UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Vinícius César Jasinski

Desenvolvimento de um aplicativo assistivo para pessoas de baixa visão ou cegas

Blumenau
2023

Vinícius César Jasinski

Desenvolvimento de um aplicativo assistivo para pessoas de baixa visão ou cegas

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Orientador, Dr. Mauri Ferrandin

Blumenau
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Jasinski, Vinícius César
Desenvolvimento de um aplicativo assistivo para pessoas de baixa visão ou cegas / Vinícius César Jasinski ; orientador, Mauri Ferrandin, 2023.
56 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Aplicativo Android. 3. Acessibilidade. 4. Deficiência Visual. 5. Reconhecimento de Objetos. I. Ferrandin, Mauri. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Vinícius César Jasinski

Desenvolvimento de um aplicativo assistivo para pessoas de baixa visão ou cegas

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 23 de março de 2023.

Banca Examinadora:

Documento assinado digitalmente
Mauri Ferrandin
Data: 13/06/2023 20:44:24-0300
CPF: ***.580.869-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Mauri Ferrandin, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Carlos Roberto Moratelli
Data: 14/06/2023 08:50:50-0300
CPF: ***.999.369-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Roberto Moratelli, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Maiquel de Brito
Data: 14/06/2023 08:44:12-0300
CPF: ***.657.150-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Maiquel de Brito, Dr.
Universidade Federal de Santa Catarina

Dedico este trabalho aos meus queridos pais e irmãos, que
sempre me incentivaram a ir mais longe.

AGRADECIMENTOS

Agradeço aos meus pais, que estiveram comigo em todos os momentos, apoiando, incentivando e repassando seus ensinamentos.

Agradeço também ao professor Dr. Mauri Ferrandin, pelas críticas e sugestões que proporcionaram melhorar consideravelmente a qualidade deste trabalho.

Agradeço aos meus irmãos pelos ensinamentos passados desde a infância, que ajudaram a construir meu caráter e valores. Ao meu irmão Éverton, por todo o acompanhamento durante a minha caminhada na graduação, orientações e pelo incentivo dado para a finalização deste trabalho. Ao meu irmão Márcio, por todo o suporte na área de tecnologia, ensinamentos e explicações que me ajudaram durante o curso e minha carreira.

”Many of life’s failures are people who did not realize how close they were to success when they gave up.” (EDISON, s.d.)

RESUMO

No Brasil, existem aproximadamente 6.5 milhões de pessoas cegas ou com deficiência visual severa. Tendo em vista esse aspecto, foram analisados 2068 aplicativos com mais de 10 milhões de downloads no Brasil, e destes, apenas 1.64% possui um nível básico de acessibilidade. Baseado nisso, o presente trabalho descreve uma solução através do desenvolvimento de um aplicativo móvel para Android, o qual tem como objetivo auxiliar pessoas deficientes visuais a ter uma rotina mais independente. O aplicativo possibilita que o usuário assistido possa solicitar a ajuda de um voluntário remotamente, a qualquer momento, através de uma chamada de vídeo, ou, caso não deseje auxílio de uma pessoa terceira, ele também pode utilizar o reconhecimento de objetos para identificar o artefato de seu interesse. Atrelado a este desenvolvimento, foi utilizado como tecnologias auxiliares o Firebase, para armazenamento de dados, envio de notificações e autenticação, a biblioteca da Agora, para realizar uma chamada de video em tempo real entre o usuário voluntário e o usuário assistido e a biblioteca do TensorFlow para reconhecimento de objetos em tempo real.

Palavras-chave: Deficiência visual; Acessibilidade; Aplicativo móvel para Android; Reconhecimento de objetos; Firebase; Chamada de vídeo.

ABSTRACT

In Brazil, there are approximately 6.5 million people who are blind or have severe visual impairment. Considering this aspect, 2068 applications with more than 10 million downloads in Brazil were analyzed, and of these, only 1.64% have a basic level of accessibility. Based on this, the present work describes a solution through the development of a mobile application for Android, which aims to help visually impaired people to have a more independent routine. The application allows the assisted user to request the help of a volunteer remotely, at any time, through a video call, or, if he doesn't want help from a third person, he can also use object recognition to identify the artifact. of your interest. Connected to this development, Firebase was used as auxiliary technologies for data storage, sending notifications and authentication, Agora library, was used to perform a real-time video call between the volunteer user and the assisted user, and TensorFlow library was used for recognition of objects in real time.

Keywords: Visual impairment; Accessibility; Mobile application for Android; Object recognition; Firebase; Video call.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
DDD	Domain Driven Design
FCM	Firebase Cloud Messaging
$IBGE$	Instituto Brasileiro de Geografia e Estatistica
MVP	Minimum Viable Product
$MVVM$	Model; View; View Model
$NoSQL$	Not Only Structured Query Language
$REST$	Representational State Transfer
RTC	Real Time Connection
RTD	Real Time Database
$SDRTN$	Software-Defined Real-Time Network
SDK	Software Development Kit
$SIFT$	Scale-Invariant Feature Transform
$SOLID$	Single responsiblity; Open-closed; Liskov substitution; Interface segregation; Dependency inversion
UID	Unique Identifier

SUMÁRIO

1 INTRODUÇÃO

Da invenção da internet à Indústria 4.0, a velocidade do desenvolvimento de novas tecnologias, assim como a maneira como elas se inserem no cotidiano das pessoas, cresce de maneira constante. Entretanto, devido a essa velocidade de desenvolvimento elevada, as empresas e soluções tecnológicas acabam marginalizando temas como a acessibilidade. Smartphones, que passaram a ser uma tecnologia essencial considerada pelos jovens (DA SILVA; PEREIRA, 2015), são repletos de aplicativos, os quais apesar de serem muito difundidos, em sua maioria não possuem suportes adequados a acessibilidade (BIGDATACORP, 2020).

Segundo a BigDataCorp (BIGDATACORP, 2020), a qual realizou um estudo com 2068 aplicativos com mais de 10 milhões de downloads no Brasil, notou-se que apenas 1.64% destes aplicativos possuem ao menos metade dos elementos de interface visual acessíveis. Este número se torna mais alarmante quando é levado em consideração o número de pessoas com deficiência visual severa e cegas no Brasil, que segundo o censo do *IBGE* de 2010 (IBGE, 2010), chega a aproximadamente 6.5 milhões de pessoas.

De tarefas cotidianas até afazeres mais complexos, as pessoas com deficiência visual severa possuem grande dificuldade durante o seu dia a dia. Isso ocorre principalmente para realizar atividades de maneira independente, precisando, muitas vezes, do auxílio de alguma pessoa próxima. Atividades como ir ao mercado, tomar um remédio ou até mesmo escolher uma roupa são muito complicadas de serem feitas com a debilidade da visão.

Nesse contexto, a proposta deste trabalho consiste em desenvolver uma solução que auxilie os dados evidenciados pelo *IBGE* (IBGE, 2010) e BigDataCorp (BIGDATACORP, 2020), associados com a dificuldade dessas pessoas em realizar suas tarefas cotidianas de maneira independente. Para abranger esses três problemas, este trabalho consiste em desenvolver um aplicativo Android acessível, destinado a auxiliar pessoas com deficiências visuais severas. Ele será desenvolvido com o intuito de ajudá-las a terem maior independência para realizar atividades cotidianas, como encontrar locais, itens e ler embalagens. Isso será feito através de reconhecimento de objetos e chamadas audiovisuais entre duas pessoas.

1.1 OBJETIVOS

Esta seção é destinada aos objetivos gerais e específicos que fundamentam este trabalho.

1.1.1 Objetivo Geral

O presente trabalho possui como objetivo geral auxiliar pessoas cegas através do desenvolvimento de um aplicativo nativo Android na linguagem Kotlin, que possua um

método de comunicação audiovisual entre duas pessoas e que possua uma alternativa de reconhecimento de objetos, utilizando um modelo pré-treinado.

1.1.2 Objetivos Específicos

- Desenvolver um aplicativo com uma interface intuitiva com suporte a acessibilidade;
- Proporcionar que o usuário com deficiência visual possa criar chamadas de vídeo com voluntários;
- Tornar o projeto open source para que qualquer pessoa possa contribuir para a evolução do projeto;
- Implementar um método de notificação de chamada;
- Implementar um modelo de reconhecimento de objetos, com suporte a acessibilidade e com o objeto identificado descrito em português;
- Utilizar das boas práticas de arquitetura de software.

