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

RECONHECIMENTO DE LOCUTOR COM USO DE GEOFENCING

MAYCHON DOUGLAS DUTRA PIRES

GOIÂNIA
2021

MAYCHON DOUGLAS DUTRA PIRES

RECONHECIMENTO DE LOCUTOR COM USO DE GEOFENCING

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 obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a): Marcelo Antônio Adad de Araújo

GOIÂNIA

2021

MAYCHON DOUGLAS DUTRA PIRES

RECONHECIMENTO DE VOZ COM USO DE GEOFENCING

Este trabalho de Conclusão de Curso julgado adequadamente para obtenção o título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da pontifícia Universidade Católica de Goiás, em 07 de junho de 2021.

_____

Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora 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: Profa. Mirian Sandra Rosa Gusmão, MSc.

GOIÂNIA
2021

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder a dádiva da existência, fôlego de vida, sopro da esperança, metodologia da esperança, coragem para caminhar, saúde para trabalhar, sabedoria para conquistar os objetivos que foram alcançados até aqui. A Ele dirijo o meu melhor obrigado.

Agradeço a minha família, em especial aos meus pais, Maria do Bonfim Dutra da Silva e Tyrone Pires de Lima, que foram base para todas as minhas conquistas, me acompanharam durante toda jornada acadêmica, antes que o dia chegasse a amanhecer acordavam junto comigo, para ajudar-me no preparo de ir à universidade. Estão comigo nas horas boas e nas horas difíceis, sob quaisquer circunstâncias. Em especial, agradeço ao meu irmão Ruan Dutra Maciel que, ainda na minha busca por um ingresso na graduação, em seu horário de almoço do trabalho, se dispôs a me levar à universidade para entrega de documentação de minha candidatura à lista de espera do Prouni (benefício que cobriu todos os custos referentes a minha graduação) e ressalto ainda que, quando pensei em desistir da entrega da documentação, ele me convenceu a insistir, graças a isso estou concluindo a minha graduação. Para eles dirijo o meu muito obrigado.

Agradeço a Pontifícia Universidade Católica de Goiás, através dos meus professores que se puseram disponíveis e se fizeram facilitadores na busca do conhecimento. Em especial agradeço o orientador deste trabalho, Professor Marcelo Antonio Adad de Araújo, que se tornou um Amigo e que fez da realização deste trabalho um momento de descontração, acompanhando e tranquilizando-me nas horas do medo de não conseguir. A eles o meu muito obrigado.

Agradeço aos meus amigos que estiveram comigo durante grande parte da caminhada da graduação, que se fizeram presente em auxílio às dificuldades acadêmicas em ajuda mútua, em especial Isabel Cristina Araújo Rubim Alves que é uma grande amiga e parceira, peça chave na minha conquista da graduação. Posso citar também Carlos Eduardo de Oliveira Bueno, amigo que esteve presente como apoio e na troca de experiências da vida social, profissional e acadêmica. Agradeço ao meu amigo Carlos Henrique Garcia que, antes do início da graduação, ajudou-me no descobrimento da área da computação. A eles dirijo o meu muito obrigado.

Agradeço à UNIGOIÁS, local de trabalho que me forneceu muitas ferramentas para chegar até aqui, onde obtive grande parcela do conhecimento que tenho hoje,

conhecimentos que carrego comigo no dia a dia. Em especial gostaria citar o meu amigo, colega de trabalho e parceiro Gilberto Marques que, de forma direta e indireta, colaborou para o meu aprendizado até aqui. Cito ainda o meu amigo e gestor direto Waldenyr Martins de Souza, que me abriu portas e sempre se fez facilitador do meu sucesso. Para eles dirijo o meu muito obrigado.

RESUMO

Este trabalho apresenta o desenvolvimento de uma aplicação que realiza o acionamento de uma tranca localizada na residência do usuário. A aplicação utiliza autenticação por meio da voz do usuário e de sua localização em relação a uma cerca virtual previamente cadastrada. A utilização de georreferenciamento em consonância com a biometria de voz tem o intuito de aumentar a segurança no acesso à aplicação que realiza o acionamento da tranca na residência do usuário. A gravação da voz do usuário é enviada ao serviço externo responsável por retirar as características da voz e gerar um identificador único do usuário, para que, em fase de autenticação, seja utilizado no reconhecimento de sua voz. A aplicação desenvolvida é passível de acesso via navegador e de instalação em múltiplas plataformas.

Palavras-Chave: *Geofencing*. Reconhecimento de Locutor. Acionamento. Internet das Coisas.

ABSTRACT

This work presents the development of an application that triggers a lock located in the user's home. The application used authentication through the user's voice and its location in relation to a previously registered virtual fence. The use of georeferencing in line with voice biometrics is intended to increase security in accessing the application that triggers the residence in the user's residence. The user's voice recording is channeled to the external service responsible for removing the characteristics of the voice and generating a unique identifier of the user, so that, in the authentication phase, it can be used in the recognition of their voice. The developed application is accessible via browser and can be installed on multiple platforms.

Keywords: *Geofencing. Speaker Recognition. Actuation. Internet of Things*.

LISTA DE FIGURAS

LISTA DE FÓRMULAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	*Application Programming* Interface (Interface de Programação de Aplicação)
BLOB	*Binary Large OBject* (objeto grande binário)
CSS	*Cascading Style Sheets* (Folha de Estilos em Cascata)
HTML	*Internet of Things* (Internet das Coisas)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IDE	*Integrated Development Environmente* (Ambiente de Desenvolvimento Integrado)
IoT	*Internet* of *Things* (Internet das coisas)
IP	Internet *Protocol* (Protocolo de Internet)
JSON	*JavaScript Object Notation* (Notação de Objeto JavaScript)
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
PWA	*Progressive* Web App (Aplicativo Web Progressivo)
SDK	Software *Development* Kit (Kit de Desenvolvimento de Software)
Prof.	Professor
Profa.	Professora
M.E.E.	Mestre em Engenharia Elétrica
MSc.	Mestre em Ciências
Ma.	Mestra

SUMÁRIO

1 INTRODUÇÃO

O trabalho em questão tem como objetivo o estudo e desenvolvimento de um sistema de acionamento de uma tranca, que utiliza o reconhecimento de voz e o georreferenciamento através do uso de APIs, possibilitando o usuário ser autenticado utilizando a biometria e a sua localização. Um Aplicativo Web Progressivo (PWA) é desenvolvido para fazer a interface entre o usuário do dispositivo móvel e o sistema. Este aplicativo faz, para autenticar o usuário, a coleta da voz e localização deste, com os quais irá validar a identidade do usuário. Em fase de autenticação, o PWA faz a gravação da voz do usuário para encaminhá-la para a API de Reconhecimento de Locutor a fim de verificar se é, realmente, o mesmo usuário cadastrado que está falando na gravação enviada. Se sim, o PWA realiza uma verificação, através da API de georreferenciamento, submetendo a localização a uma cerca virtual previamente estabelecida. Caso o usuário não esteja dentro do perímetro da cerca virtual, este não estará autorizado a fazer o acionamento pretendido. Se o usuário estiver dentro da cerca, então, será disponibilizado o acionamento de uma tranca em sua residência. Após ação do usuário, para acionar a tranca, o PWA enviará uma requisição ao banco de dados em nuvem, no qual o Microcontrolador irá verificar se é preciso ou não realizar o destrave de uma tranca.

Este trabalho teve como ponto de partida o trabalho acadêmico intitulado “Verificação Automática Georreferenciada” realizado em 2018 por Paulo Victor Alexandre Alves, bacharel em Engenharia da Computação pela Pontifícia Universidade Católica de Goiás.

1.1 Objetivo

Nesta seção são apresentados os objetivos do presente trabalho.

1.1.1 Objetivo Geral

Validar o usuário que estiver localizado dentro do perímetro de uma cerca virtual previamente cadastrada e que seja autenticado através de sua voz, a fim de habilitá-lo ao desbloqueio de uma tranca, localizada em sua residência.

1.1.2 Objetivos Específicos

Os objetivos específicos são:

- Construir uma aplicação Web;
- Capturar a localização atual de um usuário;
- Verificar se este usuário está localizado dentro do perímetro de uma cerca virtual;
- Verificar se o usuário que está acessando o aplicativo tem permissão para destravar a tranca através da verificação da voz e de sua localização;
- Destravar uma tranca, conectada a um microcontrolador conectado à Internet, localizada na residência do usuário.

1.2 Justificativa

A *Web* (rede) se tornou parte da vida de qualquer indivíduo que esteja socialmente localizado nas regiões urbanas das cidades e muitas vezes nas regiões ruralmente localizadas (CRUZ, 2015). Está acessível a pelo menos 70% dos brasileiros (G1:LAVADO, 2019) e esta porcentagem de inclusão digital não para de crescer. Com ela vem a necessidade de melhorias em diversos âmbitos principalmente a segurança de seus usuários.

Atualmente, o aumento do uso de dispositivos móveis com acesso à internet impulsionou o avanço de tecnologias que colaborassem nas atividades do cotidiano, desde as mais simples até as atividades mais particulares e especializadas (CARRION e QUARESMA, 2019).

1.3. Questões de Pesquisa

A pesquisa por este assunto é justificada pelo aumento de usuários na Web estar em pleno crescimento, resultando em aumento também da demanda de políticas de segurança para benefício do usuário. Para se obter destaque e usabilidade é necessário o reconhecimento do usuário, no contexto trabalhado.

Diante deste contexto, este trabalho visa responder à seguinte questão de pesquisa: Como aprimorar a Segurança do Usuário da Biometria de Reconhecimento da Voz com o uso de *Geofencing*?

Ao longo do presente trabalho a questão anterior, será respondida.

1.4 Procedimentos Metodológicos

Este trabalho é uma Pesquisa Exploratória que se presta, para coleta de informações, com o intuito de auxiliar na construção de um sistema computacional capaz de verificar a identidade e localização do usuário, a fim de acionar um dispositivo, por exemplo uma tranca, localizado em sua residência. Trata-se de uma pesquisa bibliográfica seguida da implementação de um sistema computacional de forma a melhor atingir o objetivo.

Os métodos utilizados para responder à questão de pesquisa são:

1. Compreender os princípios de funcionamento de um microcontrolador para utilizá-lo como interface de um aparelho eletrônico que possa ser acionado por um usuário com um smartphone conectado à internet.
2. Realizar a busca de uma API que possa ser utilizada para fazer a verificação de um usuário com o uso de reconhecimento de voz.
3. Fazer o estudo da documentação de desenvolvimento Web afim de entender o funcionamento do *Geofencing*.
4. Entender como são criadas as Cercas Virtuais e como é verificado se em determinado momento o usuário está no perímetro da cerca virtual.
5. Compreender como é realizado a construção de um PWA, utilizando as Linguagens HTML, CSS e JavaScript.

6. Realizar implementação de um PWA que seja capaz de, com o clique de um botão, fazer a captura de uma gravação de áudio e enviá-la à API de reconhecimento de voz para gerar um identificador deste usuário.

7. Implementar no PWA a função de cadastro de cerca virtual utilizando *geofencing*.

8. Utilizar banco de dados em nuvem para comunicar-se com:

- PWA: faz a conexão com o banco de dados para enviar o ID do usuário caso seja identificado que, a voz e a localização pertençam ao indivíduo previamente cadastrado, autorizando assim o acionamento do microcontrolador;
- Microcontrolador: realiza a conexão com o banco de dados para verificar se é necessário destravar a tranca.

9. Implementar programa para microcontrolador capaz de conectar-se a um banco de dados em nuvem para verificar se é preciso ou não realizar abertura de uma tranca, simulando o acionamento de um dispositivo eletrônico.

1.5 Estrutura

No primeiro capítulo é abordada a introdução, com o objetivo de contextualizar o leitor do trabalho, para deixá-lo a par do que será desenvolvido.

No segundo capítulo descreve-se as ferramentas e conceitos que são necessários para o desenvolvimento do trabalho. Realiza-se a descrição de ferramentas como Sistemas Embarcados e Aplicativos móveis assim como de conceitos de Internet das Coisas, Biometria, Reconhecimento de Locutor e Geofencing.

No terceiro capítulo é apresentado o desenvolvimento da aplicação, onde serão descritas as etapas para solucionar a questão de pesquisa utilizando-se das ferramentas estudadas para atingir o objetivo.

O quarto capítulo contém as observações a respeito da solução aplicada, as considerações finais e sugestões para implementações futuras relacionadas ao tema em questão.

2 REFERENCIAL TEÓRICO

No capítulo corrente é abordado o referencial teórico utilizado para orientação e base de execução do projeto apresentado, descriminando informações essenciais para a melhor compreensão e execução da aplicação proposta.

2.1 Sistemas Embarcados

De acordo com Chase (2007), sistemas embarcados são dispositivos dedicados a realizar uma tarefa específica, de forma a interagir com o contexto a qual está sendo aplicado, através do uso de sensores e atuadores. Sistemas embarcados são aplicáveis a vários projetos como sistemas de freios em automóveis (para gerenciar sensores, sistemas que necessitam de controle de temperatura) o embarcado poderia realizar alguma ação a partir de dados coletados dos sensores de temperatura entre outras aplicações.

Sistemas embarcados são encapsulados para atender a necessidades específicas, diferentemente dos computadores de propósito geral. Este tipo de sistema realiza um conjunto de tarefas previamente estabelecidas, que são descritas para o sistema embarcado com o uso de linguagem de programação (OLIVEIRA, 2018).

2.2 *Internet of Things*

Com o aumento da capacidade de processamento e velocidade dos microchips, sendo alavancado pela evolução da mobilidade tecnológica, computadores e outros dispositivos se tornaram cada vez menores e portáteis (CARRION, QUARESMA; 2019).

Conforme Carrion et al. (2019) Internet das Coisas (IoT) trata-se de um sistema que conecta objetos físicos, através de um endereço de IP ou outra rede, para trocar, armazenar e coletar dados para consumidores e empresas através de uma aplicação de software.

Mais do que uma geladeira conectada, internet das coisas representa a automação em grande escala de setores econômicos inteiros e com base na

interface entre máquinas e os mais diversos setores de: logística, transporte de pessoas, saúde etc. Para isso é necessário favorecer um ambiente com número potencialmente maior de acesso a dispositivos. Além de favorecer um ambiente com muitos dispositivos conectados, é necessário haver privacidade e segurança do usuário como requisito fundamental para o futuro da internet das coisas (MAGRANI, 2018).

A China desde 2011 se destaca no uso de ambientes completamente conectados à Internet. No contexto educacional o governo chinês investe de forma maciça no desenvolvimento de alunos em disciplinas que estão diretamente ligadas a formação tecnológica (ZUIN, 2016).

A grande problemática do avanço da IoT está no fato de que a velocidade que ela se difunde ser maior que a capacidade de prever os impactos positivos ou negativos. Pelo fato de haver muitas áreas de aplicação da IoT, os impactos negativos são temerários (LACERDA e LIMA-MARQUES, 2015).

2.3 Biometria

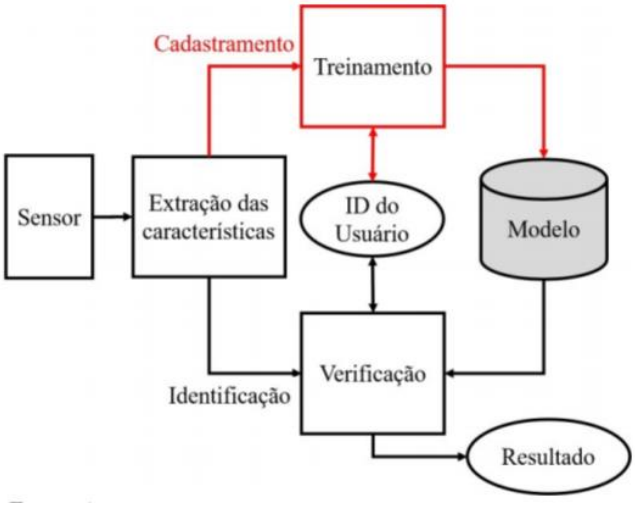
De acordo com Krueger et al. (2013), a palavra Biometria possui origem grega, formada da junção das palavras *bios* e *metron* que significam vida e medida respectivamente. Trata-se do uso de verificação ou reconhecimento do indivíduo utilizando alguma característica anatômica, fisiológica ou comportamental (apud. JAIN, FLYNN e ROSS 2008).

Por muito tempo a biometria, em modalidades diferentes, têm sido aplicadas e estudadas para o uso na segurança e acesso de forma automática. As modalidades de biometria mais utilizadas no comércio são baseadas em: impressão digital, face, íris, voz, assinatura e palma da mão (PARADA, 2018).

Segundo Galimberti (2018) na história pode-se observar povos que utilizaram em algum momento a biometria como forma de identificação de indivíduos, como exemplo o povo chinês que, de acordo com relatos do explorador João de Barros, utilizavam os dedos para carimbar em papéis após o nascimento dos indivíduos para distingui-los uns dos outros (apud. MORAES, 2006).

Um sistema biométrico genérico pode ser descrito conforme mostra-se na Figura 1.

Figura 1 – Sistema Biométrico Genérico

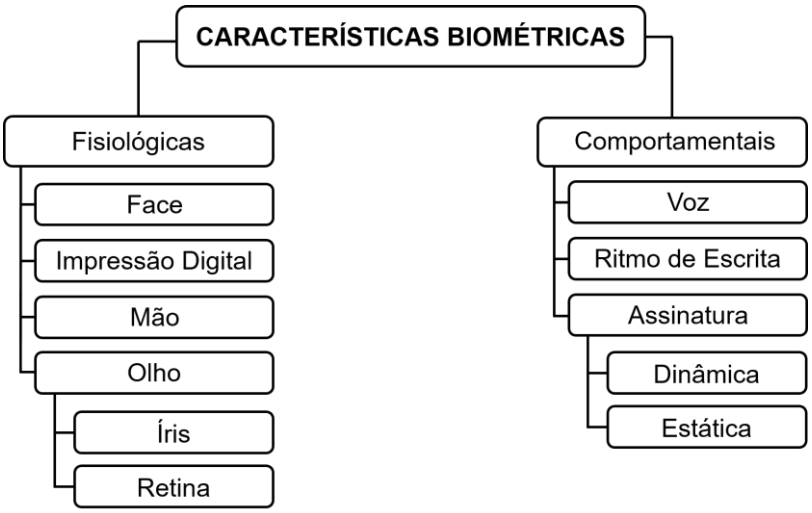


Fonte: PARADA, 2018

Com base na Figura 1 pode-se observar que um sinal é recebido utilizando-se um sensor, como um leitor de impressão digital, uma câmera ou microfone. Este sinal de entrada, para que possa ser compreendido por um sistema computacional, é transformado em um sinal digital, de modo que este seja posteriormente processado a fim de extrair as suas características biométricas. O modelo correspondente ao usuário que está sendo analisado é extraído e armazenado respectivamente. Os parâmetros desse modelo serão utilizados na fase de verificação de novos sinais biométricos de entrada, a fim de verificar a correspondência com a identificação do usuário.

Segundo Costa (2019 apud SRIVASTAVA, 2013) o reconhecimento biométrico caracteriza-se por serem procedimentos e métodos que têm como objetivo fazer a identificação de indivíduos de acordo com suas medidas e o padrão de suas características biológicas, podendo ser divididas, como mostrado na Figura 2, em duas categorias principais:

Figura 2 – Características Biométricas



Fonte: adaptado de Sinfic (2018).

Fisiológicas: características que o indivíduo leva consigo no decorrer de sua vida. São características do corpo humano.

Comportamentais: estas características estão relacionadas ao comportamento do indivíduo, que sofre interferência dos seus sentimentos ou estado psicológico.

De acordo com Galimberti (2018) sistemas biométricos podem ser agrupados em sistemas do tipo invasivo, que exige a colaboração do usuário do sistema para que sejam coletadas as características biométricas, e do tipo não invasivo, que podem ser utilizados sem a necessidade da colaboração do indivíduo ou até mesmo sem que eles saibam que estão sendo analisados. Estes sistemas biométricos podem ser utilizados para verificação, para comprovar a identidade de um indivíduo, e para identificação, onde o indivíduo fornece as características biométricas para ser identificado pelo sistema.

