosa; NASCIMENTO, Jefferson Nataline Rosa do. **Microcontroladores e Microprocessadores**. Londrina: Editora e Distribuidora Educacional S.A., 2017. 232 p. Disponível em: <https://www.passeidireto.com/arquivo/47359827/livro-unico-microcontrolador>. Acesso em: 04 out. 2020.

PRADO, Kelvin Salton do. **Comparação de técnicas de reconhecimento facial para identificação de presença em um ambiente real e semicontrolado**. 2018. 171 f. Tese (Doutorado) - Curso de Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2018. Disponível em: <https://www.teses.usp.br/teses/disponiveis/100/100131/tde-07012018-222531/pt-br.php>. Acesso em: 26 set. 2020.

RAPIDAPI. **API – What is an API?** Elaborado por RapidAPI. Disponível em: <https://rapidapi.com/blog/api-glossary/api/>. Acesso em: 02 nov. 2020.



REBOUÇAS FILHO, Pedro Pedrosa. **Microcontroladores PIC: linguagem C utilizando CCS para leigos**. Maracanaú: Instituto Federal de Educação, Ciência e Tecnologia do Ceará, 2014. 208 p. Disponível em: [https://www.researchgate.net/publication/262286119\\_Microcontroladores\\_PIC\\_Linguagem\\_C\\_utilizando\\_CCS\\_para\\_leigos](https://www.researchgate.net/publication/262286119_Microcontroladores_PIC_Linguagem_C_utilizando_CCS_para_leigos). Acesso em: 04 out. 2020.

REDAÇÃO. Chief Information Officer (ed.). **3 coisas que você precisa saber sobre geofencing**. 2018. Elaborado pela redação da CIO/EUA. Disponível em: <https://cio.com.br/tendencias/3-coisas-que-voce-precisa-saber-sobre-geofencing/>. Acesso em: 11 out. 2020.

ROUSE, Margaret. **Microsoft Azure**. Disponível em: <https://searchcloudcomputing.techtarget.com/definition/Windows-Azure>. Acesso em: 02 nov. 2020.

SANTOS, B. et al. **Internet das Coisas: da Teoria à Prática**. 1. ed. [S.l.: s.n.], 2016. p. 1-50. SCHAEFER, Lauren. **What is NoSQL?** Coordenada por mongoDB. Disponível em: <https://www.mongodb.com/nosql-explained>. Acesso em: 18 out. 2020.

SILVA, Daniela Rocha; SOBRAL, Luis Felipe. **Um Estudo em Larga Escala sobre a Estrutura do Código-fonte de Pacotes JavaScript**. Orientador: Márcio de Oliveira Barros. 2017. 58 p. Trabalho de conclusão de curso (Bacharelado em Sistema de Informação) - Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2017. Disponível em: <https://bsi.uniriotec.br/wp-content/uploads/sites/31/2020/05/201707DanielaRochaLuisSobral.pdf>. Acesso em: 13 set. 2020.

THALES GROUP (org.). **Biometrics: definition, trends, use cases, laws and latest news**. 2020. Disponível em: <https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/inspired/biometrics>. Acesso em: 26 set. 2020.

THELARKLNN. **Concepts**. Disponível em: <https://v4.webpack.js.org/concepts/modules/>. Acesso em: 12 abr. 2021.

THELARKINN. **Modules**. Disponível em: <https://v4.webpack.js.org/concepts/modules/>. Acesso em: 12 abr. 2021.

TRINDADE, Patrícia Esteves; AFFINI, Letícia Passos. APONTAMENTO A CERCA DO PROGRESSIVE WEB APPS. **Revista GEMInIS: GRUPO DE ESTUDOS SOBRE MÍDIAS INTERATIVAS EM IMAGEM E SOM**, São Carlo – SP, v. 9, n. 3, p. 1-175, abr./2019. Disponível em: <https://doity.com.br/media/doity/submissoes/artigo-6d73074b34b7631db712c0472463b10b08ed8ea4-arquivo.pdf>. Acesso em: 11 set. 2020.

UPADHYAY, Akshay. **Haversine Formula – Calculate geographic distance on earth**. Disponível em: <https://www.igismap.com/haversine-formula-calculate-geographic-distance-earth/>. Acesso em: 22 abr. 2020.

VISUAL STUDIO CODE (org.). **Getting Started**. Disponível em: <https://code.visualstudio.com/docs#vscode-in-action>. Acesso em: 12 nov. 2020.

WORKBOX. **Why Workbox?** Disponível em: <https://developers.google.com/web/tools/workbox>. Acesso em: 10 maio 2021.

WU, Dosanna. **What is Geofencing?** 2019. Disponível em: <https://developer.tomtom.com/blog/decoded/what-geofencing>. Acesso em: 11 out. 2020.

ZIN, Mohd Shahril Izuan Mohd *et al.* Geofencing-based Auto-Silent Mode Application. **Telecommunication, Electronic And Computer Engineering**. [S.I.], p. 199-204. 2016. Disponível em: <https://journal.utem.edu.my/index.php/jtec/article/view/1398>. Acesso em: 10 out. 2020.

## APÊNDICE A

```

// Models:
import firebase from 'firebase/app';
import 'firebase/firestore';
import 'firebase/auth';

export default class Auth {
  constructor() {
    // Referencias para o banco de dados e para os métodos disponibilizados para o auth
    this.db = firebase.firestore();
    this.auth = firebase.auth();

    // Pega a coleção users criada no Cloud Firestore
    this.users = this.db.collection('users');

    // Disponibiliza os dados em cache.
    firebase.firestore().enablePersistence();
  }

  /**
   * @param {string} email - emailexample@dominio.com
   * @param {string} password - *****
   * @param {string} customData - photoURL, name, uid
   */
  async signUp(email, password, customData) {
    try {
      // Cadastro do usuário com email, senha e dados customizados
      const credential = await this.auth.createUserWithEmailAndPassword(email, password);
      if (credential.user) {
        this.setUserData(credential.user, customData);
      }
    }
  }
}

```

```

    }
  } catch (error) {
    console.log('error to credential', error);
  }
}

/**
 * @param {string} email
 * @param {string} password
 * @returns email, name, photoURL, uid.
 */
async signIn(email, password) {
  try {
    // Faz o login do usuário
    const credential = await this.auth.signInWithEmailAndPassword(email, password);
    if (credential.user) {
      this.user = await this.getUser(credential.user);
      return this.user;
    }
  } catch (error) {
    console.log('error in signIn', error);
  }
}

// Obtém as propriedades do usuário salva no banco de dados cloud firestore
async getUser({ uid }) {
  const user = await this.users.doc(uid).get();
  if (user.exists) {
    return user.data();
  }
}
}

```

```

// Desloga o usuário
async signOut() {
  try {
    await this.auth.signOut();
  } catch (error) {
    console.log('error in logout')
  }
}

//
setUserData(user, customData) {
  // Adiciona propriedades do usuário no banco de dados cloud firestore
  const userRef = this.users.doc(user.uid);
  const data = {
    uid: user.uid,
    email: user.email,
    name: customData.name,
    photoURL: customData.photoURL
  };
  userRef.set(data, { merge: true });
}
}

import { detectUri, verifyUri, subscriptionKeyFace } from '../config';

export default class Face {
  /**
   * @param {string} url - URL salva no banco de dados Firebase
   */
  async detectFaceUrl(url) {

```

```
// Faz a detecção do rosto presente na foto salva
try {

  /**
   * Chamada na Face API com o endpoint para detecção dos rostos nas fotos,
   * com o cabeçalho na chamada para representação do dado do tipo JSON,
   * recebendo a URL.
   */
  const faceDetect = await fetch(detectUri, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Ocp-Apim-Subscription-Key': subscriptionKeyFace,
    },
    body: JSON.stringify({ url }),
  });

  const data = await faceDetect.json();

  if (data) {
    this.faceIdURL = data[0].faceId;
  }

} catch (error) {
  console.log('error in detect URL', error);
}

/**
 * @param {Blob} blob - objeto do tipo arquivo, com dados brutos imutáveis.
 */
```

```

async detectFace(blob) {
  // Faz a detecção do rosto presente na foto tirada

  try {
    /**
     * Chamada na Face API com o endpoint para detecção dos rostos nas fotos,
     * com o cabeçalho na chamada para representação do dado tipo binário,
     * recebendo o blob.
     */
    const faceDetect = await fetch(detectUri, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/octet-stream',
        'Ocp-Apim-Subscription-Key': subscriptionKeyFace,
      },
      body: blob,
    });

    const data = await faceDetect.json();

    if (data) {
      this.faceId = data[0].faceId;
    }

  } catch (error) {
    console.log('error in detect', error);
  }
}

async identifyFace() {
  // Faz a identificação do rosto presente na foto tirada e salva

```

```

try {
  const paramsFaceIds = {
    faceId1: this.faceId,
    faceId2: this.faceIdURL,
  };

  /**
   * Chamada na Face API com o endpoint para verificação das fotos,
   * com o cabeçalho na chamada para representação do dado tipo JSON,
   * recebendo os Ids da detecção facial.
   */
  const faceVerify = await fetch(verifyUri, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Ocp-Apim-Subscription-Key': subscriptionKeyFace,
    },
    body: JSON.stringify(paramsFaceIds)
  });

  // Resultado se as fotos são correspondentes representado no valor isIdentical.
  this.faceVerifyResult = await faceVerify.json();
} catch (error) {
  console.log('error in indentify face', error)
}

/**
 * @param {string} dataURL - URL da foto tirada
 * @returns arquivo do tipo Blob
 */