1.2 ESTRUTURA DO DOCUMENTO

Este trabalho está dividido em 5 capítulos. Sendo eles a Introdução, demonstrada na Seção 1, a qual corresponde a seção introdutória do trabalho, a Revisão Conceitual, presente na Seção 2, que corresponde aos conceitos amplamente aplicados e aprendidos durante o desenvolvimento do aplicativo nativo Android, a Metodologia, apresentada na Seção 3, que aborda a metodologia utilizada para o desenvolvimento do aplicativo, os Resultados, demonstrados na Seção 4, o qual representa a discussão e demonstração dos resultados obtidos e as Conclusões, descritas na Seção 5, nas quais serão abordadas as conclusões do trabalho assim como pontuações de possíveis melhorias.

2 REVISÃO CONCEITUAL

Com o intuito de cumprir os objetivos definidos, neste capítulo serão apresentados os conceitos aplicados ao projeto desenvolvido. Juntamente a isso, serão trazidos trabalhos similares e sua diferenciação.

2.1 TRABALHOS CORRELACIONADOS

Com o objetivo de analisar a solução proposta, foi realizado um estudo referente a soluções do mesmo segmento de acessibilidade existentes no mercado e no meio acadêmico. Após a análise individual de cada caso, estes serão analisados em um comparativo com o trabalho proposto.

2.1.1 Be My Eyes

O aplicativo Be My Eyes (EYES, 2023), é o aplicativo mais conhecido quando se trata de auxílio para pessoas que necessitam de apoio visual. Tendo mais de 6 milhões de voluntários e cerca de 550 mil pessoas cegas ou de baixa visão, o aplicativo conecta um voluntário com um usuário que necessita de apoio visual através de uma chamada de vídeo. Entretanto, apesar deste aplicativo possuir um alto volume de usuários, seu suporte para o português é bastante defasado. Como seu principal idioma é o inglês e seu suporte também, é criada uma barreira entre usuários brasileiros que desejem utilizar a plataforma e necessitem de um auxílio.

2.1.2 Aplicativo móvel para deficientes visuais

O projeto desenvolvido pelo Darren Tan Yung Shen (SHEN, 2020), como trabalho de conclusão de curso, visa fazer o reconhecimento de objetos e realização de ligações para pessoas com deficiência visual. Neste projeto, o autor visa aplicar um modelo de machine learning ao seu aplicativo, juntamente com um modelo de ligação semelhante ao já existente nativamente, possibilitando ligar e salvar um novo contato. Além disso, o autor também realiza a integração com sistemas de text-to-speech (texto para fala), para que o usuário possa ouvir qual foi o objeto reconhecido pela sua câmera.

2.1.3 BlindTool

O aplicativo BlindTool (COHEN, 2015), desenvolvido por Joseph Paul Cohen, Ph.D em ciências do computação pela Universidade de Massachusetts Boston, busca, além do reconhecimento dos objetos, também propor uma vibração no dispositivo durante o seu reconhecimento de objeto. Sendo assim, quando um objeto é reconhecido, além da sua identificação ser lida, também é realizada uma vibração no dispositivo, a qual aumenta de intensidade conforme a probabilidade de acerto no reconhecimento for alta. Cohen

utiliza Rede Neural Convolucional embarcada no aplicativo, a qual pode reconhecer até mil classes baseadas na ImageNet (LI, 2022).

2.1.4 Reconhecimento de Objetos e Imagens Baseado em Android

O trabalho (BARI; KAMBLE; TAMHANKAR, 2014), consiste em um estudo e abordagem de métodos de reconhecimento de imagem e objetos, nele foi implementado um algoritmo conhecido como *SIFT*(Scale-Invariant Feature Transform), um poderoso algoritmo de visão computacional para reconhecimento e descrição de imagens (BARI; KAMBLE; TAMHANKAR, 2014). Sua implementação de reconhecimento em tempo real, busca encontrar objetos que estejam em movimento na frente do usuário e disparar um alarme caso isso aconteça, para que este tome as devidas precauções até que este objeto pare.

2.1.5 Comparativo

Para fins comparativos, segue abaixo uma tabela que demonstra as diferentes funcionalidades entre os projetos analisados.

Tabela 1 – Comparativo entre diferentes aplicações inclusivas.

Projeto	Chamada de vídeo	Ligação	Reconhecimento de Objetos	Vibração	Suporte em Português
Solução Proposta	Contém	Não Contém	Contém	Não Contém	Contém
BeMyEyes	Contém	Não Contém	Não Contém	Não Contém	Não Contém
BlindTool	Não Contém	Não Contém	Contém	Contém	Não Contém
Aplicativo móvel para deficientes visuais	Não Contém	Contém	Contém	Contém	Não Contém
Reconhecimento de Objetos e Imagens baseado em Android	Não Contém	Não Contém	Contém	Contém	Não Contém

Fonte: O Autor (2022).

Através dessa tabela, é possível verificar que o aplicativo a ser desenvolvido, denominado Guie-me, não contempla todos os recursos que os outros projetos analisados possuem. Apesar disso, uma de suas principais diferenciações é a quebra da barreira do idioma em comparação com as outras soluções analisadas. Essa diferenciação é relevante pois, ainda que os outros projetos promovam funcionalidades semelhantes, para o uso deles surge o requisito da comunicação em inglês ou a compreensão deste idioma, devido a eles não possuírem suporte para o português. Por este motivo, este trabalho também promove uma tecnologia que possibilita as pessoas se ajudarem com o menor número de barreiras possíveis.

2.2 FERRAMENTAS UTILIZADAS PARA O DESENVOLVIMENTO

Durante o desenvolvimento do projeto foram utilizadas as seguintes ferramentas, Kotlin, Next.js, *API REST*, Agora SDK, Firebase, TensorFlow, Talkback e Kanban, os

quais serão descritos nas subsecções a seguir.

2.2.1 Kotlin

Criado em 2011, a linguagem Kotlin foi oficialmente adotada pela Google como a linguagem padrão para a programação Android em 2017. Esta linguagem, surgiu como uma substituta a linguagem Java no contexto de programação de aplicativos, devido a ferramentas disponíveis na linguagem como *lambdas*, *smart casts* e *extensions*, além de promover um código mais seguro a *null pointer exceptions*. (MARTINEZ; GOIS MATEUS, 2022).

Dentro da linguagem, os *lambdas* são basicamente uma maneira concisa de declarar uma função, os *smart casts* funcionam como uma transformação inteligente entre diferentes tipos de dados sem que seja necessário explicitá-los, e os *null pointer exceptions* são erros ocorridos devido a falta de uma tratativa adequada para um dado que pode ser nulo.

O Kotlin é uma linguagem orientada a objetos, de tipo estático e que é baseada em Java. Isso possibilita a sua interoperabilidade com códigos desenvolvidos em Java. Ademais o Kotlin é mais expressivo e conciso, aumentando a produtividade de seus desenvolvedores. Os códigos escritos nessa linguagem possuem como recursos *threads*, um recurso utilizado para a paralelização de tarefas na programação, assíncronas de uso facilitado o que acarreta em um código 20% menos propensos a falha (GOOGLE, 2022). Por fim, esta linguagem permeou todo o desenvolvimento do aplicativo, visto que o aplicativo foi desenvolvido utilizando ela.

2.2.2 Next.js

Desenvolvido pela Vercel, o Next.js é um *framework* do React, ele possui uma abstração de um código genérico para o desenvolvimento de aplicações. O React é uma biblioteca da linguagem JavaScript, e possui como objetivo facilitar a criação de interfaces para o usuário no desenvolvimento Web.

O Next.js possui como propósito resolver requisitos comuns de aplicativos como roteamento, busca de dados e integrações. Devido a isso, durante o desenvolvimento do aplicativo, este *framework* foi utilizado para a criação de uma *API REST*, a qual será detalhada na sequência. Essa ferramenta ficou responsável pela geração de identificadores de acesso para os usuários entrarem nas salas de videochamada e também para o envio de notificações aos usuários.

2.2.3 API REST

A *API* (Application Programming Interface) representa uma interface de programação de aplicação, a qual é utilizada como recurso para acessar dados e serviços. Deste modo, uma *API* pode ser compreendida como um contrato, que padroniza e simplifica o

acesso às informações. Assim que o contrato entra em vigor, a consistência e a previsibilidade dos dados retornados aumenta, tornando a conexão entre provedor e consumidor mais eficiente (JACOBSON; BRAIL; WOODS, 2011).