2.3.1 Face

O reconhecimento facial trata-se de uma atividade instantânea executada pelo ser humano. Esta tarefa pode ser executada no dia a dia do indivíduo através do contato visual social ao reconhecer pessoas já vistas de antemão. Trata-se do uso de identificação de padrões em características faciais como as dimensões da boca, da face, do nariz entre outros. Em meio a muitos métodos existentes o

reconhecimento facial se destaca principalmente por sua precisão e simplicidade na execução (COSTA, 2019).

2.3.2 Impressão Digital (fingerprint)

Segundo Bolzan (2019), impressões digitais são resíduos ou marcas de gordura da pele, deixadas pelos dedos em superfícies com os quais estes tiveram contato. As linhas curvas na camada externa da pele na região dos dedos, com determinada concentração de gordura que fica localizada ali, deixam os rastros nos lugares tocados pelos dedos.

2.3.3 Palma da Mão

O sistema que faz uso da identificação pela geometria da palma da mão é baseado nas dimensões de cada parte da mão, como a área, a largura e comprimento dos dedos. O sistema baseado na identificação de um usuário pela geometria das mãos considera a forte ligação entre essas diferentes medidas encontradas na estrutura da mão. Os primeiros sistemas que utilizam a biometria de palma da mão são de 1960 e utilizavam apenas o comprimento de quatro dedos da mão (MARCONDES, 2020).

Uma das principais vantagens da biometria de palma da mão está relacionada ao baixo custo em relação às outras. Seu custo é superior ao da biometria que utiliza a impressão digital contudo a utilização da biometria de palma da mão captura mais informações biométricas, podendo ser considerada mais confiável para utilização como forma de identificação de indivíduo (PEREIRA et al., 2018).

2.3.4 Olho

O uso do olho para obtenção de características biométricas é realizado comumente utilizando a íris, parte que corresponde a esfera colorida do olho, e a retina, que corresponde a parte sensível à luz localizado no fundo do olho (ALECRIM, 2020).

2.3.5 Voz

Como descrito por Hilleshein (2018) cada pessoa possui características que às tornam únicas graças a formação orgânica de todo o sistema responsável pela voz. Como vantagem em relação aos demais tipos de biometria, a voz de cada indivíduo, além das características orgânicas, se diferencia pela forma de falar de cada um ao utilizar as expressões, sotaques, ritmo etc.

2.3.5 Assinatura

De acordo com Marcondes (2020) a assinatura manual de um usuário contém características únicas de cada indivíduo. O reconhecimento de assinaturas divide em sistemas dinâmicos, que consiste em observar como o usuário escreve a sua assinatura, levando em consideração a pressão, aceleração e o número de vezes que a caneta é levantada do papel, e sistemas estáticos, que utilizam imagens de assinaturas e não acompanham o processo de escrita da assinatura, mas se preocupam com a inclinação dos traços, a quantidade de palavras e a razão entre altura e comprimento da assinatura.

2.4 Aplicativos *Mobile*

Dados da companhia *Statista* (2020) – especializada em análises estatísticas para universidades e empresas – apontam que no Brasil havia 136,46 milhões de usuários de smartphones, em projeção até 2025 espera-se que o número aumente para 157,85 milhões.

Segundo Silva *et al.* (2015), no mercado de dispositivos móveis, as ramificações estão em função do grande número de fabricantes diferentes, gerando a diversificação também nas plataformas de desenvolvimento, sistemas operacionais, hardware e software. Múltiplas plataformas ocasionam a necessidade de reescrita dos mesmos aplicativos diversas vezes, sendo codificado diferentemente à cada escrita para funcionamento na arquitetura à qual se destina (apud FERNANDES et. al. 2017).

2.4.1 – Aplicativos Nativos

Em aplicativos nativos são produzidos utilizando a linguagem de programação e as ferramentas da plataforma à qual o aplicativo foi proposto. Caso necessite implementação para outras plataformas, o aplicativo deverá ser produzido de forma separada para cada plataforma específica (MATOS e BRITO, 2016).

Aplicativos nativos, no que diz respeito ao seu desenvolvimento, o reuso de código é quase nulo em razão da diversidade de Sistemas operacionais e diversificação também dos ambientes de desenvolvimento (BIORN-HANSEN; MAJCHRZAK; GRONLI, 2017).

Tratando-se de potencial de uso de funcionalidades, App's Nativos possuem a capacidade de utilizar maior parte dos mecanismos do dispositivo para o qual está sendo desenvolvido, levando em consideração o fato de estar sendo codificado na linguagem nativa do sistema operacional do dispositivo, de maneira que a interação com o hardware deste seja facilitada, conversando de forma direta com o dispositivo. De forma comparativa, os Aplicativos Web Progressivos (PWA) ou Apps Híbridas, possuem maior dificuldade de acesso a todos os mecanismos e funcionalidades de um dispositivo específico, pois a comunicação aplicação – dispositivo não é realizada de forma direta (ANTUNES, 2019).

2.4.2 – Aplicativos Híbridos

Conforme Antunes (2019) aplicativos Híbridos misturam a linguagem nativa do sistema operacional para o qual a aplicação está sendo desenvolvida, com as linguagens WEB, HTML, JavaScript e CSS. O seu diferencial está na facilidade de migrar a aplicação para outra plataforma, sendo que não será necessária uma nova aplicação totalmente desenvolvida para esta nova plataforma, sendo necessário somente mais um módulo da aplicação que cuidará da comunicação com o novo dispositivo.

Aplicações Híbridas, em relação às Nativas, não necessitam de um time grande de desenvolvedores específicos para cada plataforma, sendo que seria utilizado somente um código, focando no desenvolvimento de uma única equipe. O ciclo de desenvolvimento é menor, por trabalhar apenas com linguagens Web,

facilitando a prototipação e desenvolvimento da aplicação, porém a baixa performance deve ser levada em consideração, sendo que aplicações Híbridas possuem menor capacidade de uso do hardware do dispositivo (COLUSSI, 2020).

2.4.3 – Aplicativos Web Progressivos (Progressive Web Apps)

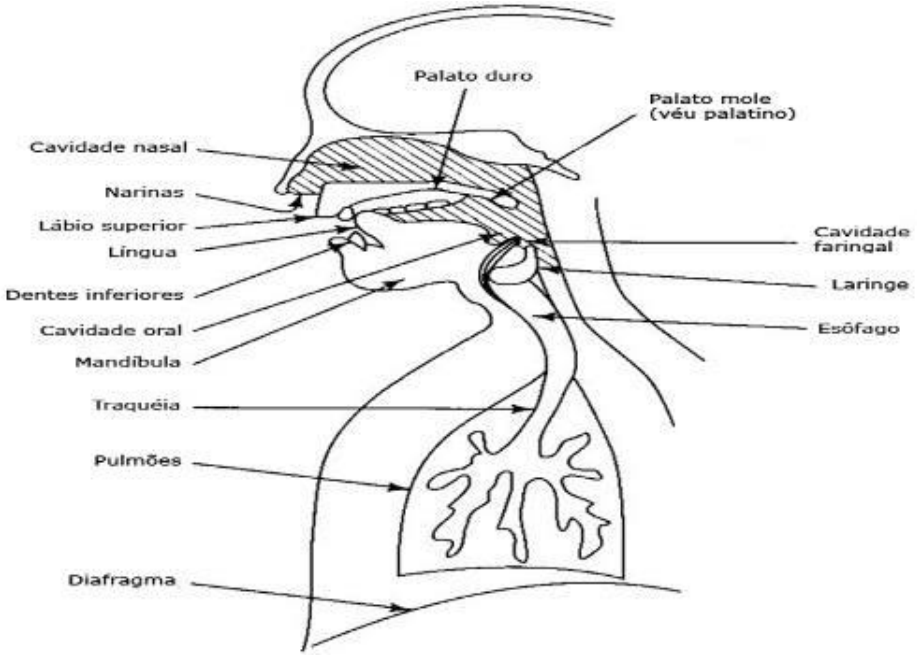
De acordo com Antunes (2019), o conjunto de tecnologias que fazem com que a usabilidade de aplicações WEB se aproximem de uma experiência numa aplicação nativa, é chamado de PWA's – Progressive Web Apps. Usuários com tipos de dispositivo e conexões distintas podem conseguir acessar a aplicação graças ao fato de PWA's permitirem, de maneira fácil, o ícone da aplicação ser adicionado ao menu do dispositivo, sem a necessidade de fazer o *download* e se comportarem, em alguns pontos, como uma aplicação Nativa.

Um PWA é semelhante a um padrão de design computacional que desenvolve experiências de aplicativos utilizando tecnologias de desenvolvimento WEB. Este tipo de aplicação pode ser desenvolvido para diversos sistemas operacionais, como Android, iOS, Windows e muitos outros (FIRTMAN, 2020).

2.5 Reconhecimento de Locutor

Segundo Parada (2018), o ser humano é capaz de reconhecer o indivíduo pela voz graças a formação particular de seu trato vocal, compreendendo a estruturação disposta desde a laringe até os lábios, inclusive as cavidades nasal e oral. Atuando como filtro, este conjunto altera o espectro de frequências da onda de ar que durante a produção da fala vêm dos pulmões, conforme mostra-se na Figura 3.

Figura 3 – Aparelho Fonador



Fonte: Rigonatto, 2017.

Legenda: Estrutura do Aparelho Fonador de um ser humano.

Com o auxílio da Figura 3 pode-se observar as diferentes entidades orgânicas presentes no processo de geração da fala. Para produção da fala o ar sai dos pulmões, é introduzido na traqueia chegando à laringe, onde é modificado pelas cordas vocais, que quando aproximadas vibram conforme a passagem do ar, produzindo sons moldados também pela posição da língua e lábios (UOL, 2020).

Conforme Mafra (2002) de forma geral, um sistema de reconhecimento de locutor comumente pode ser aplicado para identificação do locutor ou verificação do locutor. A Identificação do Locutor consiste em comparar um padrão de locução (características da voz de um indivíduo) com padrões de outras locuções previamente gravadas a fim de decidir qual mais se assemelha com o padrão da locução e locutor corrente. No caso da Verificação do Locutor, refere-se a comparar um padrão de locução de entrada, com o padrão vinculado à identidade proposta para esta locução, a fim de autenticar o Locutor.

Outra maneira de classificar o Reconhecimento de Locutor divide-se em Dependentes de Texto e Independentes de Texto. No sistema de reconhecimento de locutor Dependente de Texto o Locutor deve pronunciar um texto previamente fixado e a qual o sistema foi treinado. No sistema de reconhecimento de Locutor

Independente de Texto é avaliado na Locução características distintas e que não estão vinculadas a um texto específico, ou seja, o usuário poderá falar qualquer frase (MAFRA, 2002).

2.6 *Geofencing*

Geofencing ou Georreferenciamento é o ato de definir um cercado geográfico virtual utilizando-se do Sistema de Posicionamento Global (GPS) para estabelecer ações específicas a usuários de dispositivos (smartphones, por exemplo) que estão localizados dentro da cerca virtual (CRUZ, 2017).

Na Figura 4 mostra-se um exemplo de cerca geográfica.

Figura 4 – Representação de uma *Geofence* (geocerca)

Fonte: CROMBIE, 2020.

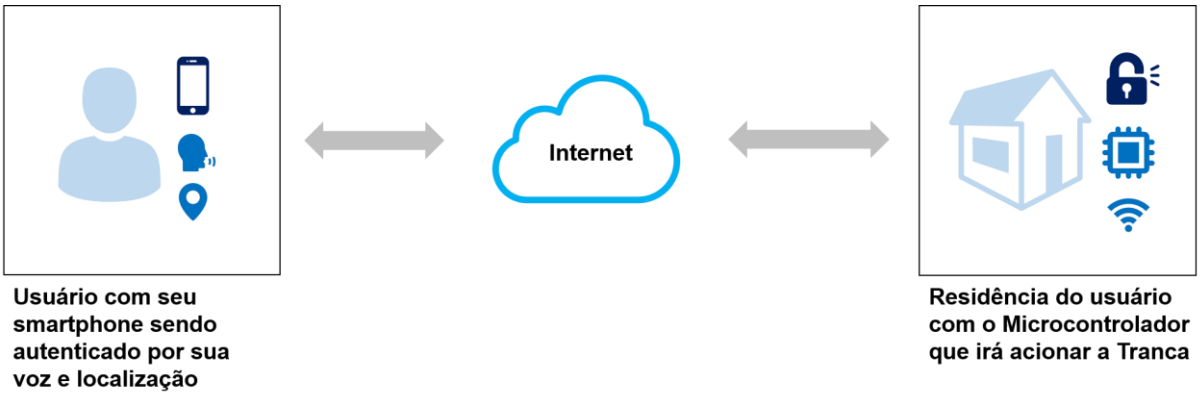
Geofences são utilizadas para demarcar um raio de uma localização específica, podendo ser utilizadas em aplicações de diversos seguimentos inclusive: Marketing, IoT, Rastreamento de Serviços sob demanda, Limitações de uso de serviços ou até mesmo Saúde e Segurança (CROMBIE, 2020).

3 PROPOSTA DE SOLUÇÃO

Neste capítulo são descritas as tecnologias e ferramentas utilizadas no desenvolvimento do projeto a partir do referencial teórico do trabalho.

Para solucionar a questão de pesquisa, é desenvolvido um sistema para o usuário que deseja acionar dispositivos em sua casa com segurança e sem a necessidade de estar no local, através de seu smartphone conectado à internet. Conforme demonstra-se na Figura 5, o usuário utilizando-se de seu smartphone devidamente conectado à internet para, de forma autenticada com o uso de sua voz e localização, realizar o acionamento de uma tranca que está conectada à rede de internet em sua residência.

Figura 5 – Simples Execução da aplicação



Fonte: Elaborado pelo autor.

O sistema desenvolvido é composto por:

- aplicação *Web*;
- API de reconhecimento de voz (Azure);
- API de georreferenciamento (Azure);
- Banco de dados em nuvem (Firebase);
- Microcontrolador (ESP32).

A aplicação será desenvolvida utilizando de tecnologia WEB, de forma que esta poderá ser acessada através de qualquer dispositivo que conceda acesso a: um navegador (Chrome, Firefox, Opera, Edge e outros) que realize requisições HTTP e que consiga obter dados da localização e de gravação de voz.

O usuário da aplicação terá seus dados gravados em banco de dados hospedado em nuvem. Estes dados guardados são referentes as informações pessoais do usuário juntamente com um nome de usuário. O nome de usuário será utilizado para iniciar a autenticação em conjunto aos dados da cerca virtual

cadastrada, os dados referentes a identificação do locutor e posteriormente o acionamento da tranca na residência do usuário.

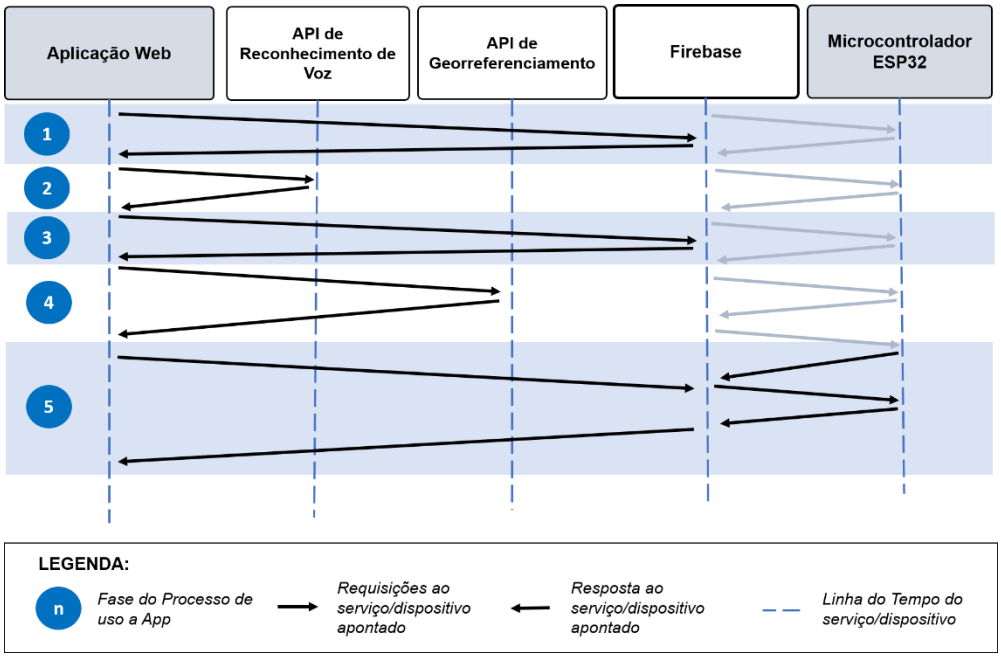
A tranca será conectada a um microcontrolador que terá acesso à Internet e ao banco de dados em nuvem, onde este dispositivo fará verificações a fim de obter o sinal de acionamento da tranca.

Na Figura 6 é ilustrado o processo de autenticação do usuário e acionamento da tranca em sua residência, que pode ser dividido em 5 etapas, sendo elas:

Etapa 1: a App, após obter o *username* (nome de usuário) que é digitado pelo usuário, verifica se existe algum usuário cadastrado com este identificador.

Etapa 2: Se existir usuário cadastrado com o *username* informado, a aplicação fará a gravação de voz do usuário por aproximadamente 10 segundos. Esta gravação será encaminhada à API de Reconhecimento de Locutor, juntamente com o ID de Locutor previamente cadastrado, para comparar as características da voz obtida na gravação atual com as características obtidas anteriormente em fase de cadastro. A reposta da API de Reconhecimento de Locutor é um percentual de aproximação que está no intervalo de 0,0 a 100,0. Quanto menor o percentual, menos provável de ser a voz autêntica do usuário.

Figura 6 – Diagrama de Autenticação do Usuário e Destrave da Tranca



Fonte: elaborado pelo Autor.

Etapa 3: Se o percentual de aproximação da gravação da voz do usuário for maior ou igual a 60,0, a Aplicação solicita os dados referentes a cerca virtual cadastradas ao banco de dados Firebase.

Etapa 4: A aplicação utiliza a API de *Geofencing* para mostrar ao usuário sua posição atual bem como a cerca virtual cadastrada.

Etapa 5: Após a validação da voz e de localização do usuário, realiza o acionamento da tranca, localizada em sua residência, representada por um microcontrolador com acesso ao banco de dados Firebase, que faz a leitura de uma variável que armazenará o estado da tranca.

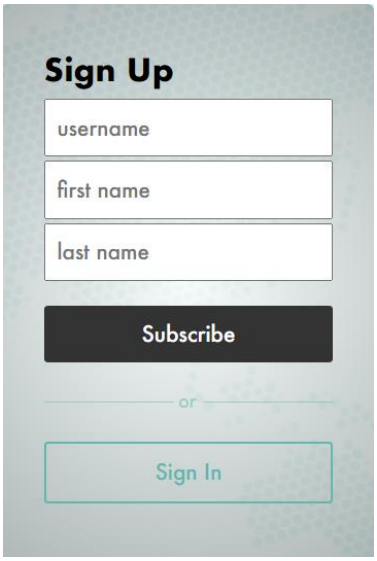
3.1 A Aplicação

A aplicação desenvolvida realiza o interfaceamento do usuário com todo o sistema. Nesta interface, o usuário deverá previamente realizar o cadastro para posteriormente utilizar a aplicação para autenticação e acionamento da tranca.

3.1.1 Cadastro

O usuário realizar o cadastro utilizando-se de: seus dados pessoais, bem como nome de usuário, demonstrado na Figura 7, concessão de uma gravação da voz do usuário em fala por aproximadamente 15 segundos, de sua localização no momento do cadastro e as coordenadas, que serão coletadas a partir de uma cerca virtual desenhada por ele.