```



```

async makeBlob(dataURL) {
  try {
    // Transforma a URL da foto tirada em um tipo Blob.
    const response = await fetch(dataURL);
    const responseBlob = await response.blob();

    return responseBlob;
  } catch (error) {
    console.log('error to make blob', error);
  }
}

import * as atlas from 'azure-maps-control';

import { subscriptionKeyMaps, clientID } from '../config';

export default class Maps {

  getMap() {
    try {
      this.map = new atlas.Map('myMap', {
        view: 'Auto',
        authOptions: {
          authType: 'subscriptionKey',
          subscriptionKey: subscriptionKeyMaps,
          clientId: clientID,
          getToken: function (resolve, reject, map) {
            //URL to your authentication service that retrieves an Azure Active Directory Token.
            const tokenServiceUrl =
              "https://azuremapscodesamples.azurewebsites.net/Common/TokenService.ashx";

```

```

        fetch(tokenServiceUrl).then(r => r.text()).then(token => resolve(token));
    }
}
});

this.getUserPositionAccuracyCircle();
} catch (error) {
    console.log('error in get map', error);
}
}

getUserPositionAccuracyCircle() {
    this.map.events.add('ready', () => {
        //Request the user's location
        navigator.geolocation.getCurrentPosition(position => {
            //Create a data source and add it to the map.
            this.dataSource = new atlas.source.DataSource();
            this.map.sources.add(this.dataSource);

            //Create a circle from a Point feature by providing it a subType property set to "Circle"
            and radius property.
            this.userPosition = [position.coords.longitude, position.coords.latitude];
            this.userPoint = new atlas.data.Point(this.userPosition);

            //Add a point feature with Circle properties to the data source for the users position. This
            will be rendered as a polygon.
            this.dataSource.add(new atlas.data.Feature(this.userPoint, {
                subType: 'Circle',
                radius: position.coords.accuracy
            }));
        });
    });
}

```

```

//Add the users position point.
this.dataSource.add(this.userPoint);

this.map.layers.add([
  //Create a polygon layer to render the filled in area of the accuracy circle for the users
  position.
  new atlas.layer.PolygonLayer(this.dataSource, null, {
    fillColor: 'rgba(0, 153, 255, 0.2)'
  }),
  //Create a symbol layer to render the users position on the map.
  new atlas.layer.SymbolLayer(this.dataSource, null, {
    filter: ['any', ['==', ['geometry-type'], 'Point'], ['==', ['geometry-type'], 'MultiPoint']]
  })
]);
//Only render Point or MultiPoints in this layer.

});
//Center the map on the users position.
this.map.setCamera({
  center: this.userPosition,
  zoom: 16
});
});
});
}
}

import * as atlas from 'azure-maps-drawing-tools';
import * as atlasControl from 'azure-maps-control';

import { calculateDistance } from '../utils/index';

```

```

let drawingManager;
export default class MapsDrawing {
  constructor(map) {
    this.map = map;
  }

  setMeasuringToolCircle() {
    try {
      this.map.events.add('ready', () => {
        drawingManager = new atlas.drawing.DrawingManager(this.map, {
          toolbar: new atlas.control.DrawingToolbar({
            buttons: ['draw-circle'],
            position: 'top-right',
            style: 'dark'
          })
        });

        this.map.events.add('drawingmodechanged', drawingManager,
this.drawingModeChanged);

        this.map.events.add('drawingcomplete', drawingManager, this.drawingComplete);
      });
    } catch (error) {
      console.log('error in set draw', error);
    }
  }

  drawingModeChanged(mode) {
    //Clear the drawing canvas when the user enters into a drawing mode.
    if (mode.startsWith('draw')) {
      drawingManager.getSource().clear();
    }
  }
}

```

```

    }
}

drawingComplete() {
    //Exit drawing mode.
    drawingManager.setOptions({ mode: 'idle' });
    window.dataSource = drawingManager.getSource();
    document.querySelector('.btn-cta--mark-area').classList.remove('is-hidden');
}

getCircleToMap({ latitude, longitude, radius }) {
    try {
        this.map.events.add('ready', () => {
            // Create a data source and add it to the map

            this.dataSource = new atlasControl.source.DataSource();
            this.map.sources.add(this.dataSource);

            this.startTrackingUserPostion(latitude, longitude, radius);

            // Create a circle
            this.dataSource.add(
                new atlasControl.data.Feature(
                    new atlasControl.data.Point([longitude, latitude]), {
                        subType: 'Circle',
                        radius
                    }
                )
            );
        });
    }
}

```

// Create a polygon layer to render the filled in area of the circle polygon, and add it to the map.

```

this.map.layers.add(new atlasControl.layer.PolygonLayer(this.dataSource, null, {
  fillColor: 'rgba(250, 30, 14, 0.8)'
}));
});
} catch (error) {
  console.log('error in get circle map', error);
}
}

```

startTrackingUserPosition(geoFenceLatitude, geoFenceLongitude, radius) {

// Watch the users position.

```

this.map.events.add('ready', () => {
  navigator.geolocation.watchPosition(geoPosition => {
    let userPosition = [geoPosition.coords.longitude, geoPosition.coords.latitude];
    let userShape = new atlasControl.Shape(new atlasControl.data.Feature(new
atlasControl.data.Point(userPosition), geoPosition));

```

```

this.dataSource.add(userShape);

```

```

let distanceBetweenTwoPoints = calculateDistance(geoFenceLatitude,
geoFenceLongitude, userPosition[1], userPosition[0]);

```

```

this.showPopup(distanceBetweenTwoPoints, userPosition, radius);

```

//Create a symbol layer to render the users position on the map.

```

new atlasControl.layer.SymbolLayer(this.dataSource, null, {
  filter: ['any', ['==', ['geometry-type'], 'Point'], ['==', ['geometry-type'], 'MultiPoint']]
//Only render Point or MultiPoints in this layer.
})

```

```

    this.map.setCamera({
      center: userPosition,
      zoom: 16
    });
  }, (error) => {
    this.geoError(error.code);
  }, {
    enableHighAccuracy: true, // Pede para o browser não economizar bateria, trazendo a
informação mais precisa de geolocalização
    timeout: 10000, // Até quando você vai esperar pra ter esses dados, 10s
    maximumAge: 3000 // Uma chamada a cada 3s
  });
});
}

showPopup(distanceBetweenTwoPoints, userPosition, radius) {
  const km = 1000;

  if (distanceBetweenTwoPoints <= radius) {
    this.message = `Você está dentro da área demarcada.`;
  } else if (distanceBetweenTwoPoints < km) {
    this.message = `Você está perto da sua área demarcada: <b>${distanceBetweenTwoPoints}
m</b>`;
  } else {
    this.message = `Você está há <b>${distanceBetweenTwoPoints} km</b> da sua área
demarcada.`;
  }

  let popupNotification = new atlasControl.Popup({
    content: `

```

```

    <div style="padding:10px;color:white;" aria-label="Description">
      ${this.message}
    </div>
  `,
  position: userPosition,
  pixelOffset: [0, -18],
  fillColor: 'rgba(0,0,0,0.8)',
  closeButton: false
});

popupNotification.open(this.map);
}

geoError(errorCode) {
  // 0 desconhecido
  // 1 permissão
  // 2 posição indisponível
  // 3 timeout
  const error = {
    0: alert('Ops, erro desconhecido para assistir sua localização :('),
    1: alert('Ops, você não autorizou para assistir sua localização :('),
    2: alert('Ops, sua posição está indisponível :('),
    3: alert('Ops, não foi possível pegar seus dados da localização :('),
  };

  return error[errorCode];
}
}

import firebase from 'firebase/app';

```



```

import 'firebase/database';

export default class RealTimeDb {
  constructor() {
    // Referencia para o banco de dados Realtime
    this.dbRealTime = firebase.database();
  }

  /**
   * @param {number} latitude
   * @param {number} longitude
   * @param {number} raio
   * @param {string} uid
   * @param {boolean} locked
   */
  async saveGeoFence({ latitude, longitude, radius }, uid, locked) {
    try {
      // Salva os dados no banco de dados Realtime, como referencia o uid do usuário
      await this.dbRealTime.ref(`geofence/${uid}`).set({
        latitude,
        longitude,
        radius,
        locked
      });
    } catch (error) {
      console.log('error in save geofence', error);
    }
  }

  async getGeoFence(uid) {
    try {

```

```

// Obtém os dados do usuário salvo no banco de dados Realtime
await this.dbRealTime.ref(`geofence/${uid}`).once('value').then(snapshot => {
  this.markedArea = snapshot.val();
});
} catch (error) {
  console.log('error in get geofence', error);
}
}
}

// Views:

export const elements = {
  mainContainer: document.querySelector('.container')
};

export const renderLoader = () => {
  const markup = `
    <div class="loader">
      <div class="line"></div>
      <div class="line"></div>
      <div class="line"></div>
      <div class="line"></div>
    </div>
  `;

  elements.mainContainer.insertAdjacentHTML('afterbegin', markup);
};

export const clearLoader = () => {
  elements.mainContainer.innerHTML = "";
};