O *REST* (Representational State Transfer) constitui um modelo de arquitetura a ser seguido durante o desenvolvimento de uma *API*. Para uma *API* se enquadrar na arquitetura *REST*, antes de sua criação, deve-se ter um conhecimento prévio das necessidades do sistema. Unido a esse requisito, a *API* também deve seguir algumas restrições (FIELDING, 2000).

Segundo Fielding (2000), a primeira restrição se dá com o estilo de arquitetura cliente-servidor, em que devem ser separadas as responsabilidades da interface do usuário das preocupações de armazenamentos de dados, com o objetivo de simplificar os componentes do servidor e aprimorar a escalabilidade. Para a segunda regra, durante a comunicação entre cliente-servidor não devem ser armazenados estados, ou seja, a cada requisição feita o servidor deve receber todos os dados necessários para processá-la.

A terceira condição é que quando um servidor puder armazenar uma resposta em cache, ou seja, guardar temporariamente a mesma resposta, ele pode adicioná-lo para caso o usuário necessite em solicitações equivalentes posteriores. A quarta regra, a qual distingue o estilo arquitetônico *REST* de outras arquiteturas, enfatiza a necessidade de uma interface uniforme entre componentes, o qual padroniza o método de comunicação entre servidor, cliente e seus intermediários.

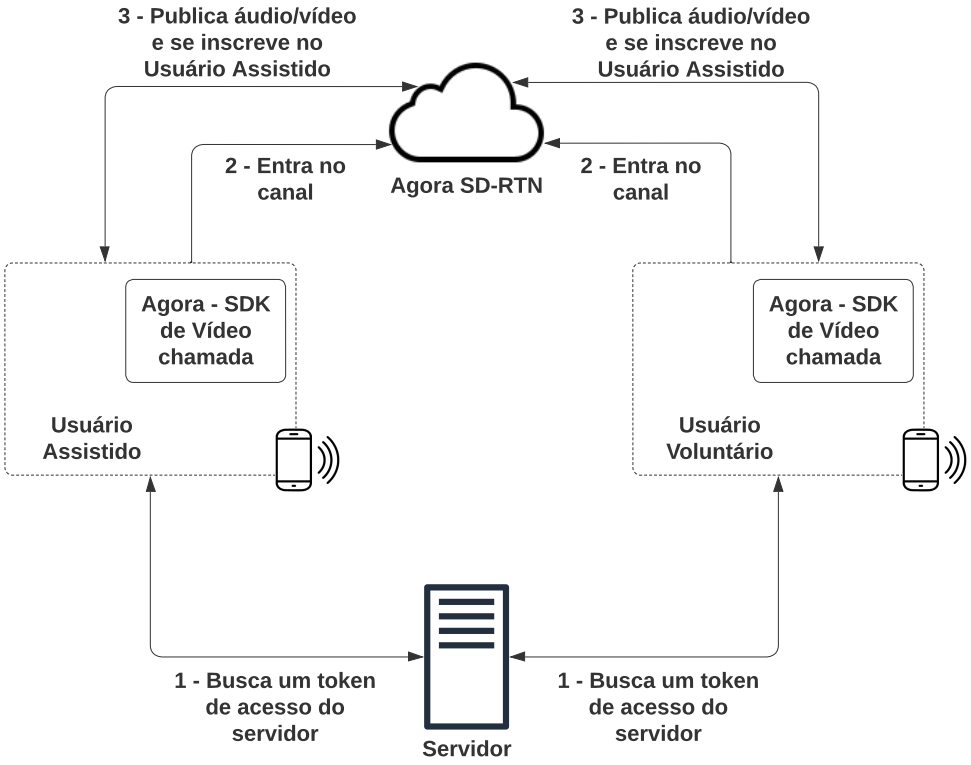
A quinta e última restrição obrigatória consiste na criação de camadas, em que cada uma deve possuir responsabilidades distintas. O cliente não tem conhecimento de quantas camadas existem no total ou quais são os servidores intermediários utilizados durante a requisição.

Em síntese, estes dois conceitos são utilizados para que o cliente possa enviar solicitações ao servidor no formato de dados e que o servidor realize uma devolutiva destes dados para o cliente. Ao utilizar esses recursos, obtém-se um ganho de escalabilidade nas interações dos componentes, redução de acoplamento, reforça a segurança e encapsula sistemas legados (PAUTASSO; WILDE, 2013).

2.2.4 Agora SDK

A empresa Agora, especializada em comunicação em tempo real, fornece uma biblioteca que pode ser implementada em Kotlin. A empresa desenvolveu suas bibliotecas a partir de seu *SDRTN* (Software-Defined Real-Time Network), o qual será descrito abaixo, e a partir de sua biblioteca é possível enviar informações para a sua *API REST* para instanciar uma chamada de vídeo em tempo real. A tecnologia utilizada foi o *Video Call SDK* e seu funcionamento pode ser visto na Figura 1.

Figura 1 – Funcionamento do SDK da Agora integrado ao aplicativo.



Fonte: (AGORA, 2022).

O $SDRTN$, é uma abordagem de rede que busca utilizar controladores baseados em softwares ou APIs para poder se comunicar com uma estrutura física, conhecida como *hardware*, e direcionar o tráfego em uma rede.

2.2.5 Firebase

O Firebase é uma plataforma da Google para desenvolvimento de aplicativos que fornece serviços de banco de dados, autenticação, envio de notificações ao dispositivo e análise de dados. Seus serviços são hospedados em um servidor virtual e possuem uma integração bem facilitada com o Android. No desenvolvimento do aplicativo, foram utilizadas as ferramentas RTD (Real Time Database), o Firebase Authentication e o FCM (Firebase Cloud Messaging), descritos nas subsecções a seguir.

2.2.5.1 Real Time Database

O RTD ou Real Time Database, é uma ferramenta de banco de dados não relacional e de sincronização de dados em tempo real, o qual notifica qualquer alteração a todos os dispositivos conectados em milissegundos. Como ele é um banco de dados $NoSQL$, ou seja, não relacional, há uma ampla flexibilidade para modificações e liberdade para a

criação de tabelas. Isso ocorre pois não é necessário a criação de uma chave primária para esses valores e seus dados não precisam armazenar a mesma quantidade de itens.

Dado que a sua sincronização é em tempo real, é possível identificar mudanças dentro da aplicação sem que seja feita uma nova requisição de dados para o *RTD*, o que faz com que não seja necessário realizar alguma atualização manual ou que o usuário necessite fechar a aplicação para obter dados atualizados.

2.2.5.2 Firebase Authentication

O Firebase Authentication funciona como uma ferramenta de identificação da identidade do usuário. Através dela, é possível salvar os dados e preferências de um usuário, assim como fornecer uma experiência mais personalizada para ele. Seu uso em conjunto com o *RTD* torna possível fazer a identificação e diferenciação entre usuários voluntários e assistidos.

Ademais, essa ferramenta fornece bibliotecas de autenticação com uso simplificado e ampla documentação para validar o acesso dos usuários dentro do aplicativo. Dentro desse escopo, ela fornece recursos como autenticação com senha, número de telefone, via e-mail Google, Facebook e Twitter.

2.2.5.3 Firebase Cloud Messaging

O *FCM* ou Firebase Cloud Messaging, fornece uma solução de envio de mensagens para dispositivos de uma maneira confiável e sem custo. Através dessa ferramenta é possível enviar *push notifications* aos usuários, notificando o usuário voluntário quando um usuário assistido está solicitando ajuda.

O *push notification* é o nome do método utilizado para notificar um usuário via celular. Através dele, é possível encaminhar uma notificação no dispositivo do usuário sem que haja nenhuma ação requisitante. Sendo assim, o *push notification* viabiliza que o usuário voluntário seja notificado assim que um usuário assistido solicitar auxílio.

2.2.6 TensorFlow

O TensorFlow é uma biblioteca de código aberto para aprendizado de máquina e computação numérica, desenvolvida e mantida pela Google. Essa biblioteca facilita a criação e uso de modelos de aprendizado de máquina para diferentes plataformas. Dentre seus principais exemplos, existem modelos de reconhecimento de imagens estáticas e de dados em tempo real.

O TensorFlow utiliza redes neurais para seu funcionamento. Essas redes foram desenvolvidas para simular o mecanismo de aprendizado dos seres biológicos, como os neurônios humanos, e são aplicadas em tarefas que utilizam o aprendizado de máquina (AGGARWAL, 2018). Unido a isso, ele emprega cálculos usando um modelo semelhante

a um fluxo de dados e mapeia-os em uma ampla variedade de diferentes hardwares e plataformas (ABADI *et al.*, 2016), como por exemplo o Android.