Figura 7 – Formulário de Cadastro



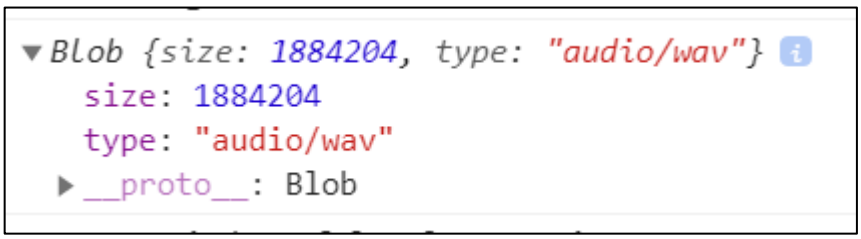
Fonte: Elaborado pelo autor.

Para realizar o cadastro do perfil de Locutor do usuário, utilizando a API da Azure, é necessário obter a gravação da voz do usuário por no mínimo 15 segundos. Essa gravação é armazenada em um BLOB (*Binary Large OBject*).

Um BLOB é, na tradução literal, um objeto binário grande que no JavaScript é utilizado para representação de dados brutos imutáveis como documentos, arquivos de áudio ou vídeo. O BLOB possui também as propriedades *size,* que é um número inteiro que representa o tamanho do arquivo em *bytes* e *type, uma string* que descreve o tipo do arquivo armazenado no BLOB (PAREIN, 2020).

Na aplicação desenvolvida o BLOB utilizado será do tipo “áudio/wav” (mostrado na Figura 8), armazenará um áudio do tipo WAV (*Waveform Audio File Format*), um formato de áudio descomprimido e de alta qualidade, criado pela Microsoft e IBM (FARINACCIO, 2016).

Figura 8 – Captura do Console mostrando o BLOB da gravação da voz do usuário.



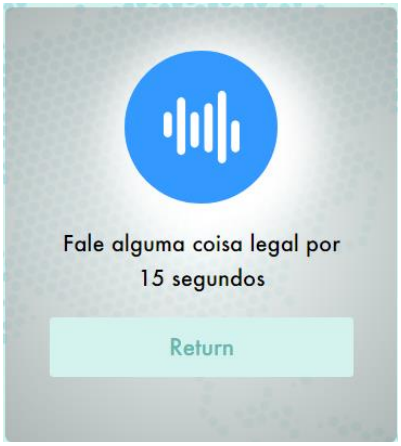
Fonte: Elaborado pelo autor.

Para obter uma gravação da voz do usuário em um BLOB foi utilizado o método *Navigator.getUserMedia()*. Este método é responsável por solicitar a autorização do usuário para uso de 1 (um) dispositivo de entrada de vídeo (como uma câmera) que esteja disponível e/ou 1 (um) dispositivo de entrada de áudio (como um microfone) para capturar a transmissão de mídia. O método *Navigator.getUserMedia()* tem em sua lista de parâmetros: um JSON contendo as permissões que serão solicitadas ao usuário; uma *callback* de sucesso, será utilizada caso possua dispositivo compatível (câmera/microfone) disponível e houver autorização do usuário para uso deste dispositivo de transmissão; e uma *callback* de erro, utilizado caso o usuário não permita a captura da transmissão ou não haja dispositivo compatível. (MOZILA, 2021).

Conforme Gama (2011) um JSON é uma notação de dados para troca de informação entre sistemas.

O método *Navigator.getUserMedia()*, após concessão de permissão de gravação de voz do usuário via navegador, realiza a gravação, apresentada na Figura 9, e retorna ao *callback* de sucesso um objeto do tipo BLOB contendo o áudio gravado, conforme o trecho de código na Figura 10.

Figura 9 – Captura do Momento de Gravação do Áudio de Voz do Usuário



Fonte: Elaborado pelo autor.

Figura 10 – Uso do Método navigator.getUserMedia() para Gravação de Áudio em JavaScript

```
440   navigator.getUserMedia({
441     audio: true
442   }, stream => {
443     onRecorderSuccess(stream, setarAudioBlob, seconds)
444   }, onRecorderError);
```

Fonte: Elaborado pelo autor.

A *callback* de sucesso, *onRecorderSuccess()*, passada por parâmetro para o método *Navigator.getUserMedia()* é responsável por iniciar e encerrar a gravação do áudio da voz do usuário além de guardar esta gravação (BLOB) em um JSON de escopo global, que posteriormente será enviado a API de Reconhecimento de Locutor.

A *callback* de erro *onRecorderError(e)* é responsável apenas por imprimir no console o erro, caso o usuário não dê permissão de utilização de algum dos dispositivos de transmissão de áudio ou vídeo solicitados ou não haja um dispositivo disponível para realizar a captura de áudio ou vídeo.

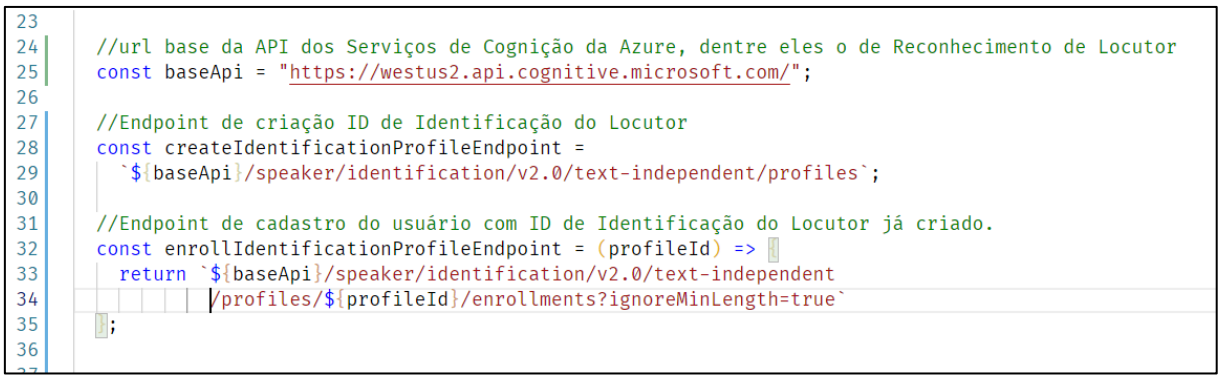
Após a gravação do áudio de voz do usuário, o BLOB é enviado à API de Reconhecimento de Locutor utilizando-se de um *Endpoint* (ponto de extremidade) de criação de perfil de identificação do locutor.

Os *Endpoints,* no contexto de protocolos de comunicação, representam pontos de acesso a um servidor ou API, onde o cliente (navegador, aplicação *mobile*

e outros) consegue obter dados necessário ao usuário através de requisições e respostas (SCHULTZ, 2020).

Ao encaminhar o BLOB da gravação de voz do usuário em uma requisição para a API de Reconhecimento de Locutor, esta retorna em resposta com um ID do Locutor que será armazenado para utilização de reconhecimento posteriormente. As variáveis e funções que fornecem os *Endpoints* utilizados até aqui são declarados conforme mostrado no trecho de código na Figura 11.

Figura 11 – Definição das Variáveis e Funções Responsáveis por Fornecer os Endpoints de Criação e Cadastro de Perfil de Identificação do Locutor.



```
23
24  //url base da API dos Serviços de Cognição da Azure, dentre eles o de Reconhecimento de Locutor
25  const baseApi = "https://westus2.api.cognitive.microsoft.com/";
26
27  //Endpoint de criação ID de Identificação do Locutor
28  const createIdentificationProfileEndpoint =
29    `${baseApi}/speaker/identification/v2.0/text-independent/profiles`;
30
31  //Endpoint de cadastro do usuário com ID de Identificação do Locutor já criado.
32  const enrollIdentificationProfileEndpoint = (profileId) => {
33    return `${baseApi}/speaker/identification/v2.0/text-independent
34                  /profiles/${profileId}/enrollments?ignoreMinLength=true`
35  };
36
```

Fonte: Elaborado pelo autor.

O Perfil de Identificação do Locutor é criado e armazenado pela própria API de Identificação do Locutor. Para criá-lo o serviço (API) coleta as principais características da voz do usuário, gerando uma assinatura de voz exclusiva que será utilizada para a confirmação da identidade deste, posteriormente quando for solicitado. Em futura autenticação do usuário, na aplicação construída, encaminha-se para a API o ID do Perfil de Identificação do Locutor juntamente com uma nova gravação de voz registrada em BLOB para comparação da voz. Em razão disto o ID do Perfil de Identificação do Locutor deve ser armazenado juntamente com os demais dados do usuário que, em fase de registro, foram coletados. Na Figura 12 é mostrado o uso do *Endpoint* da API de Reconhecimento de Locutor para gerar o ID do perfil de identificação, através do método *createProfile* da classe *Speaker* (Locutor) criada para controle de dados do Locutor.

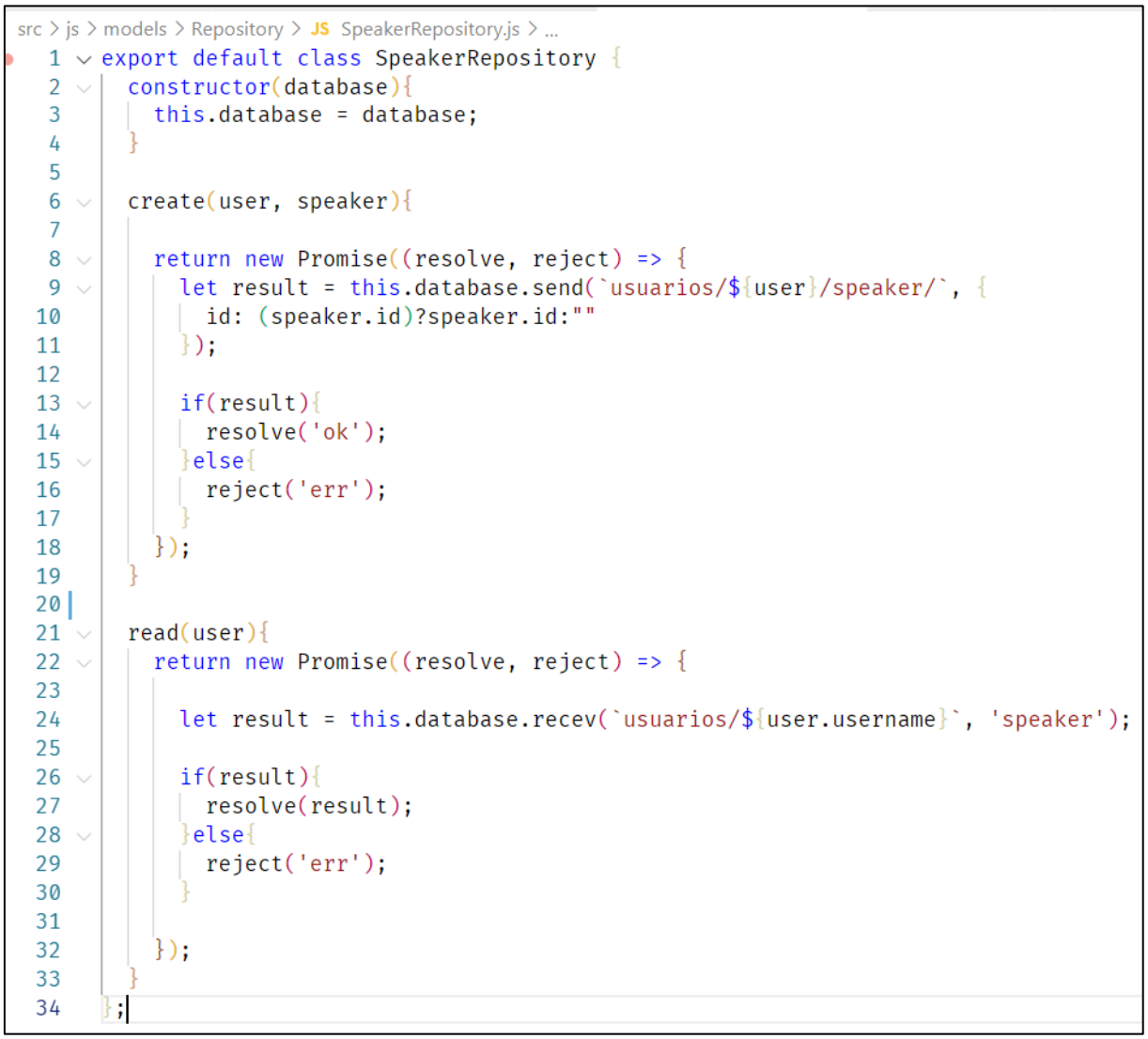
Figura 12 – Método createProfile que utiliza o Endpoint de Criação de Perfil de Identificação do Locutor

```
47
48  //cria o perfil de usuário na API de Reconhecimento de Locutor
49  createProfile(blob){
50
51    //retorna uma Promisse contendo o ID do Perfil de Identificação do Locutor
52    return new Promise((resolve, reject) => {
53
54      let request = new XMLHttpRequest();
55      request.open("POST", createIdentificationProfileEndpoint, true);
56
57      request.setRequestHeader('Content-Type','application/json');
58      request.setRequestHeader('Ocp-Apim-Subscription-Key', key);
59
60      request.onload = function () {
61
62        let json = JSON.parse(request.responseText);
63        console.log(json);
64
65        let profileId = json.profileId;
66
67        //resolve a promisse retornando o ID do Perfil de Identificação do Locutor
68        resolve(profileId);
69      };
70
71      request.send(JSON.stringify({ 'locale' :'en-us'}));
72    });
73  }
74
```

Fonte: Elaborado pelo autor.

Após o ID de Perfil de Identificação de Locutor ter sido criado, este é enviado para o banco de dados utilizando-se do método *create da* classe *SpeakerRepository* que pode ser vista na Figura 13, armazenando-o junto aos dados do usuário anteriormente solicitados.

Figura 13 – Classe SpeakerRepository



Fonte: Elaborado pelo autor.

A classe SpeakerRepository é responsável por realizar a criação e leitura dos dados do locutor de um determinado usuário. Todavia faz-se necessário, no ato de instanciação de um objeto desta classe, encaminhar para o construtor, uma instância da classe *DataBase,* mostrada na Figura 14.

Figura 14 – Classe DataBase

src > js > models > Repository > JS DataBase.js > ...

```
export default class DataBase {

  constructor(data){
    firebase.initializeApp(data);
    this.database = firebase.database();
  }

  async send(where, what){
    return await this.database.ref(where).set(what, err => {
      if(err){
        return err;
      }else{
        return true;
      }
    });
  }

  async receive(where, who){

    return await this.database.ref(where).child(who).get().then( result =>  {
      if (result.exists()) {
        return result.val();
      }
      else {
        console.log("No data available");
        return false;
      }
    }).catch(err => {
      console.error(err);
    });
  }
}
```

Fonte: Elaborado pelo autor.

A classe *DataBase* é instanciada em um único momento da execução da App. No ato da instanciação de um objeto desta classe, faz-se necessário enviar para seu construtor as configurações de autenticação da aplicação em formato JSON de forma que, ainda no construtor da classe, seja inicializado o acesso ao banco de dados, podendo então a aplicação efetuar leitura e escrita em sua base de dados e consequentemente nos dados deste usuário. As informações que devem estar contidas dentro do JSON, que será passado para o construtor, podem ser visualizadas na Figura 15.

Figura 15 – JSON de dados necessários para acesso ao Firebase

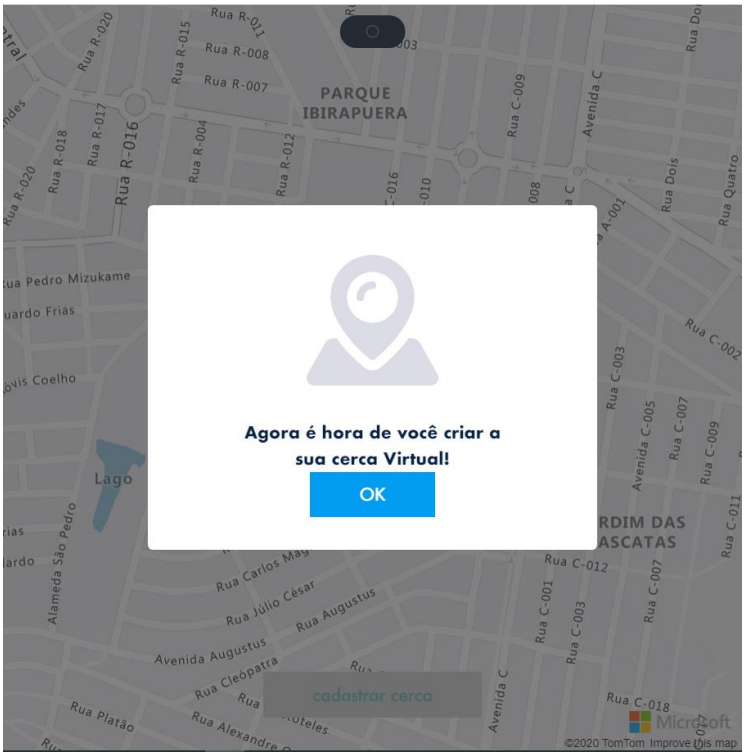
```
99
100   let dbData =  {
101     apiKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
102     authDomain: "XXXXXXXXXXXXXXXXXXXXXXX",
103     projectId: "XXXXXXXXXXX",
104     storageBucket: "XXXXXXXXXXXXX",
105     messagingSenderId: "XXXXXXXXXXXXXX",
106     appId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
107   };
108
```

Fonte: Elaborado pelo autor.

Após a criação e armazenamento do ID do locutor, no banco de dados, o usuário será encaminhado, na App, para a interface de mapa, onde este fará o cadastro da cerca virtual. Esta parte da aplicação tange ao georreferenciamento, onde será obtido os dados da localização do usuário e a cerca que será cadastrada.

Na Figura 16 é mostrado uma mensagem informando que na fase atual da execução da aplicação o usuário fará o cadastro da cerca virtual.

Figura 16 – Modal de Apresentação da Interface de Criação da Cerva Virtual



Fonte: Captura da App, realizada pelo autor.

Clicando no botão “OK” a aplicação irá mostrar o mapa, com a marcação da localização atual do usuário, que poderá criar a cerca virtual, como mostra-se na Figura 17.

Figura 17 – Renderização do Mapa na Aplicação

Fonte: Captura da Aplicação, realizada pelo autor.

Para renderização do mapa e uso de suas propriedades, foram utilizados dois pacotes JavaScript da Azure Maps: *azure-maps-control*, responsável por realizar a renderização e o controle do mapa; e o *azure-maps-drawing-tool*, responsável pelas ferramentas de desenho do mapa. Estes pacotes foram instalados utilizando NPM (*Node Package Managment*), um gerenciador de pacotes Node.js que auxilia no desenvolvimento JavaScript.

O Node.js pode ser descrito como um ambiente de execução JavaScript *server-side* (lado do servidor), possibilitando a criação de aplicações JavaScript sem depender do navegador (LENON, 2018).

Conforme Longen (2019), NPM é um gerenciador de pacotes do Node que é utilizado pelos desenvolvedores que programam em JavaScript, no compartilhamento de ferramentas, módulos e dependências.

Utilizando-se dos pacotes, criou-se uma classe denominada Maps, responsável por controlar as atividades do mapa na App, gerenciando então as propriedades dos pacotes da Azure Maps utilizados. Entre muitos papéis desta classe encontra-se o principal, a criação do mapa na App. Para realizá-lo, o construtor espera receber em sua lista de parâmetros um JSON contendo as configurações do mapa, conforme mostra-se na Figura 18.