```

```
};
```

```
export const clearUI = () => {  
  elements.mainContainer.innerHTML = "";  
};
```

```
export const clearModal = (modalElement) => {  
  modalElement.innerHTML = "";  
};
```

```
export const getElement = (selector) => {  
  const element = document.querySelector(selector);  
  return element;  
}
```

```
export const removeElement = (selector) => {  
  document.querySelector(selector).remove();  
};
```

```
export const renderToast = ({ title, action }) => {  
  const markup = `  
    <div class="toast ${action}">  
      <div class="toast__content">  
        <div class="toast__header">${title}</div>  
      </div>  
    </div>  
  `;  
  
  elements.mainContainer.insertAdjacentHTML('afterbegin', markup);  
};
```

```
export const clearToast = () => {
  setTimeout(() => {
    removeElement('.toast--show');
  }, 2000);
};

const stateVideo = {};

const verifyCameraInNavigator = () => {
  if (!"mediaDevices" in navigator || !"getUserMedia" in navigator.mediaDevices) {
    alert("Camera API is not available in your browser");
    return;
  }
};

export const constraints = {
  video: {
    width: {
      min: 300,
      ideal: 200,
      max: 500,
    },
    height: {
      min: 100,
      ideal: 200,
      max: 500,
    },
    facingMode: 'user'
  },
};
```

```

export const stopVideoStream = () => {
  if (stateVideo.videoStream) {
    stateVideo.videoStream.getTracks().forEach((track) => {
      track.stop();
    });
  }
};

const renderCameraActions = (modalElement) => {
  const markup = `
    <div class="modal__camera-actions">
      <a class="modal__btn-camera btn--screenshot">
        
      </a>
    </div>
  `;

  modalElement.insertAdjacentHTML('beforeend', markup);
};

export const renderCamera = async (modalElement) => {
  verifyCameraInNavigator();
  stateVideo.videoStream = await navigator.mediaDevices.getUserMedia(constraints);
  stateVideo.video = document.createElement('video');
  stateVideo.video.setAttribute('autoplay', 'autoplay');
  stateVideo.video.srcObject = stateVideo.videoStream;

  modalElement.appendChild(stateVideo.video);
  renderCameraActions(modalElement);
  renderPicture(modalElement);
  renderButtonToCheckFace(modalElement);
};

```

```
};
```

```
export const renderScreenshot = () => {
  stateVideo.canvas = document.createElement('canvas');
  stateVideo.screenshotImage = document.querySelector('.modal__screenshot-image');
  stateVideo.btnToCheckFace = document.querySelector('.btn-check-face');

  stateVideo.canvas.width = stateVideo.video.videoWidth;
  stateVideo.canvas.height = stateVideo.video.videoHeight;
  stateVideo.canvas.getContext('2d').drawImage(stateVideo.video, 0, 0);
  stateVideo.screenshotImage.src = stateVideo.canvas.toDataURL('image/png');
  stateVideo.screenshotImage.classList.remove('is-hidden');
  stateVideo.btnToCheckFace.classList.remove('is-hidden');
};
```

```
export const renderPicture = (modalElement) => {
  const markup = `
    <img class="modal__screenshot-image is-hidden" alt="">
  `;

  modalElement.insertAdjacentHTML('afterbegin', markup);
};
```

```
export const renderButtonToCheckFace = (modalElement) => {
  const markup = `
    <button class="btn-check-face is-hidden">
      
      Verificar face
    </button>
  `;
};
```

```

modalElement.insertAdjacentHTML('beforeend', markup);
};

import { getElement } from './base';

export const renderFaceScanner = () => {
  const markup = `
    <div class="face-id-wrapper">
      <svg class="face-id-default" enable-background="new 0 0 91 91" height="91px"
id="Layer_1" version="1.1" viewBox="0 0 91 91" width="91px">
        <g>
          <path d="M53.561,66.82c-6.977,1.621-15.125,0.238-15.207,0.225c-1.371-0.238-
2.668,0.678-2.906,2.043
c-
0.238,1.367,0.674,2.668,2.041,2.906c0.23,0.041,3.746,0.643,8.297,0.643c2.757,0,5.896-
0.221,8.914-0.922 c1.352-0.314,2.192-1.666,1.879-3.018S54.912,66.504,53.561,66.82z"/>
          <path d="M81.23,37.238c-0.004-0.332,0.002-0.662,0.008-0.99c0.01-0.6,0.018-1.199-
0.012-1.801 c-0.281-5.872-2.088-11.524-5.373-
16.801C72.736,12.634,68.6,8.535,63.561,5.46c-5.156-3.146-10.758-4.838-16.649-5.029 c-
5.744-0.189-11.285,1.043-16.453,3.656c-6.607,3.344-11.824,8.342-15.504,14.854c-
2.902,5.145-4.432,10.541-4.551,16.04 c-
0.082,3.902,0.088,7.844,0.506,11.717c0.723,6.692,2.018,12.549,3.963,17.897c3.328,9.148,9.
785,15.926,14.615,20.002
c4.035,3.406,7.309,5.098,12.693,5.98c1.221,0.199,2.359,0.297,3.479,0.297c4.856,0,8.603-
1.18,11.925-3.311 c8.402-5.381,14.986-13.229,18.541-
22.092C79.543,56.947,81.213,47.723,81.23,37.238z M54.873,83.334 c-4.834,3.098-
7.609,2.982-11.88,2.285c-4.133-0.678-6.635-1.797-10.266-4.861c-4.363-3.682-10.184-
9.773-13.133-17.879 c-1.805-4.965-3.012-10.434-3.689-16.721c-0.395-3.66-0.555-7.385-
0.479-11.07c0.102-4.663,1.414-9.264,3.902-13.676 c3.184-5.633,7.691-9.953,13.396-
12.842c4.111-2.078,8.496-3.133,13.035-3.133c0.328,0,0.656,0.006,0.988,0.016
c5.012,0.164,9.789,1.609,14.194,4.299c4.363,2.66,7.943,6.209,10.646,10.549c2.826,4.541,4.
381,9.383,4.619,14.389 c0.024,0.494,0.017,0.984,0.008,1.475c-0.008,0.375-0.014,0.752-
0.01,1.098c-0.017,9.803-1.567,18.418-4.744,26.34
C68.293,71.504,62.402,78.512,54.873,83.334z"/>

```

```
<path d="M42.311,62.016c0.455,0.844,1.32,1.32,2.213,1.32c0.404,0,0.813-0.096,1.191-0.301c2.623-1.416,4.889-2.807,4.984-2.865 c1.182-0.729,1.549-2.275,0.822-3.457c-0.729-1.182-2.275-1.549-3.457-0.822c-0.021,0.014-2.236,1.373-4.734,2.724 C42.108,59.271,41.653,60.795,42.311,62.016z"/>
```

```
<path d="M41.139,44.154c0.672,0,1.344-0.27,1.838-0.801c0.947-1.014,0.891-2.604-0.125-3.551 c-7.838-7.307-17.375-4.771-21.322-2.412c-1.191,0.713-1.578,2.256-0.867,3.447c0.713,1.189,2.256,1.576,3.447,0.865 c0.08-0.049,8.25-4.809,15.316,1.775C39.911,43.93,40.526,44.154,41.139,44.154z"/>
```

```
<path d="M69.711,37.311c-0.369-0.188-9.098-4.557-16.709-0.723c-6.434,3.236-6.973,10.004-6.816,14.814 c0.043,1.358,1.158,2.432,2.508,2.432c0.027,0,0.058-0.002,0.084-0.002c1.388-0.045,2.476-1.205,2.431-2.592 c-0.185-5.678,0.992-8.623,4.054-10.164c5.319-2.678,12.102,0.676,12.17,0.711c1.233,0.631,2.743,0.137,3.375-1.096 C71.436,39.455,70.945,37.941,69.711,37.311z"/>
```

```
<path d="M2.84,18.222c1.389,0,2.514-1.125,2.514-2.514V5.437h10.27c1.389,0,2.514-1.125,2.514-2.514 c0-1.387-1.125-2.512-2.514-2.512H2.84c-1.387,0-2.512,1.125-2.512,2.512v12.785C0.329,17.097,1.454,18.222,2.84,18.222z"/>
```

```
<path d="M88.334,0.412H75.551c-1.389,0-2.514,1.125-2.514,2.512c0,1.389,1.125,2.514,2.514,2.514h10.27v10.271 c0,1.389,1.125,2.514,2.515,2.514c1.388,0,2.513-1.125,2.513-2.514V2.923C90.846,1.537,89.721,0.412,88.334,0.412z"/>
```

```
<path d="M15.624,85.768H5.354V75.496c0-1.389-1.125-2.514-2.514-2.514c-1.387,0-2.512,1.125-2.512,2.514v12.785 c0,1.387,1.125,2.512,2.512,2.512h12.783c1.389,0,2.514-1.125,2.514-2.512C18.137,86.893,17.012,85.768,15.624,85.768z"/>
```

```
<path d="M88.334,72.982c-1.389,0-2.514,1.125-2.514,2.514v10.271H75.55c-1.389,0-2.514,1.125-2.514,2.514 c0,1.387,1.125,2.512,2.514,2.512h12.783c1.388,0,2.513-1.125,2.513-2.512V75.496C90.846,74.107,89.721,72.982,88.334,72.982z"/>
```

```
</g>
```

```
</svg>
```

```
<svg class="face-id-checked" enable-background="new 0 0 91 91" height="91px" id="Layer_1" version="1.1" viewBox="0 0 91 91" width="91px">
```

```
<g>
```



<path d="M2.669,18.273c1.388,0,2.513-1.125,2.513-2.513V5.488h10.271c1.388,0,2.513-1.125,2.513-2.513s-1.125-2.513-2.513-2.513 H2.669c-1.388,0-2.513,1.125-2.513,2.513V15.76C0.156,17.148,1.281,18.273,2.669,18.273z"/>

<path d="M88.163,0.462H75.38c-1.389,0-2.514,1.125-2.514,2.513s1.125,2.513,2.514,2.513h10.27V15.76c0,1.388,1.125,2.513,2.514,2.513c1.387,0,2.512-1.125,2.512-2.513V2.975C90.675,1.587,89.55,0.462,88.163,0.462z"/>

<path d="M15.452,85.818H5.182V75.547c0-1.388-1.125-2.513-2.513-2.513s-2.513,1.125-2.513,2.513v12.784 c0,1.388,1.125,2.513,2.513,2.513h12.783c1.388,0,2.513-1.125,2.513-2.513S16.84,85.818,15.452,85.818z"/>