Para este projeto, foi utilizado um modelo pré-treinado, disponibilizado pela própria plataforma, denominado MobileNet. Este modelo, foi treinado utilizando a base de dados da ImageNet, que possui mais de 14 milhões de imagens de alta qualidade cadastradas, das quais cada imagem é associada a um grupo de categoria. Com base nisso, o MobileNet foi treinado e utilizou estas imagens associadas com as suas respectivas categorias. O objetivo deste modelo é auxiliar na identificação dos objetos assim como prover a probabilidade de acerto de sua predição.

2.2.7 Talkback

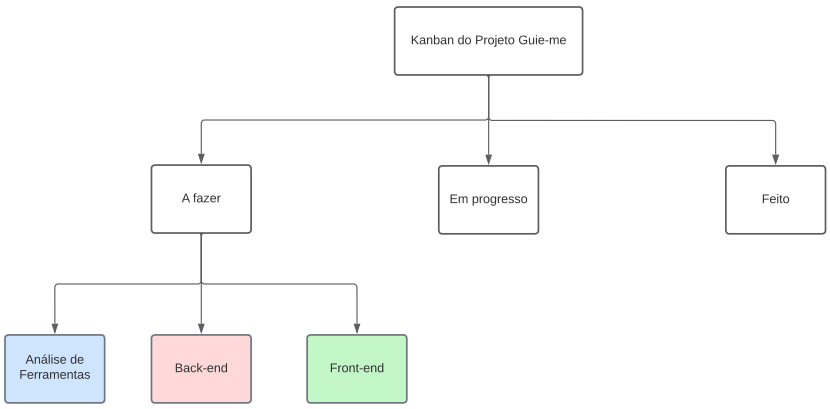
O Talkback é o aplicativo de leitor de tela nativo da Google sugerido para a acessibilidade. Este aplicativo já vem instalado no dispositivo juntamente com todo o sistema operacional Android, por este motivo ele é a principal ferramenta de acessibilidade utilizada como referência durante o desenvolvimento.

O Talkback funciona como uma ferramenta base utilizada por pessoas não videntes e sua aplicação consiste em realizar uma transcrição dos elementos dispostos na tela do celular para fala, ou seja, cada elemento do aplicativo é lido de uma maneira verbal, para que o usuário compreenda o que existe na tela e para onde deseja navegar.

2.2.8 Kanban

O Kanban, desenvolvido por David J. Anderson, é um sistema de gestão visual, que visa medir e acompanhar o desenvolvimento de tarefas. Através desse modelo, o projeto pode ser dividido em três principais colunas, tarefas a fazer, tarefas em progresso e tarefas feitas (ANDERSON, 2010). Por este motivo, utilizou-se o diagrama da Figura 2 para realizar o desenvolvimento do projeto Guie-me.

Figura 2 – Estrutura do Kanban do aplicativo Guie-me.



Fonte: O Autor (2022).

2.3 ARQUITETURA UTILIZADA NO DESENVOLVIMENTO

Para que o projeto tivesse boas práticas de desenvolvimento e fosse de fácil manutenção ele foi desenvolvido seguindo os princípios que serão descritos nas próximas subsecções:

- *Clean Architecture*;
- *SOLID*;
- *MVVM*;
- *Domain Driven Design.*

2.3.1 *Clean Architecture*

O conceito do *Clean Architeture* (MARTIN, 2019), consiste em um padrão organizacional do projeto de software promovido por Robert C. Martin. Seu objetivo com essa arquitetura é promover, através de uma abstração, que os sistemas sejam independentes das interfaces de usuário, banco de dados, *frameworks*, agentes externos e sejam testáveis. Sendo assim, o projeto deve depender apenas da linguagem de programação utilizada para desenvolvê-lo.

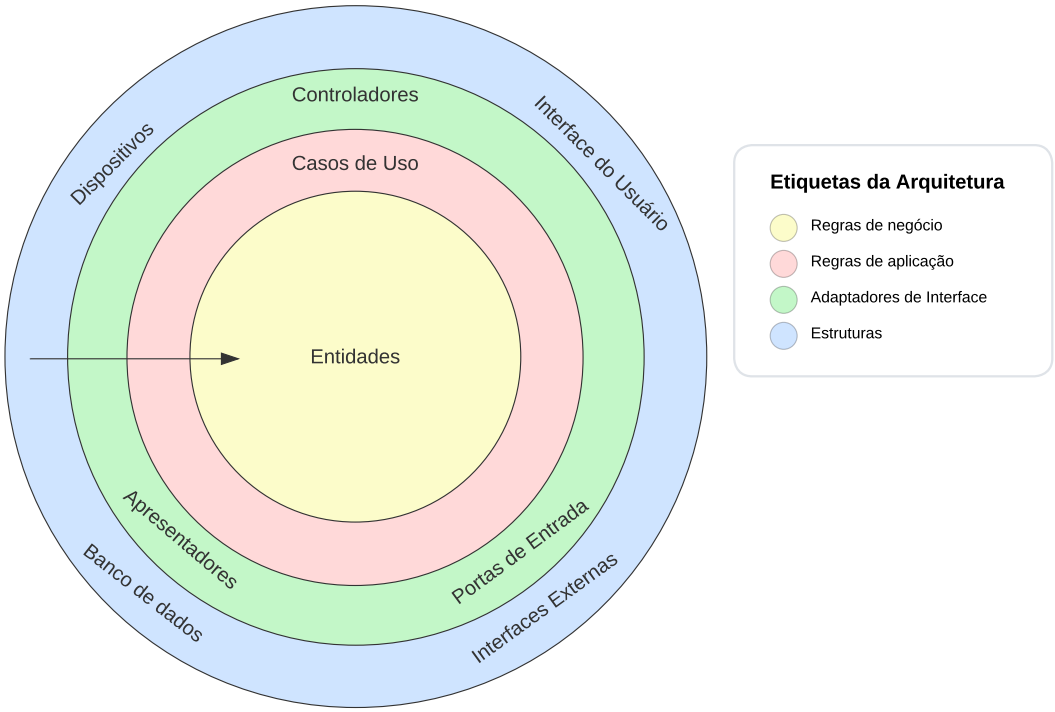
Para que isso seja possível, o projeto deve estar segmentado em camadas, nas quais as camadas mais internas são responsáveis pelas regras de negócio aplicadas e as camadas mais externas são os mecanismos que auxiliam para que estas regras sejam de fato aplicadas. A demonstração das segmentações sugeridas pelo Robert C. Martin podem ser vistas na Figura 3.

As ”Entidades” são os locais nos quais as regras de negócio do projeto estão inseridas, nessa camada é feita a definição da regra de como funcionará a aplicação. Para o caso do aplicativo, as regras de negócio definidas foram as seguintes:

- Um usuário voluntário quando disponível poderá ser notificado de um pedido de chamada;
- Cada sala terá um identificador único para entrar no canal;
- Cada sala terá no máximo duas pessoas;
- O usuário assistido pode enviar a notificação de solicitação de ajuda de um voluntário.

Essas regras funcionam como a base da aplicação, tendo em vista isso, elas podem sofrer pequenas ou nenhuma mudança durante o projeto. Já os ”Casos de Uso” funcionam como um intermediador entre as entidades e as solicitações externas do aplicativo. Por este motivo, eles gerenciam as regras de negócio entre a camadas das entidades e dos adaptadores de interface, tratando de uma maneira que seja compreendida entre as duas camadas. Na aplicação, os casos de uso são utilizados para enviar os dados para a camada das entidades e tratar sua resposta que será retornada até a interface do usuário.

Figura 3 – A Arquitetura Limpa



Fonte: (MARTIN, 2019).

Os ”Adaptadores de Interface” são responsáveis por realizar a conversão dos dados para que eles possam ser enviados para os casos de uso. Sendo assim, eles podem encapsular ou desencapsular um objeto para que este fique no formato adequado para ser enviado para as entidades. Já a ”Estrutura” é a camada mais externa da arquitetura, e também, onde podem ocorrer mais mudanças. Nessa camada ficam as estruturas e ferramentas como o banco de dados utilizado, frameworks e as telas com seus componentes.