Figura 18 – Trecho de Código responsável por definir as configurações do Mapa

```
1
2   navigator.geolocation.getCurrentPosition(local => {
3     mapsController({
4       /*encaminhas as coordenadas atuais do usuário*/
5       longitude: local.coords.longitude,
6       latitude: local.coords.latitude,
7
8       /*Identificador do Elemento que irá renderizar o Mapa no HTML*/
9       idElementMap: 'map',
10
11      /*Opções do Marcador da Localização atual do usuário*/
12      markerOptions: {
13        htmlContent: '<div><div class="pin bounce"></div><div class="pulse"></div></div>',
14        color: 'DodgerBlue',
15        text: 'o',
16        position: [local.coords.longitude, local.coords.latitude]
17      },
18
19      /*Propriedades Gerais do Mapa*/
20      properties: {
21        center: [local.coords.longitude, local.coords.latitude],
22        zoom: 15,
23        view: "Auto",
24        authOptions: {
25          authType: 'subscriptionKey',
26
27          /*Dados para Autenticação da App na Azure Maps*/
28          subscriptionKey: 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
29          clientId: 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX',
30
31          /*Obtendo o Token de Acesso a Azure Maps*/
32          getToken: function (resolve, reject, map) {
33              var tokenServiceUrl = "https://azuremapscodesamples.azurewebsites.net/Common/TokenService.ashx";
34              fetch(tokenServiceUrl).then(r => r.text()).then(token => resolve(token));
35          }
36        }
37    }})
38  });
39
```

Fonte: elaborada pelo Autor.

Figura 19 – Trecho de Código que insere no HTML o elemento que irá renderizar o Mapa

```
181
182     document.body.innerHTML = `<div id="map"></div>
183                                  <main>
184                                  <div class="buttons-controll-fence"></div>
185                                </main>`;
186
```

Fonte: elaborada pelo Autor.

Para obter a localização atual do usuário utilizou-se o método da API de Geolocalização *navigator.geolocation.getCurrentPosition()*, que captura a localização atual do usuário sob sua permissão, devolvendo-a no parâmetro de uma *callback* de sucesso em um objeto Position, contendo as coordenadas (FARIA, 2021).

Como mostrado na figura 18, as coordenadas são acessadas utilizando os métodos: *coords.longitude*, que fornece a longitude; e *coords.latitude*, que fornece a latitude.

Figura 20 – Desenho da Cerca no Mapa

Fonte: elaborada pelo Autor.

Para criar e gerenciar as propriedades do mapa é criado uma instância da classe Maps. Esta classe possui, além da função de criar o mapa e gerenciar os recursos da Azure Maps que são utilizados na App, cumpre com o papel de habilitar o desenho da cerca virtual, através de uma chamada ao método *buildToolbar()*, apresentado na Figura 21.

Figura 21 – Método buildToobar() da classe Maps

```
buildToolbar(){
  /*
    Executa a criação da barra de ferramentas
    de desenho assim que o mapa estiver carregado
  */
  this.map.events.add('ready', () => {

    /*Habilita o desenho no mapa para captura da cerca virtual*/

    let drawingManager = new atlasDraw.drawing.DrawingManager(this.map, {
        toolbar: new atlasDraw.control.DrawingToolbar({
          position: 'non-fixed',
          style: 'dark',
          buttons: ['draw-circle']
        })
    });

    /*Habilita o Botão de Cadastrar Cerca quando o desenho é terminado*/

    this.map.events.add('drawingcomplete', drawingManager, ()=> {

      this.fenceSubscribeButton.disabled = false;

      this.fenceSubscribeButton.addEventListener('click', ()=>{
        document.querySelector('#modalConfirmSubscribeFence').classList.remove('hidden');
      })
      this.measureShape;
    });

  });
}
```

Fonte: elaborada pelo Autor.

Quando é realizada a chamada do método *buildToolbar()*, é adicionado ao evento *ready* (pronto) a criação do *drawingManager*, o objeto que gerencia os desenhos no mapa renderizado é instanciado pela classe DrawingManager, presente no pacote *azure-maps-drawing-tools*. Dentro do método *buildToolbar()*, ainda, é adicionado ao evento *drawingComplete*, que ocorre quando o usuário completa o desenho da cerca virtual, o botão “CADASTRAR CERCA” é habilitado, permitindo ao usuário, então, salvar a cerca desenhada. Ao completar o desenho da cerca virtual, no evento *drawingComplete*, é realizada a chamada ao método *measureShape()*, que irá calcular as propriedades raio, área e perímetro, como mostra-se no trecho de código na Figura 22.

Figura 22 – Método measureShape() da classe Maps

```
35  measureShape() {
36
37  this.fence.coordinates = {};
38
39  /*Obtém as coordenadas da cerca virtual*/
40
41  let result = JSON.parse(JSON.stringify(drawingManager.getSource().toJson()));
42
43  this.fence.coordinates = result.features[0]["geometry"]["coordinates"];
44
45  this.fence.type = 'circle';
46
47  /*
48    Calcula as propriedades da Cerca Virtual Desenhada
49    - RAIO em quilômetros
50    - AREA em quilômetros quadrados
51    - PERÍMETRO em quilômetros
52  */
53
54  this.fence.radius = atlas.math.convertDistance(shape.getProperties().radius, 'meters', 'kilometers', 5);
55  this.fence.area = Math.round(2 * Math.PI * fence.radius * fence.radius * 100) / 100;
56  this.fence.perimetro = Math.round(2 * Math.PI * fence.radius * 100) / 100;
57
58  }
```

Fonte: elaborada pelo Autor.

O método *measureShape()*, utiliza a função *math.convertDistance*(), presente no pacote *azure-maps-control*, para converter o Raio da cerca virtual desenhada de quilômetros para metros obtendo 5 casas decimais após a vírgula. As coordenadas e as demais propriedades da cerca estão contidas dentro de um objeto da classe Fence (cerca). Os dados de um objeto da classe Fence, mostrada na Figura 23, são persistidos no banco de dados utilizando-se da classe FenceRepository.

Figura 23 – Classe Fence

```
src > js > models > JS Fence.js > Fence > onFenceArea
1   import * as atlas from 'azure-maps-control';
2
3   export default class Fence{
4     constructor(fence){
5       this.type = fence.type;
6       this.perimeter = fence.perimetro,
7       this.area = fence.area,
8       this.coordinates = fence.coordinates;
9       this.radius = fence.radius;
10    }
11    /*
12      Método responsável por verificar se o usuário está dentro da cerca cadastrada
13      Primeiramente calcula-se a distância entre o centro
14      da circunferência da cerca e a localização atual
15      depois é comparado a distância obtida com o raio da cerca
16      Se esta distância for menor ou igual, o usuário está dentro da cerca
17      Se for menor, o usuário está fora da cerca
18    */
19    onFenceArea({lat, lng}) {
20
21      let distance = atlas.math.getDistanceTo([lng, lat], [this.coordinates[0], this.coordinates[1]], "kilometers");
22      console.log('Distancia calculada: ' + distance);
23
24      return this.radius*1000 <= distance;
25    }
26  }
27
```

Fonte: elaborada pelo Autor.

O método *onFenceArea()* da classe *Fence*, mostrada na Figura 23, é responsável por dizer se o usuário está ou não dentro da cerca virtual, no ato da chamada do método, utilizando-se da função *math.getDistanceTo()* presente no pacote *azure-maps-control*, que faz uso da fórmula matemática de Haversine.

Fórmula 1 – Haversine

$$d = 2 * r * arc\sin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\varphi_1 * \cos\varphi_2 * \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (1)$$

Fonte: VENESS, 2021.

Na Fórmula 1, $\boldsymbol{\varphi_1}$ e $\boldsymbol{\varphi_2}$ representam respectivamente as latitudes do primeiro ponto e do segundo ponto em radianos. De forma semelhante $\boldsymbol{\lambda_1}$ e $\boldsymbol{\lambda_2}$ representam as longitudes do primeiro ponto e do segundo ponto, respectivamente, em radianos. O $\boldsymbol{r}$ representa o raio da esfera, neste caso, a Terra, considerado como 6.371 quilômetros por não ser uma esfera perfeita. A variável $\boldsymbol{d}$ recebe a distância entre o primeiro e o segundo ponto, conforme a unidade de medida utilizada na Fórmula 1, resultado do cálculo de Harversine (VENESS, 2021).

3.1.2 Autenticação

Posterior à etapa de cadastro está a autenticação do usuário. Quando desejar, este poderá fazer acesso a aplicação para acionamento da tranca localizada em sua residência. Para realizar tal atividade o usuário, primeiramente, deverá lembrar o seu *username* (nome de usuário), criado na fase de cadastro, conforme mostra-se na Figura 24.

Figura 24 – Interface de Login da Aplicação

Fonte: elaborada pelo Autor.

Com o *username* digitado, o usuário deverá clicar no botão "*Sign In*" para que a aplicação inicie a autenticação. Nesta fase a aplicação irá verificar no banco de dados se existe um usuário cadastro com este *username*. Se sim, a aplicação irá então buscar o ID de Locutor, gerado na fase de cadastro. Esta informação será utilizada quando a gravação da voz do usuário, mostrada na Figura 25, ter sido concluída.

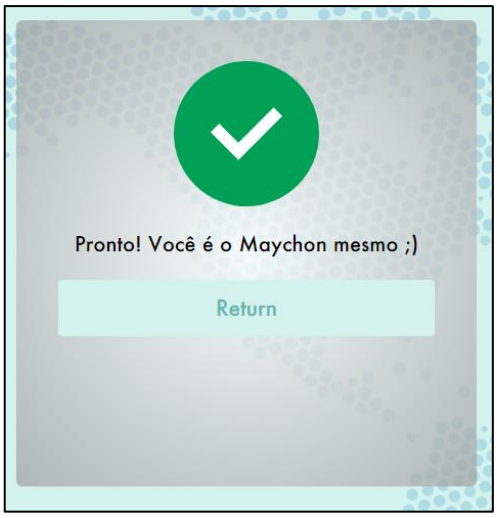
Figura 25 – Gravação da Voz do Usuário para Autenticação

Fonte: elaborada pelo Autor.

Após a gravação ter sido finalizada, a aplicação fará uma requisição a API de Reconhecimento de Locutor para verificar se esta voz é a mesma gravada na fase

de cadastro. Para isso será necessário encaminhar nesta requisição o ID do Locutor e esta nova gravação de voz em um BLOB. A API irá retornar, em percentagem de 0 a 100, se a voz da nova gravação corresponde a voz cadastrada. Se o percentual for maior ou igual a 60, então o usuário será autenticado, conforme mostra-se na Figura 26, e logo será encaminhado para a tela do mapa, onde será verificado se ele está dentro da cerca virtual cadastrada.

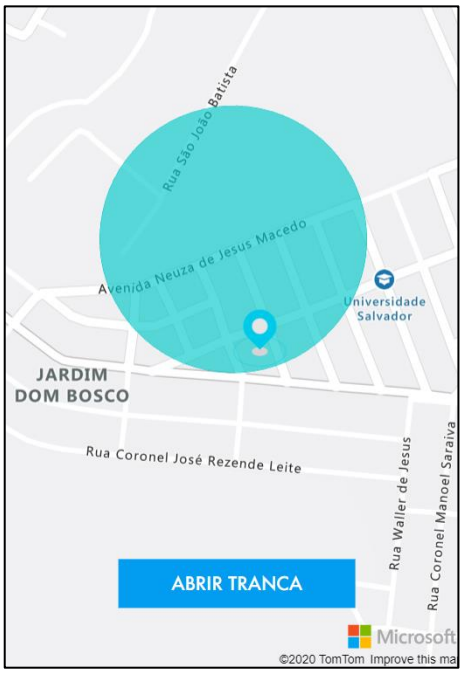
Figura 26 – Voz reconhecida pela App.



Fonte: elaborada pelo Autor.

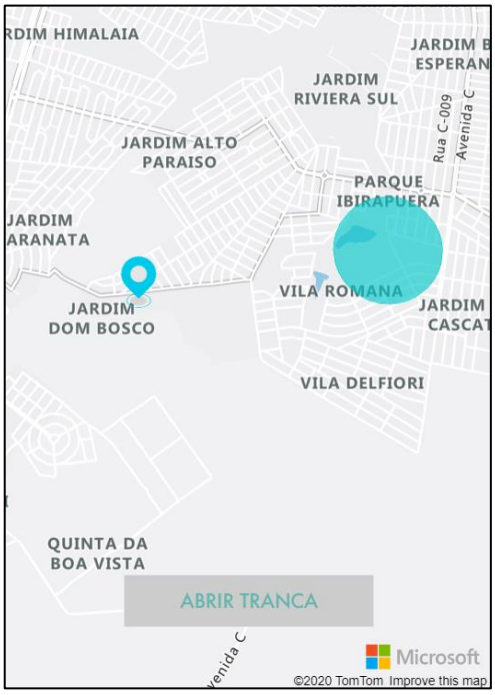
Após a validação por voz, o usuário terá sua localização validada em relação à cerca virtual cadastrada. Se o usuário estiver localizado dentro da cerca virtual cadastrada, o botão “DESTRAVAR TRANCA” estará habilitado, conforme a Figura 27. De forma contrária, se o usuário estiver fora da cerca, o botão estará desabilitado, conforme a Figura 28, impossibilitando-o de realizar o acionamento da tranca em sua residência.

Figura 27 – Usuário Dentro da Cerca – Acionamento da Tranca Habilitado



Fonte: elaborada pelo Autor.

Figura 28 – Usuário Fora da Cerca – Acionamento da Tranca Desabilitado.



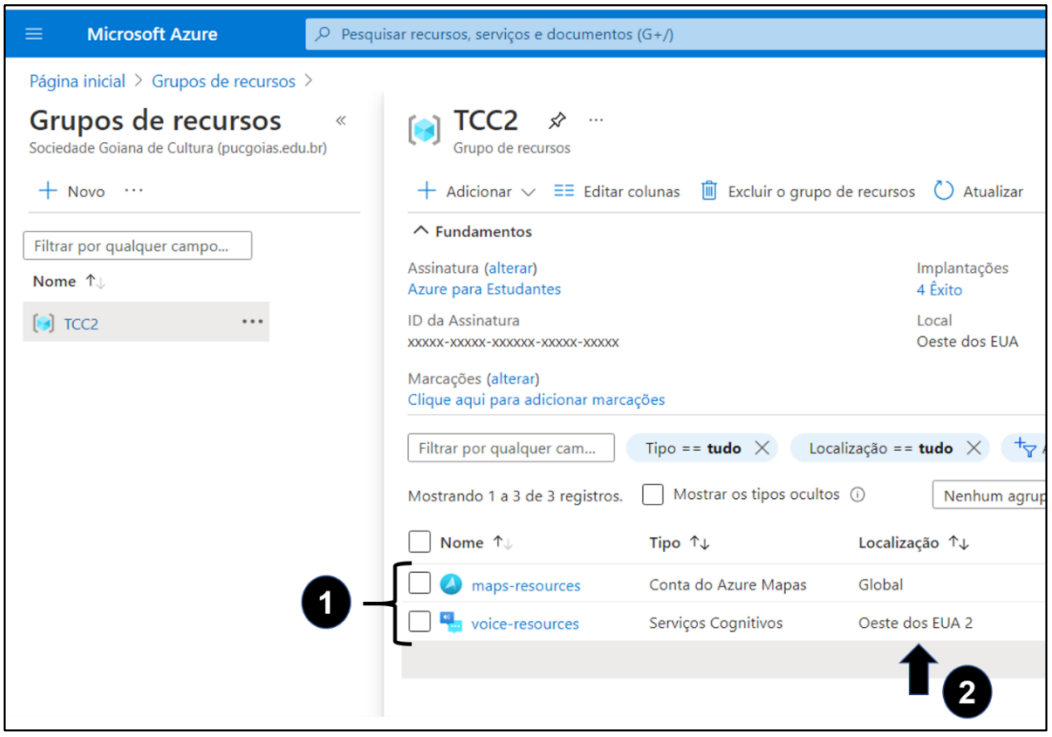
Fonte: elaborada pelo Autor.

3.1.3 Microsoft Azure

O reconhecimento de Locutor e os recursos de mapas para georreferenciamento, utilizados neste trabalho, são fornecidos pela Azure. Criada em 2010 pela Microsoft, a Azure é uma plataforma de nuvem composta por produtos e serviços que auxiliam o desenvolvedor a projetar soluções bem como empresas na utilização de serviços em nuvem (GARRETT, 2020).

Para usufruir dos recursos da Azure, para uso na implementação desta App, utilizou-se da conta acadêmica da Microsoft 365 fornecida pela Pontifícia Universidade Católica de Goiás (PUC Goiás) a todos os acadêmicos regularmente matriculados. Com esta conta é possível realizar o cadastro na Azure para uso gratuito (por tempo determinado) das aplicações e serviços oferecidos pela plataforma. Após realizar o cadastro no site da Azure (azure.com) foi criado, no Portal de Serviços (portal.azure.com), um grupo de recursos, que irá conter os serviços da Azure que são utilizados pela aplicação desenvolvida: Serviços Cognitivo (reconhecimento de locutor) e Azure Maps (georreferenciamento), como pode ser observado na Figura 29 na marcação de número 1.

Figura 29 – Captura da Tela do Grupo de Recursos Azure



Fonte: Interface Azure adaptada pelo Autor.

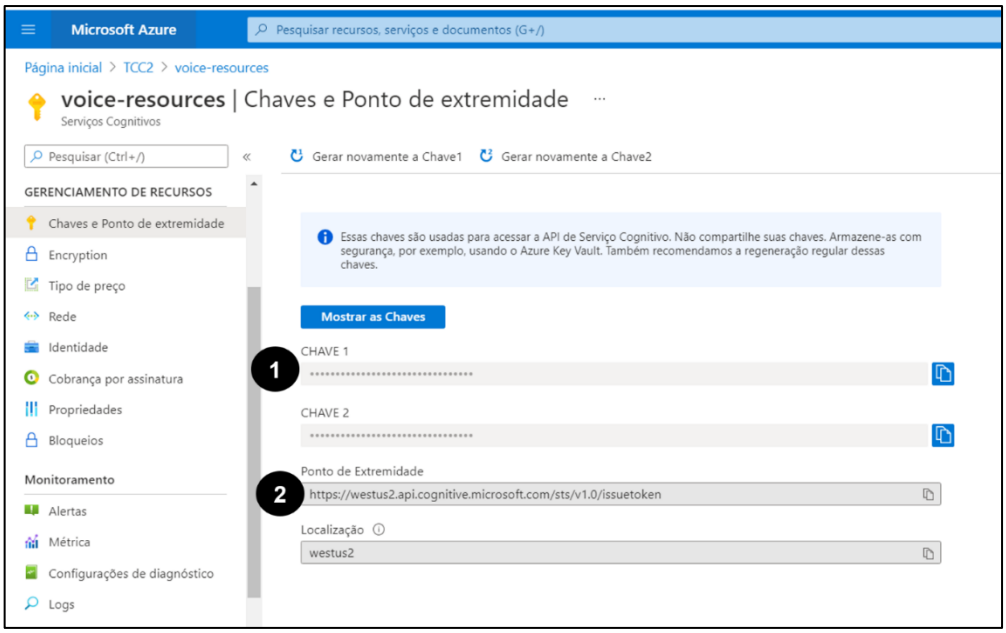
Na Figura 29 na marcação de número 2, observa-se que a localização do recurso referente a Serviços Cognitivos (reconhecimento de locutor) está nos servidores do Oeste dos Estados Unidos. O reconhecimento de Locutor está disponível apenas para as regiões do Oeste dos Estados Unidos no período em que a aplicação foi desenvolvida (HILLEBRAND, 2020).

3.1.3.1 Reconhecimento de Locutor

O Reconhecimento de Locutor é um dos serviços de cognição oferecidos pela Azure, que concede acesso a algoritmos que verificam e identificam os locutores de acordo com as características de voz, utilizando-se da biometria da voz (TREVORBYE; OLPROD, 2020).

Na presente aplicação o Reconhecimento de Locutor é utilizado para verificar se o usuário que está falando é de fato quem ele diz ser, através dos dados salvos em banco de dados na nuvem. Para utilização do recurso de Reconhecimento de Locutor, cria-se uma instância de serviços cognitivos dentro do Portal Azure, de forma que, esta instância possibilite o acesso da aplicação ao serviço com o uso de uma chave de autenticação, como mostra-se na Figura 30.