<path d="M88.163,73.034c-1.389,0-2.514,1.125-2.514,2.513v10.271H75.38c-1.389,0-2.514,1.125-2.514,2.513s1.125,2.513,2.514,2.513 h12.783c1.387,0,2.512-1.125,2.512-2.513V75.547C90.675,74.159,89.55,73.034,88.163,73.034z"/>

<path d="M64.739,27.448c-1.023-0.935-2.613-0.862-3.549,0.163l-21.336,23.39l-5.68-6.096c-0.943-1.014-2.534-1.07-3.551-0.125 c-1.015,0.946-1.071,2.536-0.125,3.551l7.537,8.09c0.475,0.51,1.141,0.8,1.838,0.8c0.004,0,0.009,0,0.014,0 c0.701-0.004,1.369-0.301,1.843-0.819l23.173-25.404C65.837,29.972,65.765,28.383,64.739,27.448z"/>

<path d="M67.331,85.843c-1.951-0.275-3.305-1.038-4.139-2.331c-1.803-2.799-0.963-7.625-0.486-9.252 c0.002-0.005,0.002-0.009,0.004-0.014c4.549-4.255,8.139-9.403,10.385-15.005c3.082-7.684,4.588-16,4.604-25.457 c-0.004-0.298,0.002-0.596,0.006-0.894c0.01-0.539,0.018-1.078-0.01-1.619c-0.254-5.314-1.891-10.433-4.863-15.213 c-2.822-4.534-6.566-8.243-11.125-11.024c-4.67-2.848-9.742-4.38-15.076-4.554c-5.207-0.166-10.223,0.944-14.902,3.31 c-5.979,3.028-10.699,7.551-14.03,13.444c-2.625,4.655-4.011,9.542-4.118,14.523c-0.074,3.522,0.079,7.073,0.456,10.552 c0.649,6.027,1.818,11.305,3.574,16.135c2.521,6.927,7.02,12.341,10.978,16.081c0.292,1.172,1.34,6.027-0.551,8.975 c-0.833,1.3-2.189,2.066-4.145,2.342c-1.374,0.194-2.331,1.466-2.137,2.84c0.177,1.255,1.252,2.161,2.485,2.161 c0.117,0,0.235-0.008,0.354-0.024c3.395-0.479,6.048-2.072,7.672-4.605c1.406-2.192,1.858-4.749,1.898-7.045 c2.379,1.704,4.821,2.938,8.167,3.484c1.11,0.182,2.146,0.271,3.166,0.271c4.43,0,7.832-1.789,10.848-3.723 c0.24-0.153,0.473-0.314,0.707-0.473c-0.012,2.404,0.408,5.151,1.904,7.485c1.625,2.533,4.277,4.126,7.672,4.605 c0.119,0.017,0.238,0.024,0.355,0.024c1.232,0,2.307-0.906,2.484-

2.161C69.661,87.309,68.706,86.037,67.331,85.843z M53.634,74.971 c-4.283,2.745-  
6.717,3.342-10.491,2.724c-3.719-0.607-5.877-2.303-9.093-5.019c-3.895-3.286-9.088-8.721-  
11.719-15.948 c-1.615-4.443-2.695-9.336-3.3-14.956c-0.354-3.267-0.498-6.601-0.428-  
9.906c0.09-4.144,1.258-8.236,3.471-12.16 c2.833-5.013,6.844-8.859,11.922-11.431c3.655-  
1.849,7.557-2.786,11.594-2.786c0.291,0,0.582,0.005,0.875,0.015  
c4.457,0.145,8.703,1.43,12.623,3.821c3.885,2.369,7.072,5.528,9.475,9.389c2.516,4.043,3.9,  
8.349,4.111,12.798 c0.021,0.434,0.014,0.865,0.006,1.296c-0.006,0.345-0.012,0.69-  
0.008,1.002c-0.014,8.772-1.402,16.48-4.242,23.562  
C65.606,64.415,60.351,70.666,53.634,74.971z"/>

</g>

</svg>

<svg class="face-id-error" enable-background="new 0 0 91 91" height="91px"  
id="Layer\_1" version="1.1" viewBox="0 0 91 91" width="91px">

<g>

<path d="M2.682,18.273c1.387,0,2.512-1.125,2.512-  
2.513V5.488h10.271c1.387,0,2.512-1.125,2.512-2.513s-1.125-2.513-2.512-2.513 H2.682c-  
1.389,0-2.514,1.125-2.514,2.513V15.76C0.168,17.148,1.293,18.273,2.682,18.273z"/>

<path d="M88.174,0.462H75.391c-1.387,0-2.512,1.125-  
2.512,2.513s1.125,2.513,2.512,2.513h10.271V15.76  
c0,1.388,1.125,2.513,2.512,2.513c1.389,0,2.514-1.125,2.514-  
2.513V2.975C90.688,1.587,89.563,0.462,88.174,0.462z"/>

<path d="M15.465,85.818H5.193V75.547c0-1.388-1.125-2.513-2.512-2.513c-1.389,0-  
2.514,1.125-2.514,2.513v12.784 c0,1.388,1.125,2.513,2.514,2.513h12.783c1.387,0,2.512-  
1.125,2.512-2.513S16.852,85.818,15.465,85.818z"/>

<path d="M88.174,73.034c-1.387,0-2.512,1.125-2.512,2.513v10.271H75.391c-  
1.387,0-2.512,1.125-2.512,2.513s1.125,2.513,2.512,2.513 h12.783c1.389,0,2.514-  
1.125,2.514-2.513V75.547C90.688,74.159,89.563,73.034,88.174,73.034z"/>

<path d="M67.344,85.843c-1.951-0.275-3.305-1.038-4.139-2.331c-1.805-2.799-0.963-  
7.625-0.486-9.252c0-0.005,0-0.009,0.002-0.014 c4.551-4.255,8.141-9.403,10.387-  
15.005c3.082-7.686,4.588-16.002,4.604-25.457c-0.004-0.298,0.002-0.595,0.006-0.892  
c0.01-0.54,0.016-1.079-0.01-1.621c-0.254-5.314-1.891-10.433-4.865-15.213c-2.82-4.534-  
6.564-8.243-11.123-11.024 c-4.67-2.848-9.742-4.38-15.076-4.554c-5.211-0.168-  
10.225,0.945-14.902,3.31c-5.98,3.028-10.699,7.55-14.029,13.444 c-2.627,4.656-

4.012,9.542-4.119,14.523c-  
0.074,3.521,0.078,7.072,0.455,10.553c0.648,6.025,1.818,11.303,3.574,16.135  
c2.521,6.929,7.021,12.344,10.98,16.084c0.48,1.835,1.156,6.321-0.559,8.981c-0.834,1.294-  
2.188,2.057-4.141,2.332 c-1.373,0.194-2.33,1.466-  
2.137,2.84c0.178,1.255,1.254,2.161,2.486,2.161c0.115,0,0.234-0.008,0.354-0.024 c3.395-  
0.479,6.049-2.072,7.674-4.605c1.406-2.192,1.857-4.749,1.898-  
7.045c2.377,1.704,4.82,2.938,8.166,3.484  
c1.111,0.182,2.146,0.271,3.166,0.271c4.43,0,7.832-1.789,10.848-3.723c0.238-0.153,0.471-  
0.314,0.707-0.473 c-  
0.014,2.404,0.406,5.151,1.904,7.485c1.623,2.533,4.277,4.126,7.672,4.605c0.119,0.017,0.238  
,0.024,0.354,0.024 c1.232,0,2.309-0.906,2.486-  
2.161C69.674,87.309,68.717,86.037,67.344,85.843z M53.646,74.971 c-4.285,2.746-  
6.721,3.343-10.49,2.724c-3.721-0.607-5.879-2.303-9.094-5.019c-3.895-3.286-9.088-8.721-  
11.719-15.948 c-1.615-4.445-2.695-9.338-3.301-14.956c-0.354-3.269-0.496-6.602-0.428-  
9.906c0.09-4.144,1.258-8.236,3.471-12.161 c2.832-5.013,6.844-8.859,11.924-11.43c3.654-  
1.849,7.555-2.786,11.594-2.786c0.291,0,0.582,0.005,0.875,0.015  
c4.455,0.145,8.703,1.43,12.623,3.821c3.885,2.369,7.072,5.528,9.475,9.389c2.516,4.043,3.89  
8,8.349,4.111,12.798 c0.021,0.436,0.012,0.868,0.006,1.301c-0.006,0.344-0.012,0.687-  
0.008,0.997c-0.014,8.771-1.402,16.478-4.242,23.562  
C65.619,64.415,60.363,70.666,53.646,74.971z"/>

<path d="M62.539,58.896c0.982-0.981,0.982-2.572,0-3.554L49.201,42.005113.338-  
13.338c0.982-0.981,0.982-2.572,0-3.553 c-0.98-0.981-2.572-0.981-  
3.553,0L45.648,38.453L32.311,25.115c-0.98-0.981-2.572-0.981-3.553,0c-0.98,0.981-  
0.98,2.572,0,3.553 113.338,13.338L28.758,55.343c-0.98,0.981-  
0.98,2.572,0,3.554c0.49,0.49,1.135,0.735,1.777,0.735s1.285-0.245,1.775-0.735 113.338-  
13.338113.338,13.338c0.49,0.49,1.135,0.735,1.777,0.735S62.049,59.387,62.539,58.896z"/>