2.3.2 SOLID

O *SOLID*, sigla que será descrita a seguir, (MARTIN, 2000) é um conceito desenvolvido por Robert C. Martin após observar que cinco princípios da orientação a objetos e de design de código que poderiam se encaixar neste acrônimo. Através da utilização desses princípios, é possível escrever um código mais limpo, assim como descrito em seu outro livro, O Código Limpo (MARTIN, 2019). Os principais ganhos da utilização dessas duas práticas durante o desenvolvimento são:

- Um software mais escalável e robusto, visto que mudanças e melhorias são facilmente visualizadas e implementadas;
- Fácil manutenção, pois se as responsabilidades estão bem definidas e suas regras

também, é possível realizar a manutenção e melhoria do código utilizando um baixo esforço.

O primeiro princípio, conhecido como Single Resonsability Principle ou Princípio da Responsabilidade Única, está atrelado a que cada classe deve ser especializada em uma funcionalidade e possuir apenas uma responsabilidade única dentro do código. O segundo é denominado Open/Closed Principle ou Princípio Aberto-Fechado, representa que os objetos devem ser abertos para a extensão, mas fechados para modificação, ou seja novos componentes e recursos inseridos no código devem estender de componentes antigos e não alterá-los em seu código fonte original.

O terceiro é chamado Liskov Substitution Principle ou Princípio da substituição de Liskov, declara que uma classe derivada deve ser substituível por sua classe base, ou seja se uma classe é derivada de outra, deve ser possível substituir as classes derivadas pela sua classe pai. O quarto é denominado Interface Segregation Principle ou Princípio da Segregação da Interface, este princípio afirma que uma classe não deve ser forçada a implementar interfaces e métodos que não serão utilizados. Este princípio demonstra que as interfaces devem ser mais específicas, sendo adaptadas para cada uso e não deve ser uma interface genérica que possui métodos que não sejam utilizados por outras classes.

O quinto princípio é chamado Dependency Inversion Principle ou Princípio da Inversão de Dependência, estabelece que para evitar acoplamento entre classes alto e baixo nível, estas devem ser desacopladas através da abstração, que pode ser alcançada com o uso de interfaces, por exemplo.

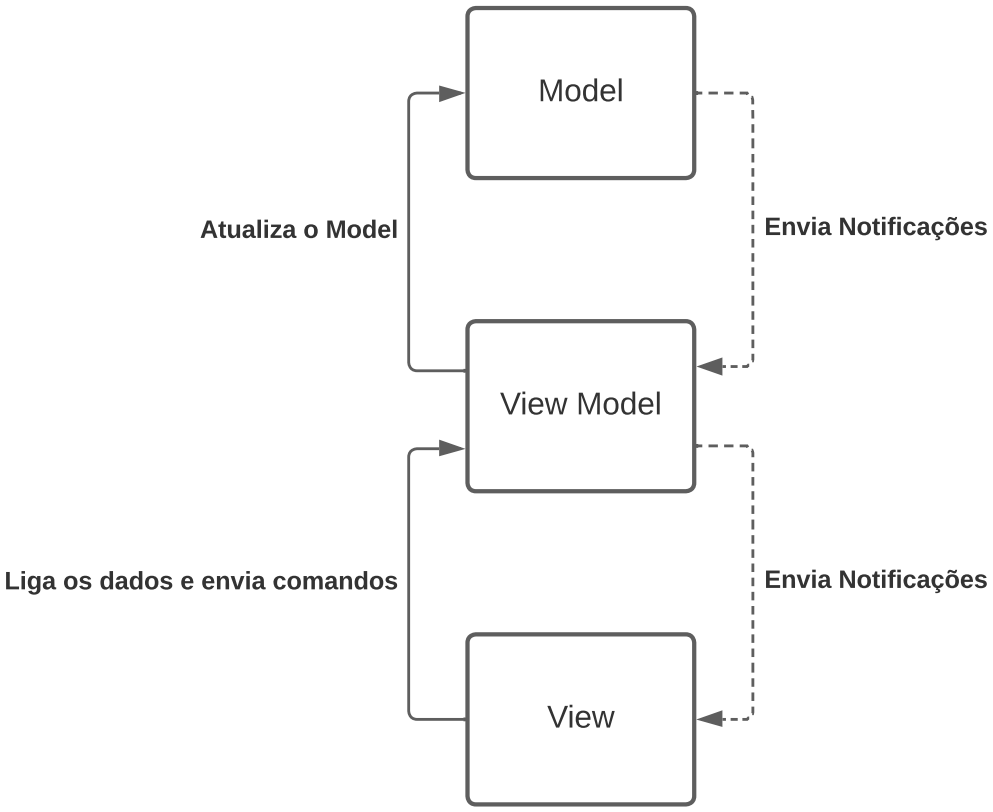
2.3.3 MVVM

O primeiro modelo de arquitetura utilizado no projeto é o $MVVM$ (Model, View, View Model). Essa arquitetura é inspirada pelo conceito descrito na *Clean Architeture*, de Robert C. Martin, o qual promoveu a abstração do *software* em camadas de desenvolvimento. A arquitetura $MVVM$, a qual é recomendado pela Google, funciona como a implementação prática dessa abstração sugerida. Sendo assim, este modelo faz parte do conceito de arquitetura limpa e é recomendado seu uso em todas as aplicações nativas Android.

O funcionamento dessa arquitetura pode ser visto na Figura 4. Esse modelo de arquitetura permite uma visão clara de segmentação entre a interface do usuário, a lógica de apresentação e seus dados. Dessa maneira, é possível separar as responsabilidades de cada classe, ter um código desacoplado e promover melhor evolução e manutenção dele (ALEXANDRU, 2022).

A primeira camada denominada Model ou modelo, no $MVVM$, é responsável pelo encapsulamento das validações, regras de negócio e dados. Ele possui como papel definir o que a aplicação visa suprir. Dentro do modelo se encontram os repositórios, objetos de transferência de dados e entidades. A segunda camada chamada View ou visualização, ou

Figura 4 – A arquitetura $MVVM$



Fonte: (MICROSOFT, 2012).

camada de apresentação, é representada por uma camada abstraída de lógica, responsável por gerenciar os componentes e por definir o que o usuário verá em sua tela. Cada View possui seu próprio View Model e ela é responsável por escutar informações advindas do View Model, assim como encaminhar para ele as interações do usuário na respectiva tela.

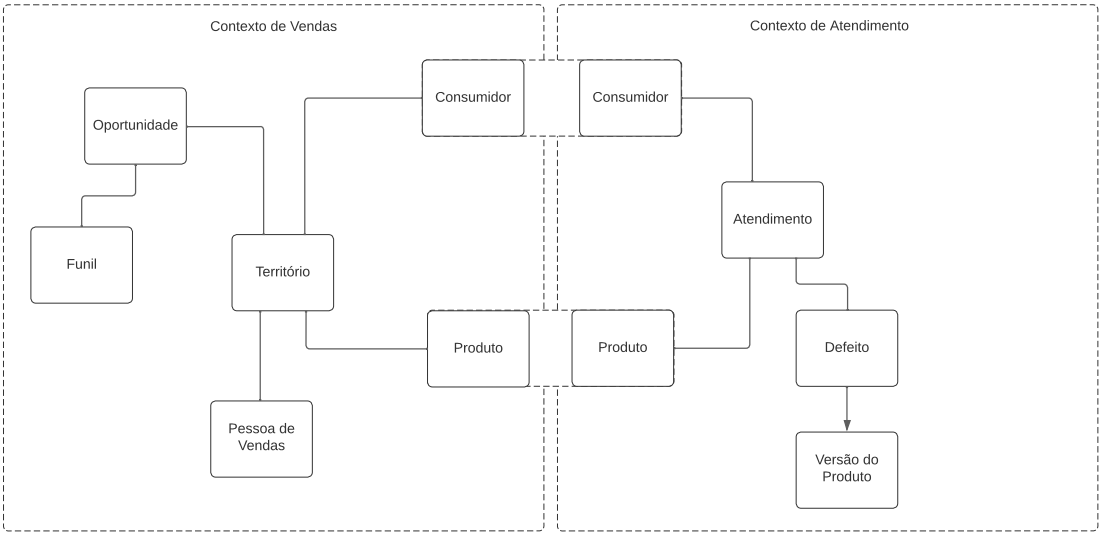
A terceira camada conhecida como View Model, é responsável por intermediar as informações entre a View e o Model. Ele também é responsável por manipular a lógica e as regras de visualização correspondentes à View, mesmo que este não possua conhecimento sobre os elementos presentes na tela. Sendo assim, assim como visto no diagrama demonstrado na Figura 4, o View Model é responsável por repassar as informações recebidas da View para o Model, assim como receber as informações do Model, deixá-las de fácil manipulação para a View e assim repassá-las.