Figura 30 – Captura da Tela do Recurso de Voz Azure



Fonte: Interface Azure adaptada pelo Autor.

Na marcação de número 1 da Figura 30 observa-se as chaves de acesso ao recurso que, após criado, são liberadas para a aplicação poder realizar requisições e utilizar a API de Serviços Cognitivos. Uma chave apenas é necessária para fazer chamadas à API. As chamadas são realizadas utilizando-se do *Endpoint* mostrado na marcação de número 2 na Figura 30.

Quadro 1 – Precificação da API de Reconhecimento de Locutor Azure

INSTÂNCIA	CATEGORIA	RECURSO	PREÇO
Gratuito - web/Contêiner 1 solicitação simultânea	Reconhecimento do Locutor	Identificação do Locutor	10.000 transações gratuitas por mês
		Verificação do Locutor	10.000 transações gratuitas por mês
Padrão - Web/Contêiner 20 solicitações simultânea 1	Reconhecimento do Locutor	Verificação do Locutor	R$ 20,246 por 1.000 transações
		Identificação do Locutor	R$ 40,492 por 1.000 transações

Fonte: AZURE, 2021; adaptada pelo Autor.

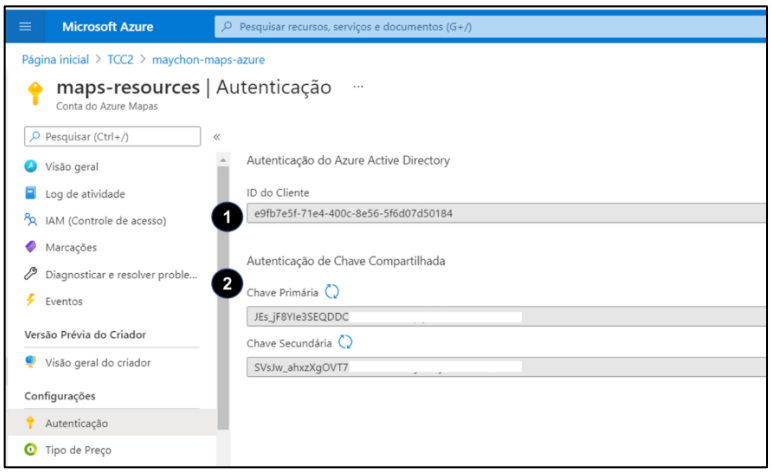
O Quadro 1 apresenta os preços de utilização do Reconhecimento de Locutor da Azure. No desenvolvimento da presente aplicação foi utilizado a cota gratuita.

3.1.3.2 *Geofencing*

Nesta aplicação utilizou-se dos serviços da API Maps da Azure, para renderização e visualização de mapa dinâmico com interação do usuário, fornecendo o georreferenciamento necessário para a validação do usuário na cerca geográfica.

Assim como o recurso de reconhecimento de locutor, o serviço de Maps da Azure necessita da utilização de credenciais de autenticação da App, mostradas na Figura 31, que são liberadas após criação do recurso na Azure Maps .

Figura 31 – Captura da Tela do Recurso de Maps da Azure



Fonte: Interface Azure adaptada pelo Autor.

A marcação de número 1 (um), na Figura 31, mostra-se o ID do cliente da conta Azure Maps. Na marcação 2 são mostradas as chaves de autenticação da App, utilizadas para realizar o acesso aos recursos do Azure Maps.

Quadro 2 – Precificação Azure Maps por 1.000 transações

		Grátis	Camada 0
	Metros		<100K
Mapas	Ladrilhos do Mapa Básico	5.000	**R $ 22,05**
	Blocos de imagens *	1.000	
	Traffic Tiles	5.000	**R $ 22,05**
	Telhas do tempo	1.000	**R $ 22,05**
	Imagens de mapa estático	1.000	**R $ 22,05**
Informações de localização	Dados	5.000	**R $ 22,05**
	Geolocalização	5.000	**R $ 22,05**
	*Elevação**	1.000	
	Procurar	5.000	**R $ 22,05**
	Encaminhamento	1.000	**R $ 22,05**
	*Mobilidade**	1.000	
	Cálculos Espaciais	5.000	**R $ 22,05**
	Fuso horário	5.000	**R $ 22,05**
	Tráfego	1.000	**R $ 22,05**
	Clima	1.000	**R $ 22,05**

Fonte: AZURE MAPS, 2021, adaptada pelo Autor.

Na presente aplicação desenvolvida foi utilizado a cota gratuita de recursos da Azure Maps, mostrada no Quadro 2.

3.1.4 Progressive Web App

Para que a aplicação seja passível de instalação num dispositivo (Android, iOS, Windows e outros) é necessário transformá-la em um PWA, *Progressive Web App*, um tipo de aplicação versátil, multiplataforma. Para transformar a aplicação desenvolvida em um PWA, realizou-se a instalação de dois *plug-ins: webpack-pwa-manifest* e *workbox-webpack-plugin*. Estes plugins são responsáveis por disponibilizar a aplicação como um PWA. Para instalação dos mencionados, é necessário a utilização do WebPack, um empacotador de módulos para aplicações JavaScript (WEBPACK, 2021).

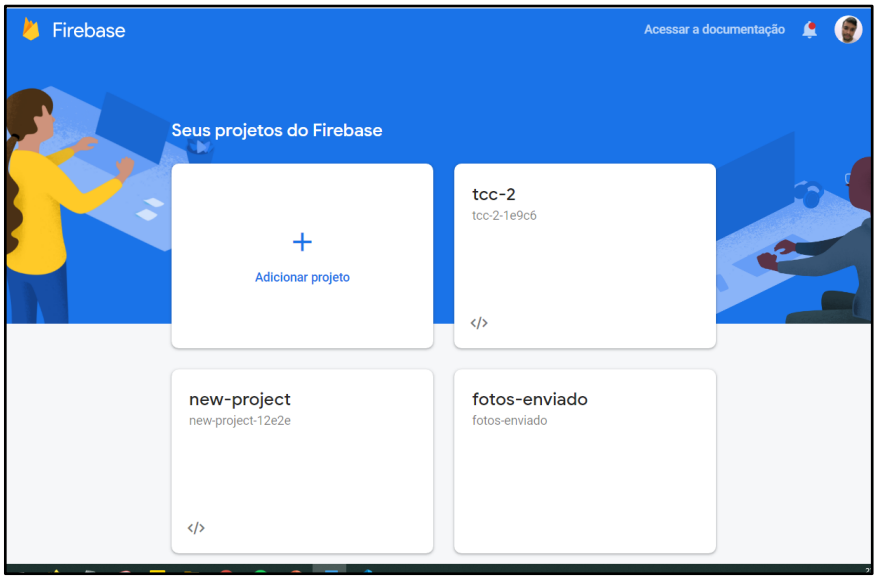
3.2 Banco de Dados Firebase

Para obter as informações necessárias para acesso a um banco de dados no Firebase, mostrados na Figura 32, é necessário realizar os seguintes passos:

1. Obter uma conta do Google para acessar a plataforma Firebase;
2. Acessar o Console Firebase;
3. Criar um Projeto, como mostrado na Figura 14;
4. Criar um Banco de Dados dentro deste Projeto;

Após a criação do banco de dados, será necessário obter as chaves de acesso utilizadas pela App, com o auxílio da documentação Firebase.

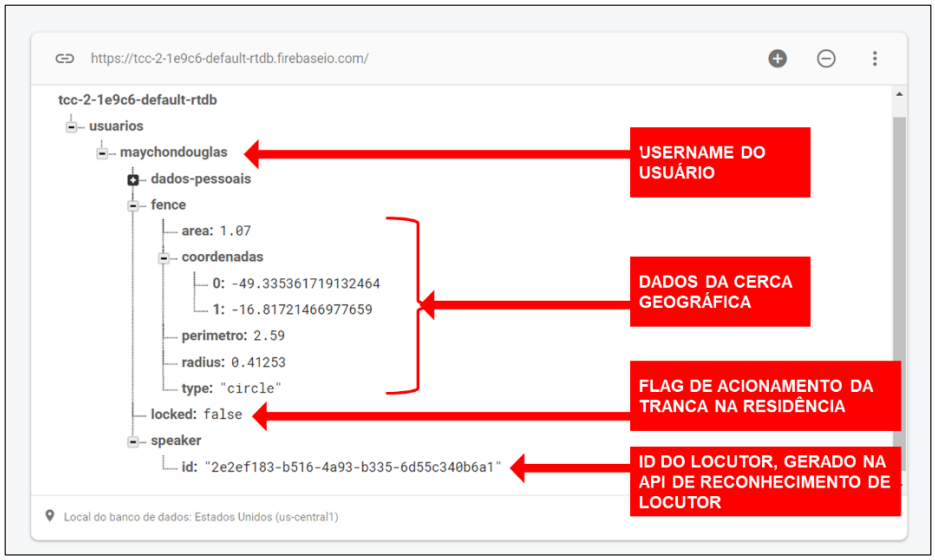
Figura 32 – Captura da Tela de Projetos Firebase



Fonte: Interface Firebase.

O banco de dados final resultante da App, que guarda algumas informações pessoais, dados da cerca geográfica, o ID de Locutor e contém a *flag* de acionamento da tranca, pode ser observado na Figura 33.

Figura 33 – Captura da do Banco de Dados criado no Firebase



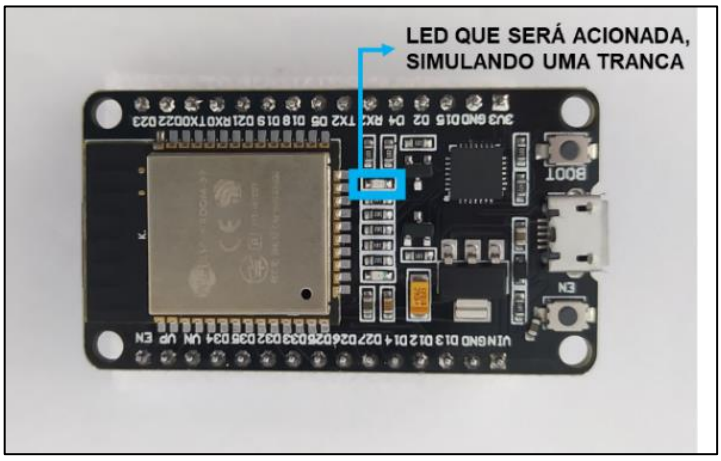
Fonte: Interface Firebase.

3.3 Sistema em Execução no Microcontrolador

Conforme Themes (2021), um microcontrolador é um computador de um único chip, que pode ser utilizado para diversas aplicações, podendo ser programado e integrado a outros periféricos. Microcontroladores apesar de seu alto número de funcionalidades possui limitação de recursos como memória e frequência de *clock* (relógio). Desta forma suas principais vantagens estão no tamanho, por ser geralmente pequeno, e no preço, dispositivos de baixo custo.

No presente trabalho utiliza-se um Módulo ESP32 NodeMCU, uma plataforma que contém um microcontrolador ESP32 e que pode ser utilizada em implementações de projetos IoT. Na Figura 34 mostra-se o Módulo ESP32 NodeMCU com a marcação da LED que simula o acionamento de uma tranca, sendo que a LED desligada representa a tranca está fechada, de modo contrário, a LED ligada representa a tranca aberta.

Figura 34 – Fotografia do Módulo ESP32 NodeMCU IoT WiFi e Bluetooth com marcação de LED que simula a tranca



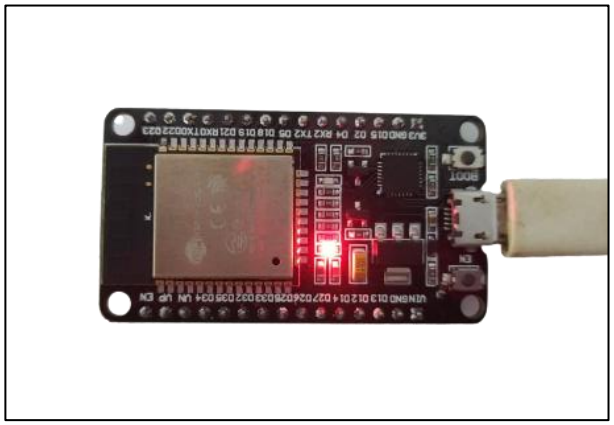
Fonte: Fotografia tirada pelo Autor.

Para programar o Módulo ESP32 NodeMCU utilizou-se da Arduino IDE 1.8.10, um ambiente de desenvolvimento integrado geralmente utilizado para programação do Arduino, mas que pode ser utilizado para programar outros Microcontroladores. Para implementar o programa fez-se necessário a utilização de duas bibliotecas, sendo elas: WiFi.h, responsável pela conexão à rede WiFi da

residência, e FirebaseESP32.h, responsável pela conexão com o banco de dados Firebase.

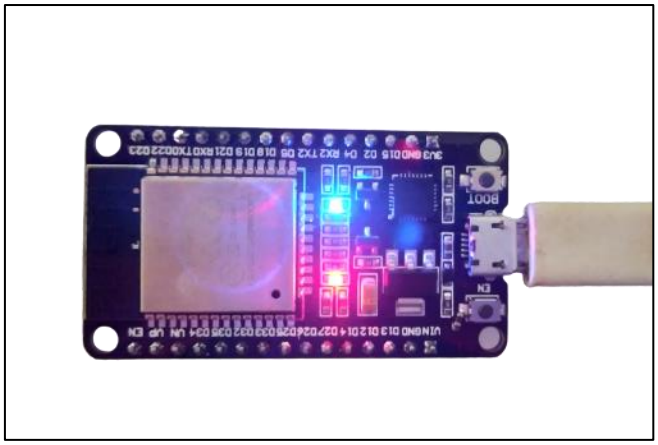
O algoritmo realizado pelo módulo ESP32 NodeMCU é responsável por realizar consultas no banco de dados em nuvem para verificar se o estado da *flag* de acionamento da tranca, *locked*, é *true* (verdadeiro) para fechado, ou *false* (falso) para aberto. Se fechado, o microcontrolador irá desligar a LED ou simplesmente mantê-la apagada, conforme mostra-se na Figura 35. De forma contrária, se aberto, o microcontrolador ligará a LED simulando, então, uma tranca aberta, registrado na Figura 36.

Figura 35 – Fotografia do Módulo ESP32 NodeMCU IoT WiFi e Bluetooth simulando a tranca fechada, LED apagado



Fonte: Fotografia tirada pelo Autor.

Figura 36 – Fotografia do Módulo ESP32 NodeMCU IoT WiFi e Bluetooth simulando a tranca aberta, LED aceso



Fonte: Fotografia tirada pelo Autor.

4 CONSIDERAÇÕES FINAIS

O presente trabalho objetivou o desenvolvimento de aplicação Web, que pode ser acessada pelo usuário autenticado, através da biometria de voz, e sua localização, em relação a uma cerca virtual. Utilizando-se desta aplicação o usuário realiza o acionamento de uma tranca localizada em sua residência. Para realizar a autenticação do usuário utilizando sua voz e disponibilizar o mapa para construção da cerca virtual utilizou-se de recursos da Azure, apresentando funcionamento adequado para atender aquilo que se propôs. A documentação disponibilizada pela Azure forneceu todo suporte para construção da aplicação e integração com os serviços utilizados, além de fornecer acesso gratuito às ferramentas para pequenos projetos, facilitando, então, a implementação desta solução.

A utilização do banco de dados Firebase, de forma positiva, fez toda a diferença na solução desenvolvida. A documentação Firebase juntamente com a biblioteca utilizada no microcontrolador fez-se suficiente para a integração, atendendo as necessidades desta solução. Agindo como intermediário entre a aplicação Web e o microcontrolador ESP32, o banco de dados Firebase recebia solicitações das aplicações persistindo informações necessárias para o funcionamento da solução, estabelecendo, então, a comunicação entre ambas.

A integração entre plataformas e dispositivos em conexão com serviços em nuvem foi fonte de grande conhecimento além da aplicação de conceitos de cerca virtual, possibilitando a utilização destas em outras soluções.

Esta pesquisa necessita, ainda, de realizar avaliação dos níveis de segurança da ferramenta de reconhecimento de voz utilizada. No entanto pode se destacar que, durante os testes da aplicação, o reconhecimento de locutor obteve assertividade quando se mantêm a fala em tom semelhante à cadastrada, sendo que os resultados podem sofrer influência dos níveis de ruídos presentes no ambiente e da qualidade do dispositivo utilizado para captura de som.

A construção de aplicações com uso de cercas virtuais mostra-se uma solução viável, visto que os principais recursos necessários estão disponíveis nos dispositivos móveis dos próprios usuários, através do uso de dados móveis, para conexão à internet, e uso de sinal GPS, para captura da localização do usuário.

Os códigos implementados nesta solução estão nos Apêndices de A a Z deste trabalho.

4.1 Trabalhos Futuros

Para sugestões de implementações futuras deixa-se:

- Implementar confirmação de realização das tarefas, de forma a garantir que o microcontrolador realizou a tarefa solicitada como, por exemplo, o fechamento de uma tranca física com a utilização de um sensor de detecção de abertura e fechamento da porta de forma a retroalimentação da informação.
- Implementar aplicação em modelo cliente-servidor de forma que as informações pertinentes ao cliente sejam fornecidas pelo servidor, deixando para a aplicação Web o papel de interface do usuário.
- Criar interface mais amigável ao usuário para o ESP32 para adicionar dispositivos a serem acionados na aplicação.

REFERÊNCIAS

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

ANTUNES, Ana. **APPS NATIVOS, HÍBRIDOS OU PWA? QUAL O MELHOR PARA A MINHA SOLUÇÃO?** 2019. Disponível em: https://gobacklog.com/blog/nativos-hibridos-pwa/. Acesso em: 29 out. 2020.

BAGS, Marcio. **Cresce setor de tecnologia para a segurança privada e pública**. In: Brasil by Bags. [S. l.], 28 jun. 2019. Disponível em: https://brasilbybags.com/cresce-setor-de-tecnologia-para-a-seguranca-privada-e-publica/. Acesso em: 25 ago. 2020.

BIØRN-HANSEN, Andreas; MAJCHRZAK, Tim A.; GRØNLI, Tor-Morten. **Progressive Web Apps: The Possible Web-native Unifier for Mobile Development**. Proceedings of the 13th International Conference on Web Information Systems and Technologies , Noruega, p. 344-351, 2017. DOI 10.5220/0006353703440351. Disponível em: https://www.scitepress.org/papers/2017/63537/63537.pdf. Acesso em: 28 set. 2020.

BOLZAN, Guilherme. **Impressões digitais: dos seus dedos até a investigação forense**. 2019. Disponível em: http://jornalismojunior.com.br/impressoes-digitais-dos-seus-dedos-ate-a-investigacao-forense/. Acesso em: 22 out. 2020.

BRANDÃO, Bruna. **O que é geolocalização: Como essa tecnologia revolucionou o cotidiano de tantas pessoas e empresas?.** In: Maplink. [S. l.], 4 mar. 2020. Disponível em: https://maplink.global/blog/o-que-e-geolocalizacao/. Acesso em: 25 ago. 2020.

CARRION, Patrícia; QUARESMA, Manuela. **Internet da Coisas (IoT): Definições e aplicabilidade aos usuários finais.** Human Factors In Design, Florianópolis, v. 8, n. 15, p. 51-63, mar. 2019. Disponível em: https://www.revistas.udesc.br/index.php/hfd/issue/view/Volume%208%2C%20n%C2%BA%2015%2C%202018. Acesso em: 17 set. 2020.

CHASE, Otavio. **Sistemas Embarcados. 2007**. Disponível em: http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf. Acesso em: 18 nov. 2020.