</g>

</svg>

<div class="scan-bar"></div>

</div>

`;

const overlay = getElement('.overlay');

```
overlay.insertAdjacentHTML('afterbegin', markup);
};
```

```
import { elements } from './base';
```

```
export const renderProfile = (user) => {
```

```
  const markup = `
```

```
    <header class="header">
```

```
      <div class="profile">
```

```
        <div class="profile__content">
```

```
          
```

```
          <span class="profile__name">${user.name}</span>
```

```
        </div>
```

```
        <span class="profile__signout">Sair</span>
```

```
      </div>
```

```
    </header>
```

```
    <main class="content-area">
```

```
      <div id="myMap" class="mark-area">
```

```
        <a href="#" class="btn-cta btn-cta--mark-area is-hidden">
```

```
          <div class="btn-cta__icon">
```

```
            <div class="btn-cta__circle-inner"></div>
```

```
            <div class="btn-cta__circle-outer"></div>
```

```
            <svg width="10" height="14" viewBox="0 0 10 14" class="btn-cta__pin">
```

```
              <path fill-rule="evenodd" clip-rule="evenodd" d="M5 14C6.5 14 10 8.5 10 5C10 1.5 7.76142 0 5 0C2.23858 0 0 1.5 0 5C0 8.5 3.5 14 5 14ZM4.99998 7.5C6.38069 7.5 7.49998 6.38071 7.49998 5C7.49998 3.61929 6.38069 2.5 4.99998 2.5C3.61927 2.5 2.49998 3.61929 2.49998 5C2.49998 6.38071 3.61927 7.5 4.99998 7.5Z" fill="#DE6D56"/>
```

```
            </svg>
```

```
          </div>
```

```
          <div class="btn-cta__text">Cadastrar área</div>
```

```

    </a>
  </div>
  <a href="#modal">
    <button class="btn">
      <span class="btn__visible">Destrancar local</span>
      <span class="btn__invisible">Clique para verificar</span>
    </button>
  </a>

  <div id="modal" class="overlay">
    <div class="modal">
      
      <h2 class="modal__title">Local trancado</h2>
      <a href="#" class="modal__close">&times;</a>
      <p class="modal__content">
        Deseja destrancar o local?
      </p>
      <div class="modal__actions">
        <button class="modal__btn">Sim</button>
      </div>
    </div>
  </div>
</main>
`;

elements.mainContainer.insertAdjacentHTML('afterbegin', markup);
};

import { elements } from './base';

export const renderSignIn = () => {

```

```

const markup = `
  <div class="form">
    <form class="form__content">
      <input type="email" name="email" id="email" placeholder="Seu e-mail"
class="form__input">
      <input type="password" name="password" id="password" placeholder="Sua senha"
class="form__input">
      <button type="submit" class="btn btn--signin">
        <span class="btn__visible">Entrar</span>
        <span class="btn__invisible">Vamos lá?</span>
      </button>
    </form>
    <div class="signup">
      <span>Não tem cadastro? <span class="signup__link-to-signup">Cadastrar-
se</span></span>
    </div>
  </div>
`;

```

```

elements.mainContainer.insertAdjacentHTML('afterbegin', markup);
};

```

```

import { elements } from './base';

```

```

export const renderSignUp = () => {

```

```

  const markup = `
    <div class="form">
      <form class="form__content">
        <input type="text" name="name" id="name" placeholder="Seu nome"
class="form__input">

```

```

    <input type="text" name="photoURL" id="photoURL" placeholder="Sua photo URL"
class="form__input">
    <input type="email" name="email" id="email" placeholder="Seu e-mail"
class="form__input">
    <input type="password" name="password" id="password" placeholder="Sua senha"
class="form__input">
    <button type="submit" value="Submit" class="btn btn--signup">
    <span class="btn__visible">Cadastrar</span>
    <span class="btn__invisible">Vamos lá?</span>
    </button>
  </form>
</div>
`;

```

```

elements.mainContainer.insertAdjacentHTML('afterbegin', markup);
};

```

```

// Utils:

```

```

const deg2rad = (deg) => {
  return deg * (Math.PI / 180)
};

```

```

export const calculateDistance = (geoFenceLatitude, geoFenceLongitude, currentLatitude,
currentLongitude) => {
  const R = 6371;
  let dLat = deg2rad(currentLatitude - geoFenceLatitude);
  let dLon = deg2rad(currentLongitude - geoFenceLongitude);
  let a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos(deg2rad(geoFenceLatitude)) * Math.cos(deg2rad(currentLatitude)) *
    Math.sin(dLon / 2) * Math.sin(dLon / 2);

```

```

let c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
let d = R * c * 1000;
return d.toFixed();
};

export const formatDataSourceToGeoFence = (dataSource) => {
  let result = JSON.parse(JSON.stringify(dataSource.toJson()));
  let geometry = result.features[0]['geometry'];

  return {
    longitude: geometry.coordinates[0],
    latitude: geometry.coordinates[1],
    radius: result.features[0]['properties'].radius
  };
};

// Config:

export const subscriptionKeyFace = "";
export const subscriptionKeyMaps = "";
export const clientID = "";

export const faceApiUrl = 'https://geo-face.cognitiveservices.azure.com/face/v1.0';

export const imageParams =
'returnFaceId=true&returnFaceLandmarks=false&returnFaceAttributes=age,gender,smile,faceHair,glasses,emotion,hair,makeup,accessories,headPose';

export const detectUri = `${faceApiUrl}/detect?${imageParams}`;

export const verifyUri = `${faceApiUrl}/verify`;

```



```
export const firebaseConfig = {
  apiKey: "",
  authDomain: "",
  databaseURL: "",
  projectId: "",
  storageBucket: "",
  messagingSenderId: "",
  appId: "",
  measurementId: ""
};

// Controller:
import firebase from 'firebase/app';

import Auth from './models/Auth';
import Face from './models/Face';
import Maps from './models/Maps';
import MapsDrawing from './models/MapsDrawing';
import RealTimeDb from './models/RealtimeDb';

import * as signupView from './views/signupView';
import * as signinView from './views/signinView';
import * as profileView from './views/profileView';
import * as cameraView from './views/cameraView';
import * as faceScannerView from './views/faceScannerView';

import {
  elements,
  clearUI,
  clearLoader,
```

```
clearModal,  
renderLoader,  
renderToast,  
clearToast,  
removeElement,  
getElement  
} from './views/base';  
  
import { formatDataSourceToGeoFence } from './utils/index';  
import { firebaseConfig } from './config';  
  
import './scss/main.scss';  
  
const state = {};  
  
firebase.initializeApp(firebaseConfig);  
  
const signUpController = async ({ email, password, name, photoURL }) => {  
  if (!state.auth) state.auth = new Auth();  
  
  clearUI();  
  renderLoader();  
  
  try {  
    await state.auth.signUp(email, password, { name, photoURL });  
    clearLoader();  
    signinView.renderSignIn();  
  } catch (error) {  
    clearLoader();  
    console.log('error add data', error);  
  }  
}
```

```
}  
};
```

```
const signInController = async ({ email, password }) => {  
  if (!state.auth) state.auth = new Auth();
```

```
  clearUI();
```

```
  renderLoader();
```

```
  try {
```

```
    state.user = await state.auth.signIn(email, password);
```

```
    clearLoader();
```

```
    profileView.renderProfile(state.user);
```

```
    localStorage.setItem('user', JSON.stringify(state.user));
```

```
    setMapsController();
```

```
    if (state.maps) {
```

```
      await getGeoFenceController(state.user.uid);
```

```
    }
```

```
  } catch (error) {
```

```
    clearLoader();
```

```
    console.log('error add data', error);
```

```
  }
```

```
};
```

```
const signOutController = async () => {
```

```
  if (!state.auth) state.auth = new Auth();
```

```
  clearUI();
```

```
  renderLoader();
```

```
  try {
```

```
    await state.auth.signOut();
    localStorage.clear();
    clearLoader();
    signinView.renderSignIn();
  } catch (error) {
    clearLoader();
    console.log('error logout', error);
  }
};

const faceController = async () => {
  const screenshotImage = document.querySelector('.modal__screenshot-image').src;

  if (!state.face) state.face = new Face();

  faceScannerView.renderFaceScanner();

  const scan = getElement('.face-id-wrapper');
  scan.classList.remove('scan-success');
  scan.classList.remove('scan-error');
  scan.classList.add('animate-scan');

  try {
    const blob = await state.face.makeBlob(screenshotImage);
    await state.face.detectFaceUrl(state.user.photoURL);
    await state.face.detectFace(blob);
    await state.face.identifyFace();

    scan.classList.remove('animate-scan');
```

```

    if (state.face.faceVerifyResult.isIdentical && state.mapsDrawing.message === 'Você está
dentro da área demarcada.') {
      scan.classList.add('scan-success');
      setTimeout(() => removeElement('.overlay'), 1500);
      renderToast({
        title: 'Local destrancado com sucesso.',
        action: 'toast--success toast--show'
      });
      clearToast();
    } else {
      scan.classList.add('scan-error');
      setTimeout(() => removeElement('.face-id-wrapper'), 1500);
    }
  } catch (error) {
    console.log('error in face controller', error);
  }
};

const cameraController = async (modalElement) => {
  cameraView.stopVideoStream();

  try {
    await cameraView.renderCamera(modalElement);
  } catch (error) {
    console.log('error in camera controller', error);
  }
};

const modalController = async (modalElement) => {
  clearModal(modalElement);