2.3.4 *Domain Driven Design*

O *Domain Driven Design* (DDD), ou design orientado a domínio, é a arquitetura utilizada para possibilitar uma escalabilidade na complexidade do código e do software, pois seguindo este conceito, o projeto começa a ser divido por contextos e não mais

apenas por responsabilidades (EVANS, 2003). A exemplo disso, tem-se a Figura 5, a qual representa dois contextos, sendo eles, vendas e suporte.

Figura 5 – Contexto Delimitado



Fonte: (FOWLER, 2014).

A partir dessa figura, nota-se que para um aplicativo ter uma arquitetura escalável, este precisa possuir contextos limitados. Ou seja, cada contexto deve ter suas responsabilidades e caso uma destas se repita, esta deve ser recriada em seu novo contexto. Através disso, é possível prover uma melhor manutenção de código e tratar suas classes de maneira independente.

Através da imagem da Figura 5, percebe-se que o contexto de vendas e o contexto de atendimento possuem o consumidor e o produto, entretanto apesar dos dois contextos possuírem os mesmos elementos, estes estão duplicados, possibilitando que uma mudança no consumidor dentro do escopo de vendas não interfira no consumidor utilizado pelo escopo de suporte. Desse modo, o aplicativo segue este mesmo conceito, e por esse motivo, é dividido em três contextos descritos a seguir:

- Área deslogada, responsável pela autenticação e cadastro de novos usuários;
- Área logada pelo lado do usuário assistido, sendo esta responsável pelo envio da notificação para o usuário voluntário, criação da tela de chamada de vídeo, transmissão da câmera frontal do dispositivo e também reconhecimento de objeto;
- Área logada pelo lado do usuário voluntário, responsável pelo recebimento da notificação, leitura da mensagem recebida com o código da sala para entrar e acesso a transmissão feita pelo usuário assistido para poder auxiliá-lo conforme a necessidade.

Este modelo funciona como uma modelagem estratégica de como separar o seu domínio em partes menores responsáveis com contextos delimitados. Através dessa abstração, se torna mais fácil a compreensão de regra de negócio e a segregação de um código complexo em pacotes menores independentes. A partir disso, o aplicativo consegue progredir em cada uma das verticais de maneira independente, separando as responsabilidades conforme a atuação necessária.

3 METODOLOGIA

Neste capítulo serão apresentados os princípios utilizados para o desenvolvimento do aplicativo, a metodologia utilizada e a integração das tecnologias previamente descritas.

3.1 PRINCÍPIOS UTILIZADOS PARA O DESENVOLVIMENTO

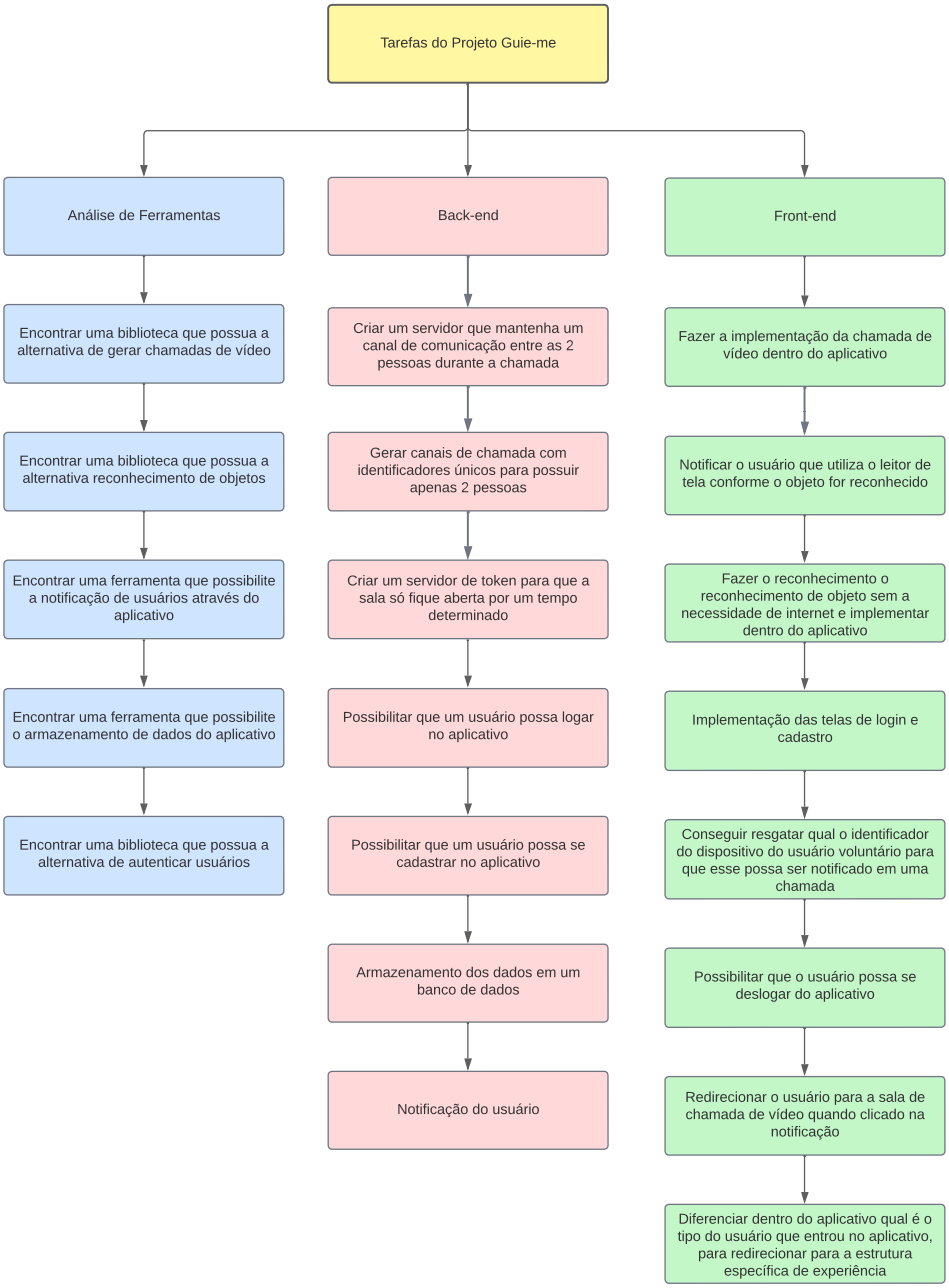
A proposta criada foi a de desenvolver um MVP (Minimum Viable Product) do aplicativo de tecnologia assistiva. O MVP é uma metodologia de aplicação do mínimo esforço necessário para que se obtenha um produto testável e comercializável. Devido a isso, este trabalho teve como objetivo abranger os seguintes requisitos:

- O aplicativo deve possibilitar o cadastro de novos usuários;
- O aplicativo deve possibilitar a autenticação do usuário;
- O aplicativo deve diferenciar o usuário que precisa de assistência visual do usuário voluntário, controlando o redirecionamento de fluxo ao se autenticar;
- No fluxo do usuário assistido, o usuário deve conseguir enviar uma notificação de chamada para um usuário voluntário e entrar na sala de vídeo chamada, de maneira simples e intuitiva;
- No fluxo do usuário assistido, caso deseje, este pode utilizar do reconhecimento de objeto para identificar objetos próximos a ele;
- No fluxo do usuário voluntário, o usuário deve conseguir receber uma notificação de chamada de vídeo e, através do clique na notificação recebida, entrar na sala de vídeo chamada;
- Ambos os usuários devem conseguir se deslogar do aplicativo também de maneira simples e intuitiva.

3.2 METODOLOGIA UTILIZADA

A metodologia utilizada para a criação do aplicativo foi o modelo Kanban para desenvolvimento de software. A partir do diagrama base demonstrado na Figura 2, foram divididas as tarefas conforme a Figura 6, com suas respectivas cores e ordem de desenvolvimento. Em vista disso, a primeira coluna que corresponde à parte de pesquisa, com suas demandas com o marcador azul, deve ser a primeira a ser executada. A segunda coluna que corresponde às tarefas e desenvolvimentos do *back-end*, na cor vermelha, é a segunda parte a ser realizada. Por fim, a terceira coluna de execução, na cor verde, corresponde ao desenvolvimento do aplicativo e do *front-end*. Assim que iniciadas, essas atividades são movidas respectivamente para a coluna ”Em progresso” e em seguida para ”Feito”.