COLUSSI, Cássio. **Aplicativos Híbridos vs Nativos: prós e contras.** Disponível em: https://www.keyworks.com.br/aplicativos-hibridos-vs-nativos-pros-e-contras/. Acesso em: 29 out. 2020.

CROMBIE, Hanna. **Os principais casos de uso para geofencing em 2020.** 2020. Disponível em: https://blog.pusher.com/top-use-cases-for-geofencing-in-2020/. Acesso em: 18 maio 2021.

CRUZ, Lucas. **O que é Geofencing?** 2017. Disponível em: https://expertdigital.net/o-que-e-geofencing/#gsc.tab=0. Acesso em: 18 maio 2021.

FARINACCIO, Rafael. **Comparativo: 10 formatos de áudio e quando você deve utilizá-los.** 2016. Disponível em: https://www.tecmundo.com.br/software/120043-diferencas-entre-formatos-audio-wav-mp3-aac-flac.htm. Acesso em: 02 maio 2021.

FERNANDES, Gabriel do N. et al. **Projeto e desenvolvimento de aplicações para dispositivos móveis híbridas a partir de tecnologias para Web: um estudo de caso em jogos digitais.** Ces Revista, Minas Gerais, v. 1, n. 1, p. 2-5, 2017. Disponível em: https://seer.cesjf.br/index.php/cesi/article/view/1263/922. Acesso em: 28 set. 2020.

FIRTMAN, Maximiliano. **Progressive Web Apps em 2020: status atual da plataforma pwa.** Status atual da plataforma PWA. 2020. Disponível em: https://firt.dev/pwa-2020/. Acesso em: 18 nov. 2020.

GAMA, Alexandre. **O que é JSON. 2011.** Disponível em: https://www.devmedia.com.br/perfil/alexandre-gama. Acesso em: 02 maio 2021.

GARRETT, Filipe. **O que é Microsoft Azure? Veja como funciona e preços do serviço de nuvem. 2020.** Disponível em: https://www.techtudo.com.br/listas/2020/07/o-que-e-microsoft-azure-veja-como-funciona-e-precos-do-servico-de-nuvem.ghtml. Acesso em: 10 maio 2021.

HILLESHEIN, HENRIQUE**. Desenvolvimento de um Sistema de Reconhecimento de Locutor Utilizando Aprendizado de Máquina.** Orientador: Mario de Noronha Neto. 2018. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Telecomunicações) - Instituto Federal de Santa Catarina, São José, SC, 2018. Disponível em: https://wiki.sj.ifsc.edu.br/wiki/images/1/17/TCC290_Henrique_Hilleshein.pdf. Acesso em: 15 out. 2020.

LACERDA, Flavia; LIMA-MARQUES, Mamede. **Da necessidade de princípios de Arquitetura da Informação para a Internet das Coisas.** Perspectivas em Ciência da Informação, Belo Horizonte, v. 20, ed. 2, p. 160-161, 2015. Disponível em: https://www.scielo.br/pdf/pci/v20n2/1413-9936-pci-20-02-00158.pdf. Acesso em: 28 set. 2020.

LENON. Node.js – **O que é, como funciona e quais as vantagens.** 2018. Disponível em: https://www.opus-software.com.br/node-js/. Acesso em: 15 maio 2021.

LONGEN, Andrei. **O Que É npm?** 2019. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-npm. Acesso em: 15 maio 2021.

MAFRA, Alexandre Teixeira. **RECONHECIMENTO AUTOMÁTICO DE LOCUTOR EM MODO INDEPENDENTE DE TEXTO POR SELF-ORGANIZING MAPS 2002.** Disponível em: https://teses.usp.br/teses/disponiveis/3/3132/tde-16052005-083957/publico/mafra_2002_dissertacao.pdf. Acesso em: 18 nov. 2020.

MAGRANI, Eduardo. **A Internet das Coisas.** 1. ed. Rio de Janeiro: FGV, 2018. 192 p. ISBN 9788522520060. ZUIN, Vânia Gomes;
ZUIN, Antônio Álvaro Soares. **A FORMAÇÃO NO TEMPO E NO ESPAÇO DA INTERNET DAS COISAS.** Revista Educação & Sociedade, Campinas, v. 37, ed. 136, p. 1-10, 2016. DOI 10.1590/ES0101-73302016167198. Disponível em: https://doi.org/10.1590/es0101-73302016167198. Acesso em: 28 set. 2020.

MARCONDES, José Sérgio. Biometria, **Sistema Biométrico: O que é, como funciona?** 2020. Disponível em: https://gestaodesegurancaprivada.com.br/biometria-sistema-biometrico-o-que-e-como-funciona/. Acesso em: 28 out. 2020.

MATOS, Beatriz Rezener Dourado; SILVA, João Gabriel de Britto. **Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando plataformas nativas e multiplataforma.** Orientador: Prof. Dr. Paulo Roberto Miranda Meirelles. 2016. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) - Universidade de Brasília - UnB, [S. l.], 2016. Disponível em: http://fga.unb.br/articles/0001/5113/Beatriz_Joao_TCC_Aplicativos_M_veis.pdf. Acesso em: 1 out. 2020.

MOZILA (org.). Navigator.getUserMedia. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/API/Navigator/getUserMedia. Acesso em: 15 maio 2021.

OLIVEIRA, Suellen. **Sistemas Operacionais Embarcados.** 2018. Disponível em: https://medium.com/@suellensantos2209/sistemas-operacionais-embarcados-8e4e025853b1. Acesso em: 18 nov. 2020.

PARADA, Marcelo Gonzaga de Oliveira. **BIOMETRIA MULTIMODAL BASEADA NOS SINAIS DE VOZ E FACIAL.** 2018. 110 f. Tese (Doutorado) - Curso de Doutorado em Engenharia Elétrica, Centro Universitário Fei, São Bernardo do Campo, Sp, 2018. Disponível em: http://sofia.fei.edu.br:8080/pergamumweb/vinculos/000033/000033e7.pdf. Acesso em: 15 out. 2020.

PAREIN, Heloise. **Pergunta da entrevista sobre JavaScript: O que é um Blob?** 2020. Disponível em: https://codeburst.io/JavaScript-interview-question-what-is-a-blob-f54482317e7f. Acesso em: 02 maio 2021.

PEREIRA, P. M. da S. et al. **Sistema de verificação biométrica baseado na impressão palmar.** In: ANAIS DOS SEMINÁRIOS DE MICROELETRÔNICA DO PARANÁ, 2018, Curitiba, Pr: Ufpr, 2018. Disponível em: http://www.gics.ufpr.br/semicropr2018/Anais%20SeMicro-PR%202018.pdf. Acesso em: 15 out. 2020.

RIGONATTO, Mariana. **"O que é Fonética?"**; Brasil Escola. Disponível em: https://brasilescola.uol.com.br/o-que-e/portugues/o-que-fonetica.htm. Acesso em 15 de outubro de 2020.

SCHULTZ, Felix. **O que é um endpoint: Como proteger este tipo de dispositivo.** 2020. Disponível em: O que é um endpoint: Como proteger este tipo de dispositivo. Acesso em: 02 maio 2021.

SINFIC. **Características biométricas e tecnologias mais comuns.** Disponível em: < http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=25032> Acesso em: nov 2020.

STATISTA. **Number of smartphone users in Brazil from 2015 to 2025.** Disponível em: https://www.statista.com/forecasts/285604/number-of-smartphone-users-in-brazil. Acesso em. Acesso em: 23 maio 2021.

THEMES, Jekyll. **Microcontroladores.** Disponível em: https://fiozera.com.br/microcontroladores-914a59cbf7de. Acesso em: 23 maio 2021.

TREVORBYE; OLPROD (ed.). **O que é Reconhecimento do Locutor.** 2020. Disponível em: https://docs.microsoft.com/pt-br/azure/cognitive-services/speech-service/speaker-recognition-overview. Acesso em: 10 maio 2021.

UOL. **Vogais e o aparelho fonador - Como são feitos os sons da fala.** Disponível em: https://educacao.uol.com.br/disciplinas/portugues/vogais-e-o-aparelho-fonador-como-sao-feitos-os-sons-da-fala.htm. Acesso em: 18 nov. 2020.

VENESS, Chris. **Calcule a distância, rumo e mais entre os pontos de latitude / longitude.** Disponível em: https://www.movable-type.co.uk/scripts/latlong.html. Acesso em: 16 maio 2021.

WEBPACK. WebPack: **Conceitos. Documentação.** Disponível em: https://webpack.js.org/concepts/. Acesso em: 18 maio 2021.

APÊNDICES

APÊNDICE A – Camada de Controle da Aplicação – index.js

```
/*
  Descrição: Camada de Controle da Aplicação
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

/*      IMPORTAÇÕES     */
/*      IMPORTAÇÕES     */
/*      IMPORTAÇÕES     */

//Importar Models
import Maps from './models/Maps';
import Login from './models/Login';
import Modal from './models/Modal';
import Button from './models/Button';
import Fence from './models/Fence';
import DataBase from './models/Repository/DataBase';
import FenceRepository from './models/Repository/FenceRepository';
import Speaker from './models/Speaker';
import UserRepository from './models/Repository/UserRepository';
import LockedRepository from './models/Repository/LockedRepository';
import Loading from './models/Loading';
import SpeakerRepository from './models/Repository/SpeakerRepository';
import User from './models/User';

//Importar as VIEWS
import * as MapsView from './views/mapsView';
import * as LoginView from './views/loginView';
import * as ModalView from './views/modalView';
import * as LoadingView from './views/loadingView';
import * as ButtonView from './views/buttonView';

//Importar alguns Elementos do DOM
import { elements } from './views/base';

//Importar arquivo de SASS
import '../scss/main.scss';


/*      DEFININDO OBJETO DE ESTADOS QUE SERÁ ACESSADO POR QUALQUER FUNÇÃO NESTE S
CRIPT     */
/*      DEFININDO OBJETO DE ESTADOS QUE SERÁ ACESSADO POR QUALQUER FUNÇÃO NESTE S
CRIPT     */
```

```
/*      DEFININDO OBJETO DE ESTADOS QUE SERÁ ACESSADO POR QUALQUER FUNÇÃO NESTE S
CRIPT     */

const state = {};
      state.loadings = {};
      state.modals = {};
      state.speaker = {};

/*      FUNÇÕES QUE IRAM ATUAR COMO CONTROLADORES DE SUAS MODELS E VIEWS     */
/*      FUNÇÕES QUE IRAM ATUAR COMO CONTROLADORES DE SUAS MODELS E VIEWS     */
/*      FUNÇÕES QUE IRAM ATUAR COMO CONTROLADORES DE SUAS MODELS E VIEWS     */

//Controle de Login
const loginController = loginProperties => {
  try{

    state.login = new Login(loginProperties);

    LoginView.renderLoading(state.login);

  }catch(err){
    console.log(err);
  }
};

//Controle de Modals
const modalController = modalProperties => {
  try{

    let modal_id = JSON.stringify(modalProperties.modalId);

    state.modals[modal_id] = new Modal(modalProperties);

    ModalView.rederTheModal(state.modals[modal_id]);

  }catch(error){
    console.log(error)
  }
};

//Controle de Mapas
const mapsController = mapsProperties => {
  try{
    state.maps = new Maps(mapsProperties);

    MapsView.renderTheMap(state.maps);
```

```
  }catch(error){
    console.log(error);
  }
};

//Controle de Botões
const buttonController = button => {
  state.buttons = {};

  let button_id = JSON.stringify(button.id);

  state.buttons[button_id] = new Button(button);

  ButtonView.renderButton(state.buttons[button_id]);

};

//Controle do Banco de Dados
const dbController = configs => {
  state.database = new DataBase(configs);
};

//Controle da Cerca
const fenceController = fence => {
  state.fence = new Fence(fence);
};

//Controle do Loading, executado quando algo está em processamento
const loadingController = loading => {

  state.loadings[loading.loadingId] = new Loading(loading);

  LoadingView.renderLoading(state.loadings[loading.loadingId]);

};

//Controle da verificação da localização do usuário em relacão à Cerca Virtual
const fenceVerificationController = mapsProperties => {

  try{

    loadingController({ loadingId: 'awaitingFenceVerification' });

    LoadingView.showLoading('awaitingFenceVerification');

    if(!state.database){
      inicializarDB();
    }
```

```
    const fenceRepository = new FenceRepository(state.database);

    if(! state.maps){

      if(!document.querySelector('#map')){
        elements.body.innerHTML = `<div id='map'></div>`;
      }

      state.maps = new Maps(mapsProperties);
    }

    let request =  fenceRepository.read({username: state.current_username});


    request.then(cercaCadastrada => {

      console.log(cercaCadastrada);

      MapsView.renderTheMapVerification(state.maps, cercaCadastrada);



      LoadingView.hideLoading('awaitingFenceVerification');

    }).catch(err => {
      console.log(err);
    })




  }catch(error){
    console.log(error);
  }

};


/*      FUNÇÕES QUE UTILIZAM OS CONTROLES ACIMA PARA EXECUTAR A APLICAÇÃO     */
/*      FUNÇÕES QUE UTILIZAM OS CONTROLES ACIMA PARA EXECUTAR A APLICAÇÃO     */
/*      FUNÇÕES QUE UTILIZAM OS CONTROLES ACIMA PARA EXECUTAR A APLICAÇÃO     */

//Envia o ID de Locutor gerado para o Banco de Dados (Firebase)
const sendIdToDB = (speaker) => {

  try{

    if(!state.database){
```

```
      inicializarDB();
    }

    const speakerRepository = new SpeakerRepository(state.database);

    return speakerRepository.create(state.username, speaker);


  }catch(err){
    console.log(err);
  }

};

//Cria um BLOB com os segundos informados no parâmetro da função
const createBlob =  (seconds) => {

  navigator.getUserMedia({
    audio: true
  }, stream => {
    onRecorderSuccess(stream, setarAudioBlob, seconds)
  }, onRecorderError);

};

//Inicia a gravação da voz do usuário para um NOVO cadastro
const recordToSubscribe = () => {
  console.log('Gravação para Cadastro de Novo Locutor Iniciada...');

  const speaker = new Speaker();

  LoginView.renderAudioRecordingToSubscribe();
  createBlob(15);


  setTimeout(() => {
    const resultado = speaker.createProfile(state.audioBlob);

    resultado.then(res1 => {

      //irá devolver o ID do novo Locutor, esse ID será enviado para o cadastro d
o usuário no Firebase
      LoadingView.showLoading('initialLoading');

      console.log(state.audioBlob);

      //Chama o método que irá inscrever o locutor na API de reconhecimento de Lo
cutor
```

```
      speaker.enrollProfileAudio(state.audioBlob, res1).then(res2 => {
        console.log();

        speaker.id = res2;

        //Cadastrar o ID de Locutor do usuário

        //Oculta o Loading
        LoadingView.hideLoading('initialLoading');

        sendIdToDB(speaker).then(res3 => {
          LoginView.deleteLoginView();
          //redicionar para cadastro de cerca
          iniciarMapa();
        });

      }).catch(err => {
        console.log(err);

      })
    }).catch(err => {
      console.log(err);
    })
  }, 15000)

};

//Inicia a gravação para reconhecimento do locutor JÁ CADASTRADO
const startRecordToRecognition = () => {

  console.log('Gravação para Reconhecimento do Locutor (já cadastrado) Iniciada..
.');


  const speaker = new Speaker();

  createBlob(10);


  if(!state.database){
    inicializarDB();
  }

  let result = callIdentifier(speaker);

  state.lockedRepository = new LockedRepository(state.database);

  result.then(res => {
```

```
    let jsonResult = JSON.parse(res);
    console.log(jsonResult);

    if(jsonResult.identifiedProfile.score > 0.5){

      console.log('FOI IDENTIFICADO! VOCÊ É MESMO QUEM DIZ SER, VAMOS CARREGAR O
MAPA');

      const userRepository = new UserRepository(state.database);

      userRepository.read({username: state.current_username}).then(res => {
        state.current_user= {};
        state.current_user = res.name;
        console.log(state.current_user);

        //Informa que o usuário FOI RECONHECIDO
        LoginView.renderAudioRecordCompleted(state.current_username,true);


        //Passando as informações para rendereizar o Mapa da
        //Localização Atual e verificar se está dentro da Cerca

        console.log('AQUI ESTOU COMEÇANDO A RENDERIZAR O MAPA');

        setTimeout(() => {
          navigator.geolocation.watchPosition(local => {

            console.log('Consegui obter a localização: Latitude: ' + local.coords
.latitude + ' Longitude:' + local.coords.longitude);

            if(!document.querySelector('#acionamento-tranca')){

              //Cria um button para Acionar a tranca
              buttonController({id: 'acionamento-
tranca', value: 'ABRIR TRANCA', parent: 'body', classList: 'btn btn-acionar-
tranca', disabled: true, position: 'beforeend'});
            }

            //Renderiza o mapa e localização atual do usuário em relação a cerca
virtual
            fenceVerificationController({
              longitude: local.coords.longitude,
              latitude: local.coords.latitude,
              idElementMap: 'map',

              //Elemento HTML5 que irá marcar a posição do usuário no mapa
              markerOptions: {
```

```
                htmlContent: '<div><div class="pin bounce"></div><div class="puls
e"></div></div>',
                color: 'DodgerBlue',
                text: 'o',
                position: [local.coords.longitude, local.coords.latitude]
              },

              //Popup acima do ícone que mostra a localização do usuário no mapa
              popupMarkerOptions: {

                htmlContent: "",
                pixelOffset: [0, -30]
              },

              //Propriedades do Mapa
              properties: {
                center: [local.coords.longitude, local.coords.latitude],
                zoom: 15,
                view: "Auto",
                authOptions: {
                  authType: 'subscriptionKey',
                  subscriptionKey: 'JEs_jF8YIe3SEQDDCY_9o4TAccluspr jHC8b0VBtvOw',
                  clientId: 'e9fb7e5f-71e4-400c-8e56-5f6d07d50184',
                  getToken: function (resolve, reject, map) {
                      var tokenServiceUrl = "https://azuremapscodesamples.azurewe
bsites.net/Common/TokenService.ashx";
                      fetch(tokenServiceUrl).then(r => r.text()).then(token => re
solve(token));
                  }
                }
            }})
          });
        }, 10);


      });


    }else{

      //Informa que o usuário NÃO FOI RECONHECIDO
      LoginView.renderAudioRecordCompleted(state.current_username,false);
    }
  }).catch(err => {
    console.log(err);
  })

  LoginView.renderAudioRecording();
```

```
};

//Chama o método que irá verificar se a voz gravada é mesmo do username informado
const callIdentifier = async (speaker) => {

  return new Promise((resolve, reject) => {
    setTimeout(() => {
      speaker.startListeningForIdentification(state.speaker.id, state.audioBlob).
then(res => {
        if(res){
          resolve(res);
        }
      }).catch(err => {
        reject(err);
      })
    }, 12000);
  })
};

//Guarda o BLOB no state, para ser acessível globalmente
const setarAudioBlob = blob => {
  //console.log(blob);
  state.audioBlob = blob;
};

//Parar a gravação de voz do usuário e chamar a callback de sucesso
function StopListening(callback){
  console.log('Parando a gravação...');
  state.recorder && state.recorder.stop();
  state.recorder.exportWAV(blob => {
      callback(blob);
  });
  state.recorder.clear();
};

//Callback de erro, utilizada caso ocorra erro na gravação de voz do usuário
function onRecorderError(e) {
  console.error('media error', e);
};

//Callback e sucesso, utilizada caso consiga fazer a gravação da voz do usuário c
orretamente
function onRecorderSuccess(stream, callback, secondsOfAudio) {


  //Criação do contexto de áudio, para captura de voz do usuário
  let audio_context = audio_context || new window.AudioContext;
```

```
  var input = audio_context.createMediaStreamSource(stream);

  //criação de um objeto de áudio que pode ser reproduzido e manipulado
  state.recorder = new Recorder(input);
  state.recorder.record();


  //Aguarda os segundos (passados na variável secondOfAudio)
  //para encerrar a transmissão do áudio
  setTimeout(() => {
    StopListening(callback);
  }, secondsOfAudio*1000);
};



/*      ASSISTE OS EVENTOS DE CLIQUE NO BODY E EXECUTA AÇÃO CONFORME O ELEMENTO C
LICADO     */
/*      ASSISTE OS EVENTOS DE CLIQUE NO BODY E EXECUTA AÇÃO CONFORME O ELEMENTO C
LICADO     */
/*      ASSISTE OS EVENTOS DE CLIQUE NO BODY E EXECUTA AÇÃO CONFORME O ELEMENTO C
LICADO     */

elements.body.addEventListener('click', e => {

  if(e.target.matches('#ok-initButton, #ok-initButton *')){

    //Oculta o Modal de Inicialização
    ModalView.hideModal('modalInit');

  } else if(e.target.matches('.close-modal, .close-modal *')){

    //Oculta o Modal de Confirmação de Cadastro de Cerca Virtual
    ModalView.hideModal('modalConfirmSubscribeFence');

  } else if(e.target.matches('#confirm-fence-subscribe, #confirm-fence-
subscribe *')){

    //Envia os dados da Cerca Virtual para cadastro no Banco de Dados
    fenceController(state.maps.fence);

    const fenceRepository = new FenceRepository(state.database);

    let envio = fenceRepository.create(state.username, state.fence);

    //Mostra o Loading até concluir o envio da cerca ao Bando de Dados
    loadingController({ loadingId: 'awaitingFenceSubscribe' });
    envio.then(deuCerto => {
```

```
      console.log(deuCerto);
      setTimeout(() => {
        LoadingView.hideLoading('awaitingFenceSubscribe');
      }, 1000 )
    }).catch(err => {
      console.log(err);
    })

  }else if(e.target.matches('#fence-subscribe, #fence-subscribe *')){

    //Mostra um Modal para confirmação dos dados da cerca virtual que será cadast
rada
    ModalView.setMensage({
      id: 'modalConfirmSubscribeFence',
      title: 'Você confirma a criação da cerca abaixo:',
      mensage: `Tipo: <strong>${state.maps.fence.type}</strong>
      <br>Área: <strong>${state.maps.fence.area} km2</strong>
      <br>Perímetro: <strong>${state.maps.fence.perimetro} km</strong>`
    })
    ModalView.showModal('modalConfirmSubscribeFence');

  }else if(e.target.matches('#btn_login, #btn_login *')){

    //Chama a Validação do Usuário (mostra o botão de início da gravação) quando
ele digita o username e clica em LOGIN
    e.preventDefault();
    initValidation();

  }else if(e.target.matches('.login-record__verify__icon, .login-
record__verify__icon *')){

    //Chama o início de reconhecimento do locutor, após clicar no microfone
    startRecordToRecognition();

  }else if(e.target.matches('#btn_subscribe, #btn_subscribe *')){

    //Mostra formulário de cadastro de novo usuário
    e.preventDefault();

    loadingController({ loadingId: 'initialLoading' });


    //obter os dados do formulário de cadastro
    const userData = LoginView.getDataNewUser();
    console.log(userData);

    if(! state.database){
      inicializarDB();
```

```
    }

    //verificar se este usuário já existe
    const userRepository = new UserRepository(state.database);

    userRepository.read(userData).then(res => {

      console.log(res);
      if(!res){

        userRepository.create(userData).then(res => {
          LoadingView.hideLoading('initialLoading');
          LoginView.renderAudioRecordToSubscribe();
          state.username = userData.username;
        })
      }else{

        //Se usuário já existir, mostra alerta informando
        LoadingView.hideLoading('initialLoading');
        alert('Este usuário já existe. Clique em Sign In e faça Login');
      }


    }).catch(err => {
      console.log(err);
    })



  }else if(e.target.matches('#go-to-sign-up, #go-to-sign-up *')){

    //Mostra formulário de cadastro
    LoginView.renderSignUpData();

  }else if(e.target.matches('.login-record__subscribe__icon, .login-
record__subscribe__icon *')){

    //Inicia a gravação de voz para cadastro de um NOVO USUÁRIO
    recordToSubscribe();

  }else if(e.target.matches('#acionamento-tranca, #acionamento-tranca *')){

    console.log("Usuário Atual: " + state.current_username);

    state.lockedRepository.read({username: state.current_username}).then(locked =
> {
      console.log('A tranca está: ' + JSON.stringify(locked));
      if(locked){
```

```
        state.lockedRepository.set(state.current_username, false).then(res => {
          document.querySelector('#acionamento-
tranca').innerHTML = "ABRIR TRANCA";
        });
      }else{
        state.lockedRepository.set(state.current_username, true).then(res => {
          document.querySelector('#acionamento-
tranca').innerHTML = "FECHAR TRANCA";
        });
      }
    })
  }
});


/*      FUNÇÕES DE INICIALIZAÇÃO     */
/*      FUNÇÕES DE INICIALIZAÇÃO     */
/*      FUNÇÕES DE INICIALIZAÇÃO     */

//Inicializa o Mapa (Azure)
const iniciarMapa = () => {

  //Verifica se já foi instanciado um objeto DataBase
  if(! state.database){
    inicializarDB();
  }

  //Cria elemento HTML5 que irá renderizar o Mapa
  document.body.innerHTML = `<div id="map"></div>
                              <main>
                              <div class="buttons-controll-fence"></div>
                            </main>`;

  //Cria modal para avisar que é hora do cadastro da cerca virtual

  modalController({
    modalId: 'modalInit',
    iconPath: '../images/icons/location.svg',
    title: 'Agora é hora de você criar a sua cerca Virtual!',
    mensage: "",
    hasCloseButton: false,
    classList: ""
  });

  //Cria botão no modal com ação de FECHAR MODAL para o alerta de cadastro de cer
ca virtual
```

```
  buttonController({id: 'ok-initButton', value: 'OK', parent: '.modal-
alert', classList: 'btn btn-accept-
alert', disabled: false, position: 'beforeend'});


  //Faz chamada ao controlador do Mapa para renderizar o cadastro de cerca virtua
l
  navigator.geolocation.getCurrentPosition(local => {
    mapsController({
      //encaminhas as coordenadas atuais do usuário
      longitude: local.coords.longitude,
      latitude: local.coords.latitude,

      //Identificador do Elemento que irá renderizar o Mapa no HTML5
      idElementMap: 'map',

      //Opções do Marcador da Localização atual do usuário
      markerOptions: {
        htmlContent: '<div><div class="pin bounce"></div><div class="pulse"></div
></div>',
        color: 'DodgerBlue',
        text: 'o',
        position: [local.coords.longitude, local.coords.latitude]
      },

      //Opções da mensagem que aparece no Popup
      popupMarkerOptions: {
        htmlContent: "",
        pixelOffset: [0, -30]
      },

      //Propriedades da visualização do Mapa
      properties: {
        center: [local.coords.longitude, local.coords.latitude],
        zoom: 15,
        view: "Auto",
        authOptions: {
          authType: 'subscriptionKey',
          subscriptionKey: 'JEs_jF8YIe3SEQDDCY_9o4TAcclusprjHC8b0VBtvOw',
          clientId: 'e9fb7e5f-71e4-400c-8e56-5f6d07d50184',
          getToken: function (resolve, reject, map) {
              var tokenServiceUrl = "https://azuremapscodesamples.azurewebsites.n
et/Common/TokenService.ashx";
              fetch(tokenServiceUrl).then(r => r.text()).then(token => resolve(to
ken));
          }
        }
    }});
```

```
  });

  //Cria um botão com ação de cadastro de cerca virtual
  buttonController({
    id: 'fence-subscribe',
    value: 'cadastrar cerca',
    parent: '.buttons-controll-fence',
    classList: 'btn btn-primary',
    disabled: true,
    position: 'afterbegin'
  });

  //Cria um modal para o usuário confirmar o cadastro da cerca virtual
  modalController({
    modalId: 'modalConfirmSubscribeFence',
    iconPath: '../images/icons/select.svg',
    title: `Você confirma a criação da cerca abaixo:`,
    mensage: '',
    hasCloseButton: true,
    classList: 'hidden',
  });

  //Cria um botão no modal com ação de confirmação da cerca virtual
  buttonController({
    id: 'confirm-fence-subscribe',
    value: 'Confirmar',
    parent: '#modalConfirmSubscribeFence>.modal-alert',
    classList: 'btn btn-primary',
    disabled: false,
    position: 'beforeend'
  });

};

//Inicializa o Banco de Dados (Firebase)
const inicializarDB = () => {

  dbController({
    apiKey: "AIzaSyD7BorZwXPshl0hoF_vU9TMwp2fh3v2DbM",
    authDomain: "tcc-2-1e9c6.firebaseapp.com",
    projectId: "tcc-2-1e9c6",
    storageBucket: "tcc-2-1e9c6.appspot.com",
    messagingSenderId: "394765204468",
    appId: "1:394765204468:web:5f6e77d6f878166bdc17a1"
  });

};
```

```
//Inicializa o reconhecimento do Locutor (Azure)
const initSpeakerRecognition = (speakerId) => {

  /*Aqui será disponibilizado o botão para o usuário
  clicar e iniciar a gravação para o Reconhecimento do Locutor*/

  LoginView.renderAudioRecord();
  state.speaker.id = speakerId;
};

//Inicializa a Validação do Usuário
const initValidation = () => {

  const speakerRepository = new SpeakerRepository(state.database);

  state.current_username = document.querySelector('#username-login').value;
  console.log(state.current_username);

  let speakerId = speakerRepository.read({username: state.current_username});

  speakerId.then(encontrado => {
    console.log(encontrado.id);
    if(encontrado){
      initSpeakerRecognition(encontrado.id)
    }else{
      alert('Usuário não Encontrado');
    }
  }, naoEncontrado => {
    console.log(naoEncontrado);
  });

};

//Inicializa a Aplicação
const inicializarApp = () => {

  inicializarDB();

  loginController({fields: "algunsFields"});
};

//Aguarda a página ser iniciada e chama o Início da Aplicação
window.onload = inicializarApp;
```