```

```
try {
  await cameraController(modalElement);
} catch (error) {
  console.log('error modal controller', error);
}
};

const setMapsController = () => {
  if (!state.maps) state.maps = new Maps();

  try {
    state.maps.getMap();
    state.mapsDrawing = new MapsDrawing(state.maps.map);
    state.mapsDrawing.setMeasuringToolCircle();
  } catch (error) {
    console.log('error in set maps controller', error);
  }
};

const databaseController = async () => {

  if (!state.databaseRealTime) state.databaseRealTime = new RealTimeDb();

  try {
    const geofence = formatDataSourceToGeoFence(window.dataSource);

    state.locked = false;

    await state.databaseRealTime.saveGeoFence(geofence, state.user.uid, state.locked);
```

```
const btnMarkArea = getElement('.btn-cta--mark-area');
btnMarkArea.classList.add('is-hidden');

renderToast({
  title: 'Área cadastrada com sucesso',
  action: 'toast--success toast--show'
});

clearToast();

location.reload();
} catch (error) {
  console.log('error in database controller', error);
}
};

const getGeoFenceController = async (uid) => {

  if (!state.databaseRealTime) state.databaseRealTime = new RealTimeDb();

  try {
    await state.databaseRealTime.getGeoFence(uid);

    if (state.databaseRealTime.markedArea) {
      await state.mapsDrawing.getCircleToMap(state.databaseRealTime.markedArea);
    }
  } catch (error) {
    console.log('error in get geofence controller', error)
  }
};
```

```
if (localStorage && localStorage.getItem('user')) {
  clearUI();
  state.user = JSON.parse(localStorage.getItem('user'));
  profileView.renderProfile(state.user);
  setMapsController();

  if (state.maps && state.user.uid) {
    getGeoFenceController(state.user.uid);
  }
}

// Service Worker to PWA

if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js')
    .then((reg) => console.log('service worker registered', reg))
    .catch((err) => console.log('service worker not register', err));
}

elements.mainContainer.addEventListener('click', event => {
  const linkToSignUp = event.target.closest('.signup__link-to-signup');
  const btnSignIn = event.target.closest('.btn--signin');
  const btnSignUp = event.target.closest('.btn--signup');
  const logout = event.target.closest('.profile__signout');
  const btnModal = event.target.closest('.modal__btn');
  const btnScreenshot = event.target.closest('.btn--screenshot');
  const btnCheckToFace = event.target.closest('.btn-check-face');
  const btnRegisterArea = event.target.closest('.btn-cta--mark-area');
```



```
if (linkToSignUp) {  
  clearUI();  
  signUpView.renderSignUp();  
}
```

```
if (btnSignUp) {  
  const data = {  
    email: document.getElementById('email').value,  
    password: document.getElementById('password').value,  
    name: document.getElementById('name').value,  
    photoURL: document.getElementById('photoURL').value,  
  };  
  signUpController(data);  
}
```

```
if (btnSignIn) {  
  const data = {  
    email: document.getElementById('email').value,  
    password: document.getElementById('password').value,  
  };  
  signInController(data);  
}
```

```
if (logout) {  
  signOutController();  
}
```

```
if (btnModal) {  
  const modalElement = document.querySelector('.modal');  
  modalController(modalElement);  
}
```

```
}

if (btnScreenshot) {
  cameraView.renderScreenshot();
}

if (btnCheckToFace) {
  faceController();
}

if (btnRegisterArea && window.dataSource) {
  databaseController();
}
});
```

*// Styles:*

```
:root {
  --color-primary: #abf0e9;
  --color-primary-light: #d4f3ef;
  --color-primary-dark: #63b7af;
  --color-black: #000;
  --color-white: #fff;

  --color-grey-light: #fbfbfb;
  --color-grey-light-2: #f4f2f2;
  --color-grey-light-3: #f0eeee;
  --color-grey-light-4: #ccc;
  --color-grey-light-5: #666;
  --color-grey-dark-1: #333;
```

```

--color-success: #28df99;

--shadow-dark: 0 2rem 6rem rgba(0,0,0,.3);
--shadow-light: 0 2rem 5rem rgba(0,0,0,.06);

--bg-gradient-primary: linear-gradient( 95.2deg,      rgba(173,252,234,1) 26.8%,
    rgba(192,229,246,1) 64% );
--bg-gradient-secondary: linear-gradient(to right, var(--color-primary-light), var(--color-
primary-dark));
}

$bp-large: 68.75em; // 1110px
$bp-medium: 56.25em; // 900px
$bp-small: 37.5em; // 600px
$bp-smallest: 31.25em; // 500px
$bp-tiny: 27.5em; // 348px

* {
  margin: 0;
  padding: 0;
}

*,
*::before,
*::after {
  box-sizing: inherit;
}

html {
  box-sizing: border-box;
  scroll-behavior: smooth;
}