Figura 6 – Divisão das tarefas relativas ao Kanban do aplicativo.



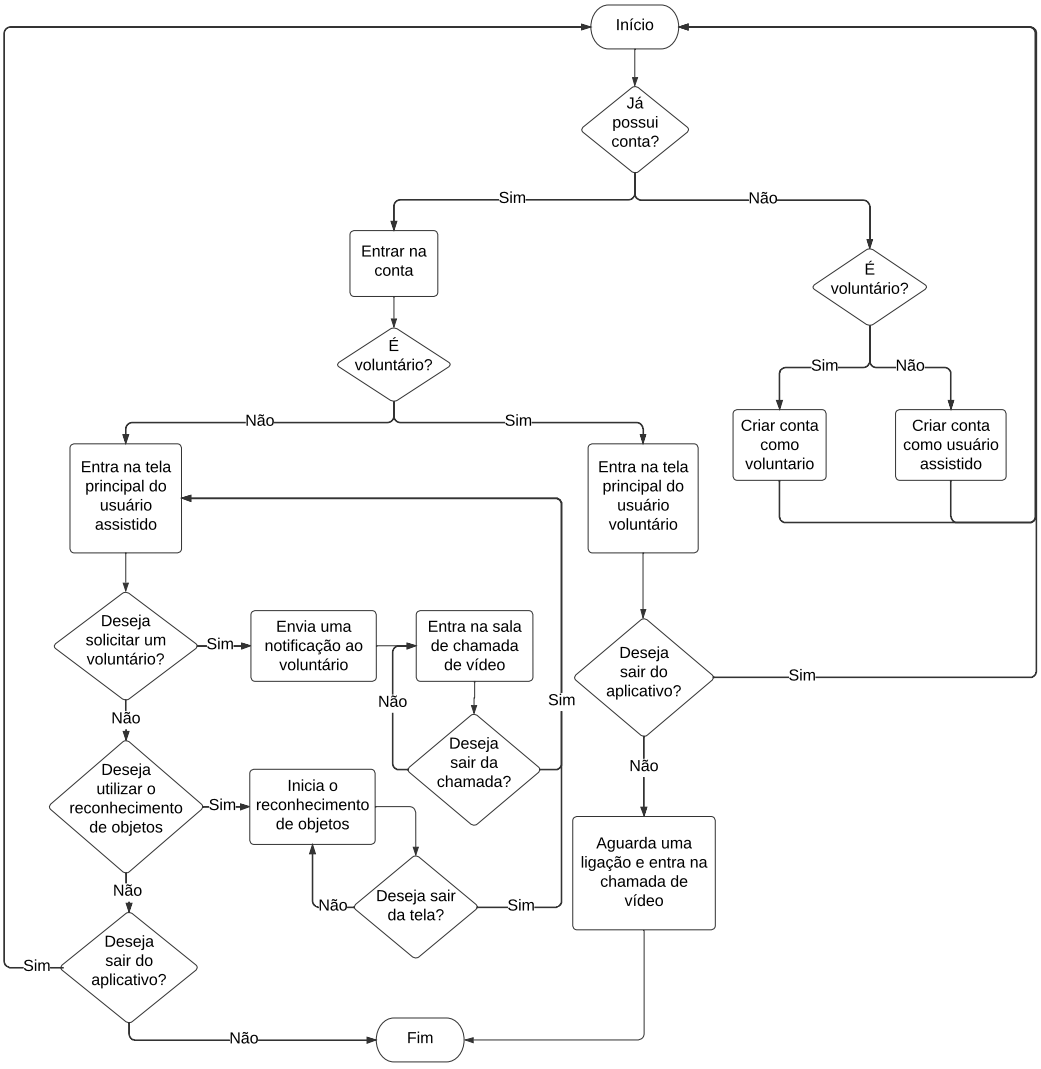
Fonte: O Autor (2022).

Esta segmentação foi realizada conforme as responsabilidades de cada demanda. As tarefas de pesquisa simbolizam à busca de informações, tecnologias e referências que poderiam ser utilizadas na aplicação. As demandas de *back-end* correspondem as atividades de execução não visível para o usuário e de intermediação entre a comunicação do aplicativo com os servidores. As tarefas de *front-end* impactam tanto na experiência do aplicativo quanto na implementação das chamadas dos serviços integrados pelo *back-end*.

3.3 DESENVOLVIMENTO DO APLICATIVO

Um dos desafios do trabalho desenvolvido foi a estruturação e integração das tecnologias apresentadas dentro do aplicativo, para que elas atuassem de maneira conjunta. Por este motivo, foi criada uma demonstração esquemática do seu funcionamento, no formato de fluxograma, demonstrado na Figura 7.

Figura 7 – Fluxograma do aplicativo Guie-me.



Fonte: O Autor (2022).

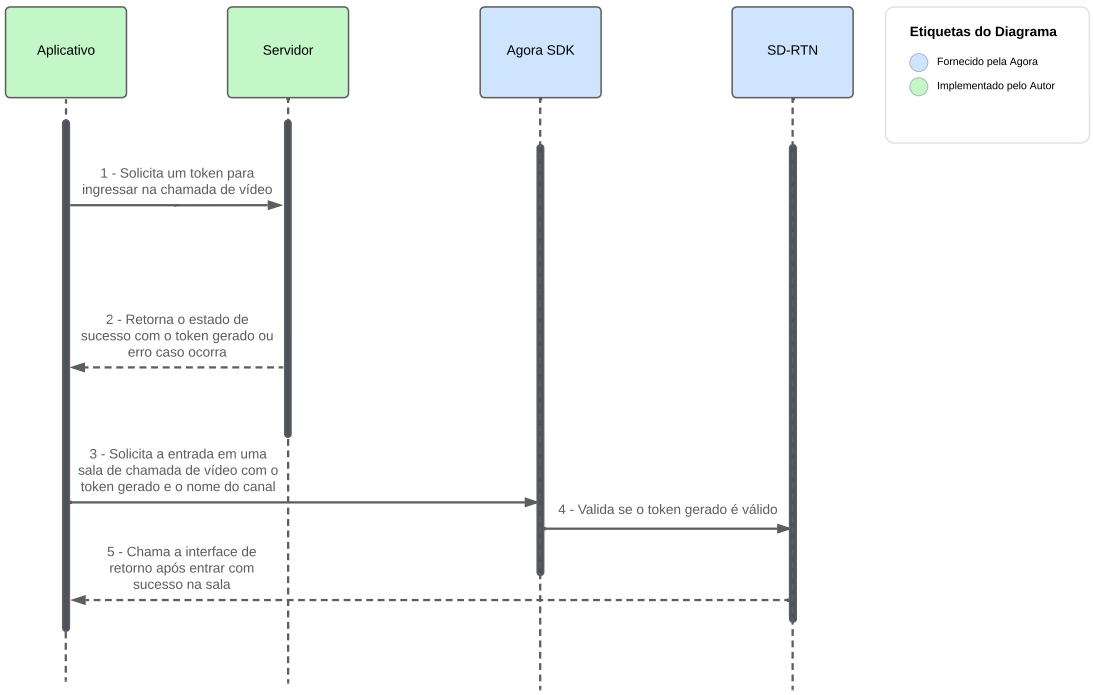
A partir desse fluxograma, é possível identificar como as diferentes tecnologias se comunicam e, junto a isso, o funcionamento do aplicativo para o usuário. O detalhamento dos desafios enfrentados para integrar cada tecnologia ao uso da solução desenvolvida, serão detalhados nas próximas secções.

3.3.1 Realização da chamada de vídeo

A principal funcionalidade do aplicativo é a comunicação entre dois usuários através de uma chamada de vídeo. Devido a isso, dentre os produtos da Agora, foi utilizado o *Agora Video SDK*.

Através desta biblioteca, obtém-se uma interface, a qual proporciona a implementação necessária para que seja possível realizar a integração ao $SDRTN$. O $SDRTN$ funciona como um servidor, o qual é responsável por receber as informações que são necessárias para realizar a chamada de vídeo em tempo real. O funcionamento completo dessa aplicação pode ser visto na Figura 8.

Figura 8 – Comunicação entre o aplicativo, servidor desenvolvido e o servidor da Agora.