APÊNDICE B – Button Model – Button.js

```
/*
  Descrição: Classe Model botões
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

export default class Button {
  constructor(button){
    this.id = button.id;
    this.value = button.value;
    this.parent = button.parent;
    this.classList = button.classList;
    this.disabled = button.disabled;
    this.position = button.position;
  }
}
```

APÊNDICE C – Fence Model – Fence.js

```
/*
  Descrição: Camada de Model de Cerca
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/


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

export default class Fence{
  constructor(fence){
    this.type = fence.type;
    this.perimeter = fence.perimetro,
    this.area = fence.area,
    this.coordinates = fence.coordinates;
    this.radius = fence.radius;
  }

  /*
    Método responsável por verificar se o usuário está dentro da cerca cadastrada
    Primeiramente calcula-se a distância entre o centro
    da circunferência da cerca e a localização atual
    depois é comparado a distância obtida com o raio da cerca
```

```
    Se esta distância for menor ou igual, o usuário está dentro da cerca
    Se for menor, o usuário está fora da cerca
  */
  onFenceArea({lat, lng}) {

    let distance = atlas.math.getDistanceTo([lng, lat], [this.coordinates[0], thi
s.coordinates[1]], "kilometers");
    console.log('Distancia calculada: ' + distance);
    console.log('Raio Guardado: ' + this.radius);

    return distance <= this.radius;
  }
}
```

APÊNDICE D – Loading Model – Loading.js

```
/*
  Descrição: Classe de Modelo de Loading
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

export default class Loading {
  constructor(loading){
    this.loadingId = loading.loadingId;
  }
}
```

APÊNDICE E – Login Model – Login.js

```
/*
  Descrição: Classe de Modelo de Login
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

export default class Login {
  constructor(login){
    this.fields = login.fields;
    this.thingsToRead = login.thingsToRead;
  }
};
```

APÊNDICE F – Maps Model – Maps.js

```
/*
  Descrição: Classe de Modelo de Mapa
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

import * as atlas from 'azure-maps-control';
import * as atlasDraw from 'azure-maps-drawing-tools';
import Fence from './Fence';

let drawingManager , area, fence = {};

export default class Maps {

  constructor(configs){

    this.mapsConfigs = configs.properties;

    this.map = new atlas.Map(configs.idElementMap, this.mapsConfigs);

    this.longitude = configs.longitude;

    this.latitude = configs.latitude;

    this.drawnshape = {};

    this.fence;

    this.markerOptions = configs.markerOptions;

    this.markerOptions.popup =  new atlas.Popup(configs.popupMarkerOptions);

    this.marker = new atlas.HtmlMarker(this.markerOptions);

    this.map.events.add('click', this.marker, () => {
      this.marker.togglePopup();
    });

    this.map.markers.add(this.marker);

    this.fenceSubscribeButton = document.querySelector('#fence-subscribe');

  }

  buildToolbar(){
```

```
    /*
      Executa a criação da barra de ferramentas de desenho
    */
    this.map.events.add('ready', () => {
      drawingManager = new atlasDraw.drawing.DrawingManager(this.map, {
          toolbar: new atlasDraw.control.DrawingToolbar({
            position: 'non-fixed',
            style: 'dark',
            buttons: ['draw-circle']
          })
      });

      this.map.events.add('drawingmodechanged', drawingManager, mode => {


        if (mode.startsWith('draw')) {
          drawingManager.getSource().clear();
          this.fenceSubscribeButton.disabled = true;
        }
      });

      this.map.events.add('drawingchanging', drawingManager, this.measureShape);

      this.map.events.add('drawingcomplete', drawingManager, ()=> {

        this.fenceSubscribeButton.disabled = false;

        this.fenceSubscribeButton.addEventListener('click', ()=>{
          document.querySelector('#modalConfirmSubscribeFence').classList.remove(
'hidden');
        })

        drawingManager.setOptions({ mode: 'idle' });

        this.measureShape;

        this.fence = fence;

      });

    });
  }

  measureShape(shape) {


    fence.coordinates = {};
    if (shape.isCircle()) {
```

```
      fence = {};

      let result = JSON.parse(JSON.stringify(drawingManager.getSource().toJson())
);

      fence.coordinates = result.features[0]["geometry"]["coordinates"];
      console.log(result);

      console.log(fence.coordinates);
      fence.type = 'circle';

      ///fence.radius = atlas.math.convertDistance(shape.getProperties().radius,
'meters', 'kilometers', 2);
      fence.radius = atlas.math.convertDistance(shape.getProperties().radius, 'me
ters', 'kilometers', 5);
      fence.area = Math.round(2 * Math.PI * fence.radius * fence.radius * 100) /
100;
      fence.perimetro = Math.round(2 * Math.PI * fence.radius * 100) / 100;


    } else {

      /*Rectangle*/
      fence = {};
      fence.type = 'retangulo';
      fence.geometry = shape.toJson().geometry;
      //fence.coordinates = fence.geometry;
      fence.coordinates = fence.geometry.coordinates[0];

      console.log(fence.coordinates);

      fence.perimetro = Math.round(atlas.math.getLengthOfPath(fence.geometry.coor
dinates[0], 'kilometers') * 100) / 100;

      fence.area = atlas.math.getArea(fence.geometry, 'squareKilometre', 2);
    }
  }

  showFenceVerification(fence) {
    this.map.events.add('ready', () => {
      let dataSource = new atlas.source.DataSource();
      this.map.sources.add(dataSource);

      console.log(fence.radius);

      dataSource.add(new atlas.data.Feature(new atlas.data.Point(
        [fence.coordinates[0], fence.coordinates[1]]), {
```

```
          subType: fence.type,
          radius: fence.radius*1000
        })
      );

      this.map.layers.add(new atlas.layer.PolygonLayer(dataSource, null, {
        fillColor: 'rgba(0, 200, 200, 0.8)'
      }));

    });
  }

};
```

APÊNDICE G – Modal Model – Modal.js

```
/*
  Descrição: Classe de Modelo de Modal
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
export default class Modal {

  constructor(modal){
    this.modalId = modal.modalId;
    this.iconPath = modal.iconPath;
    this.title = modal.title;
    this.mensage = modal.mensage;
    this.hasCloseButton = modal.hasCloseButton;
    this.classList = modal.classList;
  }
}
```

APÊNDICE H – Perfil Model – Profile.js

```
/*
  Descrição: Classe de Modelo de Perfil
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
export default class Profile {
  constructor (name, profileId) {
    this.name = name;
    this.profileId = profileId;
  }
};
```

APÊNDICE I – Locutor Model – Speaker.js

```
/*
  Descrição: Classe de Modelo de Locutor
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

import VerificationProfile from './VerificationProfile';
import Profile from './Profile';


let  recorder;
let audio_context;
let id = "";


const key = "8e962cfc24a445dbbed5cd6b9f922df1";

var profileIds = [];

(function () {
  //Verifica qual método de gravação de mídia está disponível no navegador
  navigator.getUserMedia = ( navigator.getUserMedia ||
              navigator.webkitGetUserMedia ||
              navigator.mozGetUserMedia ||
              navigator.msGetUserMedia);

})();

//url base da API dos Serviços de Cognição da Azure, dentre eles o de Reconhecime
nto de Locutor
const baseApi = "https://westus2.api.cognitive.microsoft.com/";

//Endpoint de criação ID de Identificação do Locutor
const createIdentificationProfileEndpoint = `${baseApi}/speaker/identification/v2
.0/text-independent/profiles`;

//Endpoint de cadastro do usuário com ID de Identificação do Locutor já criado.
const enrollIdentificationProfileEndpoint = (profileId) => { return `${baseApi}/s
peaker/identification/v2.0/text-
independent/profiles/${profileId}/enrollments?ignoreMinLength=true`;
};


//Endpoints
```

```
const enrollIdentificationProfileStatusEndpoint = (profileId) => `${baseApi}/spea
ker/identification/v2.0/text-independent/profiles/${profileId}`;
const identifyProfileEndpoint = (Ids) => `${baseApi}/speaker/identification/v2.0/
text-
independent/profiles/identifySingleSpeaker?profileIds=${Ids}&ignoreMinLength=true
`;


export default class Speaker {
  constructor(){
    this.resultadoDaVerificação = {};
    this.id="";
    this.verificationProfile = new VerificationProfile();
  }

  //-- Speaker Identification methods
  // 1. Start the browser listening, listen for 15 seconds, pass the audio stream
 to "createProfile"
  enrollNewProfile(){
    navigator.getUserMedia({audio: true}, function(stream){
      console.log('I\'m listening... just start talking for a few seconds...');
      console.log('Maybe read this: \n' + thingsToRead[Math.floor(Math.random() *
 thingsToRead.length)]);
      onMediaSuccess(stream, createProfile, 15);
    }, onMediaError);
  }

  //cria o perfil de usuário na API de Reconhecimento de Locutor
  createProfile(blob){

    //retorna uma Promisse contendo o ID do Perfil de Identificação do Locutor
    return new Promise((resolve, reject) => {

      let request = new XMLHttpRequest();
      request.open("POST", createIdentificationProfileEndpoint, true);

      request.setRequestHeader('Content-Type','application/json');
      request.setRequestHeader('Ocp-Apim-Subscription-Key', key);

      request.onload = function () {

        let json = JSON.parse(request.responseText);
        console.log(json);

        let profileId = json.profileId;

        //resolve a promisse retornando o ID do Perfil de Identificação do Locuto
r
```

```
        resolve(profileId);
      };

      request.send(JSON.stringify({ 'locale' :'en-us'}));
    });
  }


  //cadastra o novo ID de Locutor gerado
  enrollProfileAudio(blob, profileId){

    return new Promise((resolve, reject) => {
      let request = new XMLHttpRequest();
      request.open("POST", enrollIdentificationProfileEndpoint(profileId), true);
      request.setRequestHeader('Ocp-Apim-Subscription-Key', key);
      request.onload = function () {
        console.log('enrolling');

      if (request.status==200 || request.status==201) {
        let json = JSON.parse(request.responseText);
        console.log(json);

        resolve(profileId);


      } else {
        console.log(`Failed to submit for enrollment: got a ${request.status} res
ponse code.`);
        var json = JSON.parse(request.responseText);
        console.log(`${json.error.code}: ${json.error.message}`);

        reject(`${json.error.code}: ${json.error.message}`);



      }
      };
      request.send(blob);
    });

  }


  checkAudio(blob) {

    return new Promise((resolve, reject) => {

      let request = new XMLHttpRequest();
      request.open("POST", identifyProfileEndpoint(id), true);
```

```
      request.setRequestHeader('Ocp-Apim-Subscription-Key', key);
      request.onload = function () {

        if(request.status >= 200 &&  request.status <300){

          this.resultadoDaVerificação = JSON.parse(request.responseText).identifi
edProfile;

          resolve(request.responseText);

        }else{
          reject(request.statusText);
        }

      };

      request.send(blob);

    });

  }

  //Começa a ouvir para identificar o usuário
  startListeningForIdentification(idUser, blob){

    id = idUser;
    console.log('Estou ouvindo...fale alguma coisa por 10 segundos...');

    return this.checkAudio(blob);
  }

}
```