```

```
font-size: 62.5%;
```

```
@media only screen and (max-width: $bp-large) {
```

```
  font-size: 50%;
```

```
}
```

```
}
```

```
body {
```

```
  font-family: 'Open Sans', sans-serif;
```

```
  font-weight: 400;
```

```
  line-height: 1.6;
```

```
  color: var(--color-black);
```

```
  background-image: var(--bg-gradient-primary);
```

```
  min-height: 100vh;
```

```
}
```

```
.container {
```

```
  position: relative;
```

```
  min-height: 100vh;
```

```
  display: flex;
```

```
  justify-content: center;
```

```
  align-items: center;
```

```
  flex-direction: column;
```

```
}
```

```
.header {
```

```
  position: absolute;
```

```
  top: 0;
```

```
  box-shadow: var(--shadow-dark);
```

```
  background-color: var(--color-grey-light);
```

```
padding: 1rem;
min-height: 5rem;
width: 100%;

display: flex;
align-items: center;
justify-content: center;
}
```

```
.content-area {
width: 100%;
display: flex;
justify-content: center;
align-items: center;
flex-direction: column;
}
```

```
.overlay {
position: fixed;
top: 0;
bottom: 0;
left: 0;
right: 0;
background: rgba(0, 0, 0, 0.7);
transition: opacity 500ms;
visibility: hidden;
opacity: 0;

display: flex;
justify-content: center;
align-items: center;
```

```
flex-direction: column;
```

```
&:target {  
  visibility: visible;  
  opacity: 1;  
}  
}
```

```
.form {  
  padding: 2rem;  
  border-radius: .5rem;  
  width: 50rem;  
  background-color: var(--color-white);
```

```
  display: flex;  
  flex-direction: column;
```

```
&__input {  
  font-family: inherit;  
  font-size: 1.5rem;  
  color: inherit;  
  background-color: var(--color-grey-light-2);  
  border: none;  
  padding: .7rem 2rem;  
  border-radius: .2rem;  
  width: 100%;  
  transition: all .2s;  
  margin-right: -3.25rem;
```

```
&:focus {  
  outline: none;
```

```
background-color: var(--color-grey-light-3);
}

&:not(:last-child) {
  margin-bottom: 1rem;
}

&::-webkit-input-placeholder {
  font-weight: 100;
  color: var(--color-grey-light-4);
}
}
}

.signup {
  display: flex;
  justify-content: flex-end;
  font-size: 1.7rem;

  &__link-to-signup {
    color: var(--color-primary-dark);
    font-weight: 700;
    cursor: pointer;
  }
}

.profile {
  width: 85rem;

  display: flex;
  align-items: center;
```

```
justify-content: space-between;
```

```
&__content {  
  display: flex;  
  align-items: center;  
}
```

```
&__avatar {  
  max-width: 10rem;  
  max-height: 10rem;  
  border-radius: 50%;  
  display: block;  
  border: 5px solid var(--color-primary-dark);  
}
```

```
&__name {  
  font-size: 3rem;  
  color: var(--color-grey-dark-1);  
  font-weight: 600;  
  margin-left: 2rem;  
}
```

```
&__signout {  
  font-size: 2rem;  
  color: var(--color-primary-dark);  
  font-weight: 700;  
  text-decoration: underline;  
  cursor: pointer;  
}  
}
```



```
.mark-area {
  width: 35%;
  height: 60rem;
  margin-top: 15rem;
  margin-bottom: 4rem;
  background-color: var(--color-white);
  box-shadow: var(--shadow-light);
  border: 2rem solid var(--color-primary-dark);

  @media only screen and (max-width: $bp-medium) {
    width: 90%;
  }
}

.toast {
  position: fixed;
  z-index: 2;
  width: 30rem;
  border-radius: 1rem;
  bottom: 6rem;
  padding: 2rem;
  box-shadow: var(--shadow-dark);
  transform: translate3d(0, 20rem, 0);
  transition: all .5s;

  &__content {
    max-width: 36rem;
    margin: 0 auto;
  }

  &__header {
```

```
font-size: 1.8rem;
color: var(--color-white);
}
```

```
&--show {
  transform: translate3d(0, 0, 0)
}
```

```
&--success {
  background-color: var(--color-success);
}
}
```

```
.btn-cta {
  text-decoration: none;
  overflow: hidden;
  background-color: var(--color-white);
  box-shadow: 0 .8rem 2rem -.6rem rgba(0,0,0,0.3);
  border-radius: .8rem;
  transition: all 0.1s ease;
  border: none;
```

```
&--mark-area {
  position: absolute;
  bottom: 0;
  left: 50%;
  z-index: 1;
  transform: translate(-50%, -50%);
  margin: 1rem;
}
```

```
&__icon {
    position: relative;
width: 5.5rem;
height: 5.5rem;
    background: #fedecf;
    float: left;
}

&__pin {
    position: absolute;
    top: 0;
    left: 2.3rem;
    transform: translateY(1.8rem);
}

&__circle-inner {
    position: absolute;
    width: 3rem;
    height: 3rem;
    border-radius: 50%;
    background: rgba(255,255,255,0.9);
    margin: 1.3rem;
    box-shadow: 0 1px 2px rgba(0,0,0,0.1);
    transform: scale(1);
}

&__circle-outer {
    position: absolute;
    margin: .5rem;
    width: 4.5rem;
    height: 4.5rem;
```

```
border-radius: 50%;  
background: rgba(255,255,255,0.5);  
box-shadow: 0 1px 2px rgba(0,0,0,0.05);  
transform: scale(1);  
}
```

```
    &__text {  
float: left;  
color: #000;  
white-space: nowrap;  
line-height: 2.5rem;  
height: 100%;  
padding: 1.5rem;  
border-radius: 0 .8rem .8rem 0;  
font-size: 1.8rem;  
font-weight: 500;  
    }  
}
```

```
.btn-check-face {  
font-size: 1.6rem;  
font-weight: 600;  
color: var(--color-grey-dark-1);  
border-radius: .5rem;  
border: none;  
box-shadow: var(--shadow-light);  
cursor: pointer;  
background-image: radial-gradient(  
    circle farthest-corner at 0% 0.5%,  
    rgba(241,241,242,1) 0.1%,  
    rgba(224,226,228,1) 100.2%
```

```
);
```

```
display: flex;  
align-items: center;  
width: 100%;  
justify-content: center;
```

```
&__img {  
  margin-right: .7rem;  
}  
}
```

```
.face-id-wrapper {  
  z-index: 3;  
}
```

```
.face-id-wrapper svg {  
  fill: #aaa;  
  width: 236px;
```

```
@media only screen and (max-width: $bp-large) {  
  width: 150px;  
}  
}
```

```
svg.face-id-checked {  
  fill: #8bc34a;  
  width: 236px;
```

```
@media only screen and (max-width: $bp-large) {  
  width: 150px;
```

```
}  
}  
  
svg.face-id-error {  
  fill: #f44336;  
  width: 236px;  
  
  @media only screen and (max-width: $bp-large) {  
    width: 150px;  
  }  
}  
  
.scan-bar {  
  height: 100%;  
  position: absolute;  
  -webkit-transition: opacity .2s ease;  
  transition: opacity .2s ease;  
  width: 100%;  
  
&::before {  
  content: "";  
  background: #03a9f4;  
  border: 1px solid #9adeff;  
  border-radius: 5px;  
  box-shadow: 2px 2px 3px rgba(51, 51, 51, .3);  
  height: 8px;  
  position: absolute;  
  top: 50%;  
  transform: translateX(-50%) translateY(-50%);  
  width: 16%;  
  z-index: 4;
```

```
@media only screen and (max-width: $bp-large) {  
  width: 50%;  
}  
}  
}  
  
.face-id-default,  
.face-id-checked,  
.face-id-error {  
  height: 100%;  
  left: 50%;  
  position: absolute;  
  top: 50%;  
  -webkit-transform: translateX(-50%) translateY(-50%);  
  transform: translateX(-50%) translateY(-50%);  
  transition: opacity .3s ease-in-out .2s;  
  width: 100%;  
}  
  
.face-id-checked,  
.face-id-error {  
  opacity: 0;  
  transition: opacity .2s ease-in-out 0;  
}  
  
.face-id-wrapper.animate-scan .scan-bar {  
  animation: scanning 2s linear;  
  animation-direction: alternate;  
  animation-iteration-count: 4;  
}
```

```
.face-id-wrapper.scan-success .face-id-checked,  
.face-id-wrapper.scan-error .face-id-error {  
  opacity: 1;  
}
```

```
.face-id-wrapper.scan-success .face-id-default,  
.face-id-wrapper.scan-success .scan-bar,  
.face-id-wrapper.scan-error .face-id-default,  
.face-id-wrapper.scan-error .scan-bar {  
  opacity: 0;  
  visibility: hidden;  
}
```

```
.modal {  
  margin: 30rem auto;  
  padding: 2rem;  
  background-color: var(--color-white);  
  border-radius: .8rem;  
  width: 30rem;  
  position: relative;  
  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  
  &__icon {  
    width: 9rem;  
    height: 9rem;  
    margin-bottom: 2.2rem;  
  }  
}
```



```
&__title {  
  font-size: 2.5rem;  
  font-weight: bold;  
  margin-bottom: .8rem;  
  color: var(--color-grey-dark-1);  
}
```

```
&__close {  
  position: absolute;  
  top: 0.5rem;  
  right: 3rem;  
  font-size: 3rem;  
  font-weight: 700;  
  color: var(--color-grey-dark-1);  
  text-decoration: none;  
  transition: all 200ms;  
  
  :hover {  
    color: var(--color-primary-dark);  
  }  
}
```

```
&__content {  
  font-size: 1.5rem;  
  color: var(--color-grey-light-5);  
  margin-bottom: 2rem;  
  overflow: auto;  
}
```

```
&__btn {
```

```
font-size: 1.8rem;
font-weight: bold;
background-color: var(--color-primary-dark);
border: none;
padding: .8rem 4rem;
color: #fff;
box-shadow: 0 0 0 2px var(--color-primary-dark) inset;
border-radius: .25rem;
cursor: pointer;
transition: background .4s ease, color .4s ease;

&:hover {
  box-shadow: 0 0 0 2px var(--color-primary-dark) inset;
  color: var(--color-primary-dark);
  background-color: transparent;
}
}

&__camera-actions {
  margin: 2rem 0;

  display: flex;
  align-items: center;
  justify-content: center;
}

&__btn-camera {
  width: 100%;
  height: 100%;
  display: block;
  text-decoration: none;
```

```
cursor: pointer;
```

```
img {  
  display: block;  
  width: 100%;  
  height: 100%;  
}  
}
```

```
&__screenshot-image {  
  width: 10rem;  
  height: 6.8rem;  
  border-radius: .4rem;  
  border: 2px solid whitesmoke;  
  box-shadow: 0 1px 2px 0 rgba(0, 0, 0, 0.1);  
  position: absolute;  
  bottom: 22rem;  
  left: 1rem;  
  background: var(--color-white);
```

```
@media only screen and (max-width: $bp-large) {  
  bottom: 27rem;  
}  
}
```

```
video {  
  width: inherit;  
}  
}
```

```
.is-hidden {
```

```
display: none;
}

.btn {
  font-size: 1.5rem;
  font-weight: 300;
  text-transform: uppercase;
  border-radius: .5rem;
  width: 100%;
  margin-bottom: 1.2rem;
  border: none;
  background-image: var(--bg-gradient-secondary);
  color: #fff;
  position: relative;
  overflow: hidden;
  cursor: pointer;

  &> * {
    display: inline-block;
    height: 100%;
    width: 100%;
    transition: all .2s;
  }

  &__visible {
    padding: 1.5rem 7.5rem;
  }

  &__invisible {
    position: absolute;
    padding: 1.5rem 0;
  }
}
```

```
    left: 0;
    top: -100%;
}

&:hover {
    background-image: var(--bg-gradient-secondary);
}

&:hover &__visible {
    transform: translateY(100%);
}

&:hover &__invisible {
    top: 0;
}

&:focus {
    outline: none;
    animation: pulsate 1s infinite;
}
}

.loader {
    height: 60vh;
    display: flex;
    align-items: center;
    margin: 0 auto;
    justify-content: center;
}

.line {
```

```
animation: expand 1s ease-in-out infinite;
display: inline-block;
margin: 0 10px;
width: 10px;
height: 5rem;
transform-origin: center center;
border-radius: 10px;
```

```
@media only screen and (max-width: $bp-small) {
  width: 6px;
  height: 3.5rem;
}
```

```
@media only screen and (max-width: $bp-small) {
  width: 5px;
  height: 3rem;
}
}
```

```
.line:nth-child(1) {
  background: var(--color-primary-light);
}
```

```
.line:nth-child(2) {
  animation-delay: 180ms;
  background: var(--color-primary-dark);
}
```

```
.line:nth-child(3) {
  animation-delay: 360ms;
  background: var(--color-primary-light);
}
```

```
}
```

```
.line:nth-child(4) {  
  animation-delay: 540ms;  
  background: var(--color-primary-dark);  
}
```

```
@keyframes pulsate {  
  0% {  
    transform: scale(1);  
    box-shadow: none;  
  }  
  
  50% {  
    transform: scale(1.05);  
    box-shadow: 0 1rem 4rem rgba(0,0,0,.25);  
  }  
  
  100% {  
    transform: scale(1);  
    box-shadow: none;  
  }  
}
```

```
@keyframes expand {  
  0% {  
    transform: scale(1);  
  }  
  
  25% {  
    transform: scale(2);
```

```
}  
}
```

```
@-webkit-keyframes scanning {  
  0% { transform: translateY(0); }  
  30% { transform: translateY(-50%); }  
  70% { transform: translateY(50%); }  
  100% { transform: translateY(0); }  
}
```

```
@keyframes scanning {  
  0% { transform: translateY(0); }  
  30% { transform: translateY(-50%); }  
  70% { transform: translateY(50%); }  
  100% { transform: translateY(0); }  
}
```

```
@import '../node_modules/azure-maps-control/dist/atlas.min.css';  
@import '../node_modules/azure-maps-drawing-tools/dist/atlas-drawing.min.css';
```

```
@import 'base';  
@import 'layout';  
@import 'components';  
@import 'animations';
```

```
// HTML:
```

```
<!DOCTYPE html>  
<html lang="pt">  
<head>  
  <meta charset="UTF-8">
```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Geo Face</title>
<link rel="preconnect" href="https://fonts.gstatic.com">
<link
href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;400;500;600;700&di
splay=swap" rel="stylesheet">
<link rel="shortcut icon" type="image/png" href="/img/icons/icon-tcc.png"/>
<meta name="theme-color" content="#abf0e9">

<!-- IOS support -->
<link rel="apple-touch-icon" href="img/icons/icon-tcc.png">
<meta name="apple-mobile-web-app-status-bar" content="#abf0e9">
<!-- app title -->
<meta name="apple-mobile-web-app-title" content="Geo Face">
<!-- enable standalone mode -->
<meta name="apple-mobile-web-app-capable" content="yes">
</head>
<body>
<div class="container">
<div class="form">
<form class="form__content">
<input type="email" name="email" id="email" placeholder="Seu e-mail"
class="form__input">
<input type="password" name="password" id="password" placeholder="Sua senha"
class="form__input">
<button type="submit" value="Submit" class="btn btn--signin">
<span class="btn__visible">Entrar</span>
<span class="btn__invisible">Vamos lá?</span>
</button>
</form>
<div class="signup">

```