APÊNDICE J – Usuário Model – User.js

```
/*
  Descrição: Classe de Modelo de Usuário
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
export default class User {
  constructor(user){
    this.username = user.username;
    this.firstName = user.firstName;
    this.lastName = user.lastName;
  }
}
```

APÊNDICE K – Perfil de Verificação Model – VerificationProfile.js

```
/*
  Descrição: Classe de Modelo de Verificação de Perfil
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

export default class VerificationProfile {
  constructor (name, profileId) {
  this.name = name;
  this.profileId = profileId;
  this.remainingEnrollments = 3
  }
};
```

APÊNDICE L – Botões View – buttonView.js

```
/*
  Descrição: Botões View
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

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

export const renderButton = button => {

  const markup = `<button ${(button.disabled)?'disabled':''} class='${button.clas
sList}' id=${button.id}>${button.value}</button>`;

  document.querySelector(button.parent).insertAdjacentHTML(`${button.position}`,
markup);
}
```

APÊNDICE M – Loading View – loadingView.js

```
/*
  Descrição: Loading View
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

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

```
export const renderLoading = loading => {

  const markup = `<div id='${loading.loadingId}' class="loading-parent">
                    <div class="loading-children">
                      <div></div>
                      <div></div>
                    </div>
                  </div>`;

  elements.body.insertAdjacentHTML('afterend', markup);
}

export const hideLoading = loadingId => {
  document.getElementById(loadingId).classList.add('hidden');
}

export const showLoading = loadingId => {
  document.getElementById(loadingId).classList.remove('hidden');
}
```

APÊNDICE N – Login View – loginView.js

```
/*
  Descrição: Login View
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
import { elements } from './base.js';

export const renderLoading = login => {
  const markup = `
    <main class='login__main'>
      <div class='login__main__container'>
        <div class='login-area'>
          <div class=''>
            <h2>Login</h2>
            <form>
              <input type='text' placeholder='username' name='username' required
id='username-login' class='field field__username'>
              <input  id='btn_login'  class='btn btn-
main' type='submit' value='Sign In'>
            </form>
            <span class='line-or'><p>or</p></span>
            <button id='go-to-sign-up' class='btn btn-tertiary'>Sign Up</button>
          </div>

          <div class='sign-up-area hidden'>
```

```
            <h2>Sign Up</h2>
            <form>
              <input type='text' placeholder='username' name='username' required
id='username-signup' class='field field__username'>
              <input type='text' placeholder='first name' name='firstname' requir
ed id='firstname-signup' class='field field__firstname'>
              <input type='text' placeholder='last name' name='lastname' required
 id='lastname-signup' class='field field__lastname'>
              <input  id='btn_subscribe'  class='btn btn-
main' type='submit' value='Subscribe'>
            </form>
            <span class='line-or'><p>or</p></span>
            <button class='btn btn-tertiary'>Sign In</button>
          </div>

          <div class='login-record hidden'>
            <div class='login-record__verify'>
              <div class='login-record__verify__icon'>
                <img src='images/icons/mic.svg'>
              </div>
              <p class='record-
msg'>Clique no microfone para começar o reconhecimento</p>
            </div>
            <button class='btn btn-return'>Return</button>
          </div>

          <div class='subscribe-record hidden'>
            <div class='login-record__verify'>
              <div id='voice-subscribe-button' class='login-
record__subscribe__icon'>
                <img src='images/icons/mic.svg'>
              </div>
              <p class='record-
msg'>Clique no microfone para começar o reconhecimento</p>
            </div>
            <button class='btn btn-return'>Return</button>
          </div>
        <div>
      </div>
    </main>
  `;

  elements.body.innerHTML = markup;
}

export const renderAudioRecord = () => {

  const record_section = document.querySelector('.login-record');
```

```
  const record__verify__icon = document.querySelector('.login-
record__verify__icon');
  const login_area = document.querySelector('.login-area>div');

  record_section.classList.remove('hidden');
  record__verify__icon.querySelector('img').setAttribute('src', 'images/icons/mic
.svg');
  //record__verify__icon.classList.add('record-init');
  login_area.classList.add('hidden');
}

export const renderAudioRecording = () => {
  const record_msg = document.querySelector('.record-msg');

  const record__verify__icon = document.querySelector('.login-
record__verify__icon');

  record_msg.innerHTML = `Fale alguma frase legal ou desabafe por 10 segundos`;
  record__verify__icon.querySelector('img').setAttribute('src', 'images/icons/wav
es.svg');
  record__verify__icon.classList.add('record-init');
}

export const renderAudioRecordCompleted = (firstName, accessResult) => {

  if(accessResult){
    const record_msg = document.querySelector('.record-msg');
    const record__verify__icon = document.querySelector('.login-
record__verify__icon');

    record_msg.innerHTML = `Pronto! Você é o ${firstName} mesmo ;)`;
    record__verify__icon.querySelector('img').setAttribute('src', 'images/icons/c
heck.svg');
    record__verify__icon.classList.add('record-authorized');
  }else{
    const record_msg = document.querySelector('.record-msg');
    const record__verify__icon = document.querySelector('.login-
record__verify__icon');

    record_msg.innerHTML = `Não conseguimos validar. É você mesmo? Tente novament
e... ;)`;
    record__verify__icon.querySelector('img').setAttribute('src', 'images/icons/c
lose.svg');
    record__verify__icon.classList.add('record-denied');
  }

}
```

```
export const renderSignUpData = () => {
  const record__verify__icon = document.querySelector('.login-
record__verify__icon');
  record__verify__icon.classList.remove('record-denied');
  const login_area = document.querySelector('.login-area>div');
  const sign_up_area = document.querySelector('.sign-up-area');
  login_area.classList.add('hidden');
  sign_up_area.classList.remove('hidden');
}

export const getDataNewUser = () => {
  return {
    username: document.getElementById('username-signup').value,
    firstName: document.getElementById('firstname-signup').value,
    lastName: document.getElementById('lastname-signup').value
  };
}

export const renderAudioRecordToSubscribe = () => {

  const signup_area = document.querySelector('.sign-up-area');
  const signup_voice_record_area = document.querySelector('.subscribe-record');

  signup_area.classList.add('hidden');
  signup_voice_record_area.classList.remove('hidden');
}

export const renderAudioRecordingToSubscribe = () => {
  const record_msg = document.querySelector('.record-msg');

  const record__subscribe__icon = document.querySelector('.login-
record__subscribe__icon');

  record_msg.innerHTML = `Fale alguma frase legal ou desabafe por 15 segundos`;
  record__subscribe__icon.querySelector('img').setAttribute('src', 'images/icons/
waves.svg');
  record__subscribe__icon.classList.add('record-init');
}

export const deleteLoginView = () => {
  elements.body.innerHTML = '';
}
```

APÊNDICE O – Maps View – mapsView.js

```
/*
  Descrição: Maps View
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

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


export const renderTheMap = maps => {
    if(!document.querySelector('#map')){
      elements.body.innerHTML = `<div id='map'></div>`;
    }

    maps.buildToolbar();

}

export const renderTheMapVerification = (maps, fence) => {

  /*if(!document.querySelector('#map')){
    elements.body.innerHTML = `<div id='map'></div>`;
  }*/

  maps.showFenceVerification(fence);

  navigator.geolocation.getCurrentPosition(local => {

    if(fence.onFenceArea({lng: local.coords.longitude, lat: local.coords.latitude
})){
      //está dentro da cerca
      document.querySelector('#acionamento-tranca').disabled = false;
      console.log('Está dentro da cerca');
    }else{
      //está fora
      document.querySelector('#acionamento-tranca').disabled = true;
      console.log('Está FORA da cerca');
    }
  });

}
```

APÊNDICE P – Modal View – modalView.js

```
/*
  Descrição: Modal View
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
import { elements } from './base.js';

export const rederTheModal = (modal) =>  {

  const markup = `<div id='${modal.modalId}' class='modal ${modal.classList}'><di
v class='modal-alert'>
                    ${(modal.hasCloseButton)?`<span class='close-
modal'>x</span>`:''}
                    <img src='${modal.iconPath}'>
                    <h4>${modal.title}</h4>
                    <p>${modal.mensage}</p>
                  </div></div>`;

  document.body.innerHTML += markup;
}

export const hideModal = modalId => {

  document.getElementById(modalId).classList.add('hidden');
}

export const showModal = modalId => {

  document.getElementById(modalId).classList.remove('hidden');
}

export const setMensage = ({id, title, mensage}) => {

  const modal = document.getElementById(id);
  modal.querySelector('h4').innerHTML = title;
  modal.querySelector('p').innerHTML = mensage;
}
```

APÊNDICE Q – Base de Componentes – base.js

```
export const elements = {
  map: document.querySelector('#map'),
  body: document.querySelector('body'),
  fence_subscribe_confirm: document.querySelector('#confirm-fence-subscribe'),
  record__verify__icon: document.querySelector('.login-record__verify__icon'),
  audio_record_section: document.querySelector('.login-record'),
  record_msg: document.querySelector('.record-msg'),
  username_field: document.querySelector('#username-login'),
  btn_login: document.querySelector('#btn_login')
};
```

APÊNDICE R – Classe Database – DataBase.js

```
/*
  Descrição: Classe de Banco de Dados
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/

export default class DataBase {

  constructor(data){
    firebase.initializeApp(data);
    this.database = firebase.database();
  }

  async send(where, what){
    return await this.database.ref(where).set(what, err => {
      if(err){
        return err;
      }else{
        return true;
      }
    });
  }

  async receive(where, who){

    return await this.database.ref(where).child(who).get().then( result =>  {
      if (result.exists()) {
        return result.val();
      }
      else {
        console.log("No data available");
        return false;
      }
```

```
    }).catch(err => {
      console.error(err);
    });
  }
}
```

APÊNDICE S – Classe que Armazena a Cerca – FenceRepository.js

```
/*
  Descrição: Classe de Armazenamento da Cerca
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
import Fence from "../Fence";

export default class FenceRepository {

  constructor(database){
    this.database = database;
  }

  async create(usuario, fence){

    return new Promise((resolve, reject) => {
      let result = this.database.send(`usuarios/${usuario}/fence/`, {

        type: (fence.type)?fence.type:"",
        coordenadas: (fence.coordinates)?fence.coordinates:"",
        area: (fence.area)?fence.area:"",
        perimetro: (fence.perimeter)?fence.perimeter:"",
        radius: fence.radius?fence.radius:""
      });

      console.log(result)

      if(result){
        resolve('ok');
      }else{
        reject('err');
      }
    });
  }
  async read(user){
    return new Promise((resolve, reject) => {

      let result = this.database.receive(`usuarios/${user.username}`, 'fence');
```

```
      if(result){

        result.then(res => {
          let myFence = new Fence({
            type: res.type,
            perimeter: res.perimetro,
            area: res.area,
            coordinates: res.coordenadas,
            radius: res.radius
          });

          resolve(myFence);
        })


      }else{
        reject('err');
      }

    });
  }

}
```

APÊNDICE T – Classe que Armazena o Estado da Tranca – LockedRepository.js

```
/*
  Descrição: Classe de Armazenamento do Estado da Tranca
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
export default class LockedRepository {
  constructor(database){
    this.database = database;
  }
  set(user, locked){

    return new Promise((resolve, reject) => {
      let result = this.database.send(`usuarios/${user}/locked`, locked);

      if(result){
        resolve('ok');
      }else{
        reject('err');
```

```
      }
    });
  }
  read(user){
    return new Promise((resolve, reject) => {

      let result = this.database.receive(`usuarios/${user.username}`, 'locked');

      if(result){
        resolve(result);
      }else{
        reject('err');
      }

    });
  }
};
```

APÊNDICE U – Classe que Armazena o Locutor – SpeakerRepository.js

```
/*
  Descrição: Classe de Armazenamento do ID do Locutor
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
export default class SpeakerRepository {
  constructor(database){
    this.database = database;
  }

  create(user, speaker){

    return new Promise((resolve, reject) => {
      let result = this.database.send(`usuarios/${user}/speaker/`, {
        id: (speaker.id)?speaker.id:""
      });

      if(result){
        resolve('ok');
      }else{
        reject('err');
      }
    });
  }

  read(user){
    return new Promise((resolve, reject) => {
```

```
      let result = this.database.receive(`usuarios/${user.username}`, 'speaker');

      if(result){
        resolve(result);
      }else{
        reject('err');
      }

    });
  }
};
```

APÊNDICE V – Classe que Armazena o Usuário – UserRepository.js

```
/*
  Descrição: Classe de Armazenamento do Usuário
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/
export default class UserRepository {
  constructor(database){
    this.database = database;
  }

  create(user){

    return new Promise((resolve, reject) => {
      let result = this.database.send(`usuarios/${user.username}/dados-
pessoais/`, {
        firstName: user.firstName,
        lastName: user.lastName
      });

      if(result){
        resolve(result);
      }else{
        reject('err');
      }
    });
  }

  read(user){
    return new Promise((resolve, reject) => {

      let result = this.database.receive(`usuarios/`, user.username);
```

```
      if(result){
        resolve(result);
      }else{
        reject('err');
      }

    });
  }
};
```

APÊNDICE W – Arquivo de HTML5 – index.html

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://cdn.rawgit.com/mattdiamond/Recorderjs/08e7abd9/dist/record
er.js"></script>
  <title>Geofencing</title>
</head>
<body>
   <div id="map"></div>
   <main>
     <div class="buttons-controll-fence"></div>
   </main>
</body>
<script src="https://www.gstatic.com/firebasejs/8.2.9/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.2.9/firebase-database.js">
</script>
</html>
```

APÊNDICE X – Arquivo de Dependências – 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": "Maychon",
```

```
  "license": "ISC",
  "devDependencies": {
    "@babel/core": "^7.13.8",
    "@babel/preset-env": "^7.10.4",
    "babel-core": "^6.26.3",
    "babel-loader": "^8.2.2",
    "babel-preset-env": "^1.7.0",
    "css-loader": "^3.6.0",
    "dotenv": "^8.2.0",
    "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": "^4.46.0",
    "webpack-cli": "^3.3.12",
    "webpack-dev-server": "^3.11.0"
  },
  "dependencies": {
    "@babel/polyfill": "^7.10.4",
    "axios": "^0.21.1",
    "azure-maps-control": "^2.0.31",
    "azure-maps-drawing-tools": "^0.1.6",
    "file-loader": "^6.0.0",
    "firebase": "^8.2.9",
    "webpack-pwa-manifest": "^4.3.0",
    "workbox-webpack-plugin": "^6.1.5"
  }
}
```

APÊNDICE Y – Arquivo de configurações do WebPack – webpack.config.js

```
const path = require('path');
const webpack = require('webpack');
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');

const dotenv = require('dotenv').config( {
  path: path.join(__dirname, '.env')
} );

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: [
  new HtmlWebpackPlugin({
    filename: 'index.html',
    template: './src/index.html'
  }),
  new MiniCssExtractPlugin({
    filename: 'css/[name].css',
    chunkFilename: 'css/[id].css',
  }),
  new webpack.DefinePlugin( {
    "process.env": dotenv.parsed
  } ),
  new WorkboxPlugin.GenerateSW({
    // O nome do arquivo que será criado.
    swDest: 'sw.js',
    clientsClaim: true,
    skipWaiting: true,
    maximumFileSizeToCacheInBytes: 5000000,
    runtimeCaching: [{
      urlPattern: new RegExp('^https:\/\/fonts\.googleapis\.com/'),
      /**
       * Se tiver versão em cache, ele irá carregar esta versão
       * Caso contrário, aguarda a resposta da rede.
       * */
      handler: 'StaleWhileRevalidate',
      options: {
        //Nome do cache
        cacheName: 'google-fonts',
        //respostas consideradas armazenáveis no cache da APP.
        cacheableResponse: {
          statuses: [0, 200],
        }
      }
    }]
```

```
  }),
  /**
   * Gera um 'manifest.json' para o PWA,
   * realiza o dimensionamento automático dos icones.
   */
  new WebpackPwaManifestPlugin({
    // Nome abreviado para o aplicativo
    short_name: 'GeoLocutor',
    // Nome do aplicativo
    name: 'Speaker and Geofencing',
    description: 'Reconhecimento de Locutor com Uso de Georreferenciamento',
    start_url: '/',
    display: 'standalone',
    background_color: '#D4F3EF',
    theme_color: '#AFE0E5',
    orientation: 'portrait-primary',
    icons: [
      {
        src: path.resolve('src/images/icons/icon-tcc.png'),
        sizes: [96, 128, 180, 192, 256, 512],
        type: 'image/png',
        purpose: 'maskable any'
      }
    ]
  }),

],
module: {
  rules: [
    {
      test: /\.m?js$/,
      exclude: /node_modules/,
      use: {
        loader: 'babel-loader',
        options: {
          presets: ['@babel/preset-env'],
        },
      }
    },
    {
      test: /\.s[ac]ss$/i,
      use: [
        mode !== 'production'
          ? 'style-loader'
          : MiniCssExtractPlugin.loader,
        'css-loader',
        'sass-loader',
      ],
```

```
      },
      {
        test: /\.(png|jpg|svg)$/,
        use: [
          {
            loader: 'file-loader',
            options: {
                name: '[path][name].[ext]',
                context: path.resolve(__dirname, "src/"),
                outputPath: '/',
                publicPath: '../',
                useRelativePaths: true
            }
          }
        ]
      }
    ],
  }
});
```

APÊNDICE Z – Código em execução no ESP32 – Esp32.ino

```
/*
  Descrição: Execução no ESP32
  Autor: Maychon Douglas // @maychondouglas
  Data: 2021/1
*/


//Importando as bibliotecas que serão utilizadas
#include <WiFi.h>
#include <FirebaseESP32.h>

//Informações referentes a rede WiFi
#define WIFI_SSID "XXXXXXXX"
#define WIFI_PASSWORD "XXXXXXXX"

//Informações para autenticação no Firebase
#define FIREBASE_HOST "tcc-2-1e9c6-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "iujkpJUVXc2snvRdQ0cyaopb63URqYwseC5MOlfM"

//Led que mostra estado da TRANCA (Aceso => Aberto; Apagado => Fechado)
#define LOCKED_LED 2


FirebaseData fbdo;

unsigned long sendDataPrevMillis = 0;
```

```
String path = "/usuarios/maychondouglas";

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(){

  //AGUARDAR 3 SEGUNDOS PARA ESPERAR A REQUISIÇÃO SER COMPLETADA
  if (millis() - sendDataPrevMillis > 3000)
  {
    sendDataPrevMillis = millis();
```

```
    if(Firebase.getBool(fbdo, path + "/locked")){
      if(fbdo.boolData() == 1){
        //apagar a LED informando que a Tranca está Fechada
        digitalWrite(LOCKED_LED, LOW);
        Serial.println("-------------------------------------");
        Serial.println("Fechado!");
        Serial.println("-------------------------------------");
        Serial.println();
      }else{

        //ascender 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();
    }
  }
}
```