```

    <span>Não tem cadastro? <span class="signup__link-to-signup">Cadastrar-
se</span></span>
  </div>
</div>
</div>
</body>
</html>

```

```
// Babel:
```

```

{
  "presets": [
    /*
     * Permite usar código JavaScript mais recente sem a necessidade
     * de microgerenciar quais transformações de sintaxe/polyfills
     * são necessárias para o ambiente de destino.
     */
    ["@babel/preset-env", {
      "targets": {
        /*
         * Suporta as últimas 5 versões de cada navegador e certifica de que
         * todas as versões do IE (a partir da 8) sejam compatíveis.
         */
        "browsers": [
          "last 5 versions",
          "ie >= 8"
        ]
      }
    }]
  ]
}

```

```
// Webpack:

const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const MiniCssExtractPlugin = require('mini-css-extract-plugin');
const WorkboxPlugin = require('workbox-webpack-plugin');
const WebpackPwaManifestPlugin = require('webpack-pwa-manifest');

module.exports = (_, { mode }) => ({
  entry: [
    '@babel/polyfill',
    './src/js/index.js'
  ],
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'js/bundle.js'
  },
  devServer: {
    contentBase: path.join(__dirname, 'dist'),
  },
  node: {
    fs: "empty"
  },
  plugins: [
    /**
     * O plugin irá gerar um arquivo HTML5 para que
     * inclui todos os pacotes webpack no corpo usando script tags.
     */
    new HtmlWebpackPlugin({
      filename: 'index.html',

```

```

    template: './src/index.html'
  }),
  /**
   * Este plugin extrai CSS em arquivos separados.
   * Ele cria um arquivo CSS por arquivo JS que contém CSS.
   */
  new MiniCssExtractPlugin({
    filename: 'css/[name].css',
    chunkFilename: 'css/[id].css',
  }),
  /**
   * O plug-in criará um arquivo de service worker chamado sw
   * e o adicionará ao pipeline de ativos do webpack.
   */
  new WorkboxPlugin.GenerateSW({
    // O nome do arquivo que será criado.
    swDest: 'sw.js',
    // Controla clientes existentes assim que for ativado.
    clientsClaim: true,
    // Força o SW em espera e torna o mesmo ativo.
    skipWaiting: true,
    // Determina o tamanho máximo dos arquivos que serão pré-armazenados.
    maximumFileSizeToCacheInBytes: 5000000,
    // Define as regras de armazenamento em cache do tempo de execução.
    runtimeCaching: [{
      // Corresponde a qualquer solicitação que tem fonts.googleapis.
      urlPattern: new RegExp('^https://fonts\\.googleapis\\.com/'),
    }],
  })
  /**
   * Aplica uma estratégia de cache em paralelo com a rede.
   * A estratégia responderá com a versão em cache, se disponível,
   * caso contrário, aguardará a resposta da rede.

```

```

* */
handler: 'StaleWhileRevalidate',
options: {
  // Nome do cache personalizado.
  cacheName: 'google-fonts',
  // Código de status para as respostas que podem ser considerados armazenáveis em cache.
  cacheableResponse: {
    statuses: [0, 200],
  }
}
}],
}),
/**
* Plugin que gera um 'manifest.json' para o Progressive Web Application,
* com redimensionamento automático de ícones.
*/
new WebpackPwaManifestPlugin({
  // Nome abreviado para o aplicativo
  short_name: 'GF',
  // Nome do aplicativo
  name: 'Geo Face',
  // Descrição do aplicativo
  description: 'Geo Face App',
  start_url: '/',
  // Modo de exibição, terá aparência de um aplicativo independente.
  display: 'standalone',
  // Cor de fundo
  background_color: '#abf0e9',
  // Cor do tema
  theme_color: '#abf0e9',
  // Modo retrato principal, posição vertical com os botões na parte inferior.

```

```

orientation: 'portrait-primary',
// Ícone com redimensionamento automático.
icons: [
  {
    src: path.resolve('src/img/icons/icon-tcc.png'),
    sizes: [96, 128, 180, 192, 256, 512],
    type: 'image/png',
    purpose: 'maskable any'
  }
],
}),
],
module: {
  rules: [
    {
      // Este pacote permite transpilar arquivos JavaScript
      test: /\.m?js$/,
      exclude: /node_modules/,
      use: {
        loader: 'babel-loader',
        options: {
          presets: ['@babel/preset-env'],
        },
      }
    },
    {
      /**
      * - style-loader é usado para adicionar a tag <style>
      * ao arquivo html na tag <head>.
      * - css-loader lê em um arquivo css como uma string
      * - sass-loader carrega um arquivo SASS/SCSS e o compila em CSS

```

```

    */
test: /\.s[ac]ss$/i,
use: [
  mode !== 'production'
    ? 'style-loader'
    : MiniCssExtractPlugin.loader,
  'css-loader',
  'sass-loader',
],
},
{
  /**
   * - file-loader resolve import/require()
   * em um arquivo em uma url e emite o arquivo no diretório de saída
   */
test: /\.(png|jpg|svg)$/i,
use: [
  {
    loader: 'file-loader',
    options: {
      name: '[path][name].[ext]',
      context: path.resolve(__dirname, 'src/'),
      outputPath: '/',
      publicPath: './',
      useRelativePaths: true
    }
  }
]
}
],
}
}

```

```
});
```

```
// Package.json
```

```
{  
  "name": "tcc",  
  "version": "1.0.0",  
  "description": "Trabalho de Conclusão de Curso II",  
  "main": "index.js",  
  "scripts": {  
    "dev": "webpack --mode development",  
    "build": "webpack --mode production",  
    "start": "webpack-dev-server --mode development --open"  
  },  
  "author": "Isabel",  
  "license": "ISC",  
  "devDependencies": {  
    "@babel/core": "^7.10.5",  
    "@babel/preset-env": "^7.10.4",  
    "babel-core": "^6.26.3",  
    "babel-loader": "^8.1.0",  
    "babel-preset-env": "^1.7.0",  
    "css-loader": "^3.6.0",  
    "html-webpack-plugin": "^4.3.0",  
    "mini-css-extract-plugin": "^0.9.0",  
    "node-sass": "^4.14.1",  
    "postcss-load-config": "^2.1.0",  
    "postcss-preset-env": "^6.7.0",  
    "resolve-url-loader": "^3.1.1",  
    "sass-loader": "^9.0.2",  
    "style-loader": "^1.2.1",
```



```

    "webpack": "^4.43.0",
    "webpack-cli": "^3.3.12",
    "webpack-dev-server": "^3.11.0",
    "webpack-pwa-manifest": "^4.3.0",
    "workbox-webpack-plugin": "^6.1.2"
  },
  "dependencies": {
    "@babel/polyfill": "^7.10.4",
    "azure-maps-control": "^2.0.31",
    "azure-maps-drawing-tools": "^0.1.6",
    "file-loader": "^6.0.0",
    "firebase": "^8.2.9"
  }
}

// Sistema embarcado:

#include <WiFi.h>
#include <FirebaseESP32.h>

#define WIFI_SSID "" // WiFi Name
#define WIFI_PASSWORD "" // WiFi Password
#define FIREBASE_HOST "" // URL Firebase Server
#define FIREBASE_AUTH "" // Access Key Host Firebase

// Led que mostra o estado da TRANCA, aceso igual aberto e apagado igual fechado
#define LOCKED_LED 2

FirebaseData fbdo;

unsigned long sendDataPrevMillis = 0;

```

```
String path = "/geofence/J3dgd09T84aZ5XLSZMljlGZsT72";
```

```
uint16_t count = 0;
```

```
void printResult(FirebaseData &data);
```

```
void setup() {
```

```
  pinMode(LOCKED_LED, OUTPUT);
```

```
  digitalWrite(LOCKED_LED, LOW);
```

```
  Serial.begin(115200);
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
  Serial.print("Connecting to Wi-Fi");
```

```
  while (WiFi.status() != WL_CONNECTED) {
```

```
    Serial.print(".");
```

```
    delay(300);
```

```
  }
```

```
  if (WiFi.status() == WL_CONNECTED) {
```

```
    Serial.println();
```

```
    Serial.print("Connected with IP: ");
```

```
    Serial.println(WiFi.localIP());
```

```
    Serial.println();
```

```
  }
```

```
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
```

```
  Firebase.reconnectWiFi(true);
```

```
  if (!Firebase.beginStream(fbdo, path)) {
```

```

Serial.println("-----");
Serial.println("Can't begin stream connection...");
Serial.println("REASON: " + fbdo.errorReason());
Serial.println("-----");
Serial.println();
}
}

void loop(){

// Aguarda 3 segundos para a requisição ser completada
if (millis() - sendDataPrevMillis > 3000) {
  sendDataPrevMillis = millis();

  if(Firebase.getBool(fbdo, path + "/locked")){

    if (fbdo.boolData() == 1) {
      // Apaga a LED, informando que a Tranca está Fechada
      digitalWrite(LOCKED_LED, LOW);
      Serial.println("-----");
      Serial.println("Fechado!");
      Serial.println("-----");
      Serial.println();
    } else {
      // Ascende a LED, informando que a Tranca está Aberta
      digitalWrite(LOCKED_LED, HIGH);
      Serial.println("-----");
      Serial.println("Aberto!");
      Serial.println("-----");
      Serial.println();
    }
  }
}
}

```

```
} else {  
    Serial.println("-----");  
    Serial.println("Something was error! :(");  
    Serial.println("Description: " + fbdo.errorReason());  
    Serial.println("-----");  
    Serial.println();  
}  
}  
}
```