Fonte: (AGORA, 2022).

Na imagem acima, é possível observar que o aplicativo realiza uma solicitação de *token* para um servidor desenvolvido pelo autor. Esse *token* autoriza ao cliente requisitante acessar o recurso desejado.

Assim que o servidor retorna o *token* de acesso, é feita uma solicitação de entrada em uma sala de transmissão de tempo real, denominada RTC (Real Time Connection). A RTC é responsável por manter a conexão entre os usuários, permitindo que estes se comuniquem com baixa latência e atraso na comunicação.

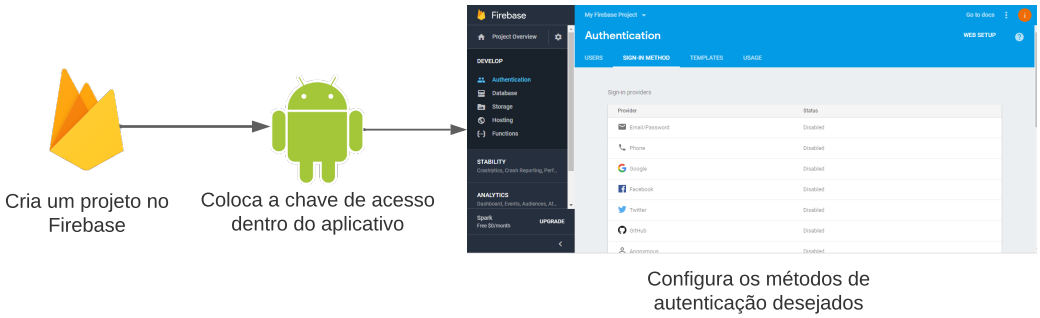
Assim que feita essa requisição, internamente o servidor da Agora realiza a validação do *token* fornecido. Caso este seja válido, será retornado um estado de sucesso e o usuário

poderá entrar na sala de chamada de vídeo, caso contrário, o usuário receberá uma mensagem de erro.

3.3.2 Autenticação dos usuários

Para a parte de autenticação, foi utilizado a ferramenta Firebase Authentication, pois esta fornece uma biblioteca de implementação simplificada e não possui custos caso tenham menos de 3 mil usuários ativos por dia. Tendo em vista esses aspectos, inicialmente foi criado um projeto dentro do Firebase, em seguida, foi inserida a chave gerada dentro do projeto Android e por fim, foi escolhido ativar apenas o método de autenticação via e-mail e senha, os passos seguidos podem ser vistos na Figura 9.

Figura 9 – Processo realizado para a habilitação do Firebase Authentication.



Fonte: O Autor (2022).

Uma vez implementado, é possível utilizar de seus recursos de cadastro, assim como conectar ou desconectar a sessão de uma conta. Junto a isso, foram tratados erros retornados pelo servidor, como e-mail já cadastrado, número mínimo de 6 caracteres necessários para a senha e e-mail de formato inválido.

Como esse processo funciona através de uma requisição para o servidor do Firebase. Foram desenvolvidas regras de casos de uso para a autenticação, descritos na sequência, com o intuito de diminuir requisições que falhariam em algumas das regras básicas de requisitos. A exemplo dos casos de uso de autenticação, é feito a validação para caso o usuário tenha inserido uma senha com no mínimo 6 caracteres e que dentre eles existam letras e números. É validado também se o e-mail possui uma formatação válida, possuindo caracteres como "." e "@". Por fim, também é verificado se a senha preenchida no campo de confirmação de senha é igual a senha preenchida no campo de criação de senha.

Assim que criada a conta do usuário, automaticamente é gerado um UID (Unique Identifier), o UID funciona como um identificador único para cada usuário, que visa anonimizar os dados do cliente, assim como gerar uma chave primária não repetida que garanta a segurança e confiabilidade dos dados. Através desse identificador é possível controlar e armazenar as preferências e dados atrelados a este usuário, em um banco de

dados, sem que sejam comprometidos seus dados sensíveis e respeitando a lei geral de proteção aos dados.

Juntamente a isso, foi inserido o sistema de *login* que busca verificar através dos dados inseridos nos campos de e-mail e senha, se o usuário pode de fato entrar no aplicativo. A verificação do *login* também acontece através do serviço do Firebase e assim que o usuário entra na aplicação, caso este não clique no botão de se desconectar, ele permanecerá logado devido ao *login* persistente inserido.

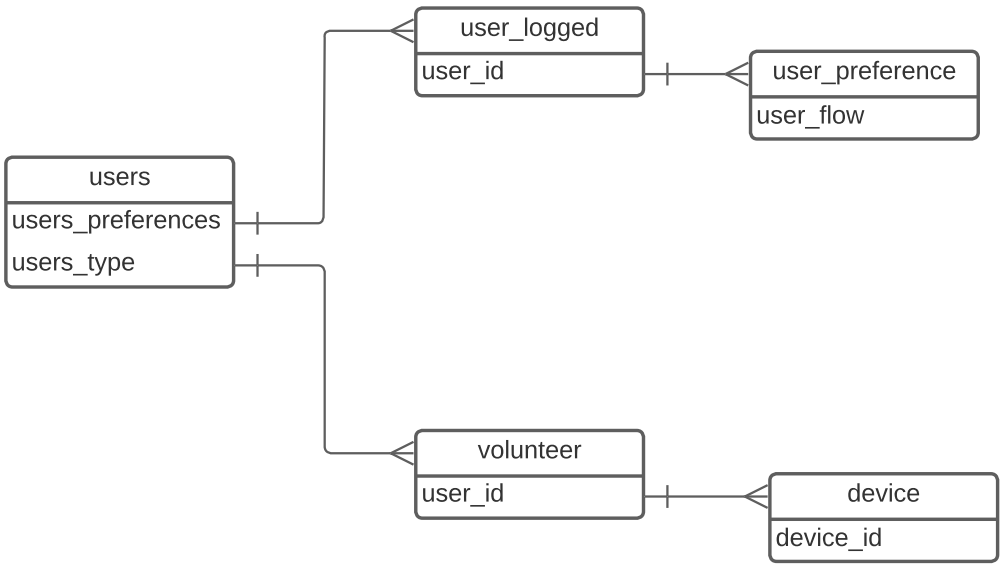
3.3.3 Armazenamento de dados dos usuários

Para o armazenamento de dados dos usuários, foi utilizado novamente o Firebase, com seu recurso Real Time Database. Dentro do aplicativo, ele é utilizado para duas principais funções.

- Diferenciar qual fluxo que o usuário deverá ter após efetuar o login com sucesso, ou seja, ele deve ter armazenado qual o tipo do usuário que entrou no aplicativo, se ele é um usuário assistido ou voluntário;
- Segmentar os usuários voluntários em uma lista, para poder resgatar e sortear um voluntário. O usuário sorteado será notificado sobre a solicitação de ajuda, através do envio de uma notificação para o seu dispositivo.

Para que essas duas funções sejam possíveis, faz-se necessário armazenar as preferências dos usuários ao se cadastrar em um banco de dados. Por estes motivos, é possível verificar a estrutura do banco de dados criado na Figura 10.

Figura 10 – Modelo de dados criado para o armazenamento do banco de dados.



Fonte: O Autor (2022).

Através dessa estrutura, torna-se menos complexo realizar a busca dentro do banco de dados, visto que este é um banco de dados não relacional. Por este motivo, ao usuário logar, ele verificará dentro do campo "*user_preferences*" qual a preferência do usuário correspondente ao UID logado, sendo ela "*assisted_user*" ou "*volunteer*", com o intuito de controlar e redirecionar para a respectiva tela inicial.

Já para o caso do usuário assistido que deseja pedir ajuda, ele verificará quais usuários estão dentro do campo "*user_type*" e dentro do valor "*volunteer*", para assim sortear um usuário aleatório. O campo de detalhes do UID, contendo o "*device_id*" corresponde ao identificador do dispositivo no qual foi feito o último login do usuário, este dado é armazenado para que posteriormente seja possível realizar o *push notification* direcionado para este dispositivo.

3.3.4 Sala única e notificação do usuário

Para que os usuários possam interagir e se comunicar, é necessário que eles entrem em uma mesma sala de conferência. Ressalta-se que o objetivo dessa sala é de que exista uma comunicação conhecida como *peer-to-peer*, ou seja de pessoa para pessoa, limitando a sala a apenas dois usuários, sendo eles um usuário assistido e um usuário voluntário.

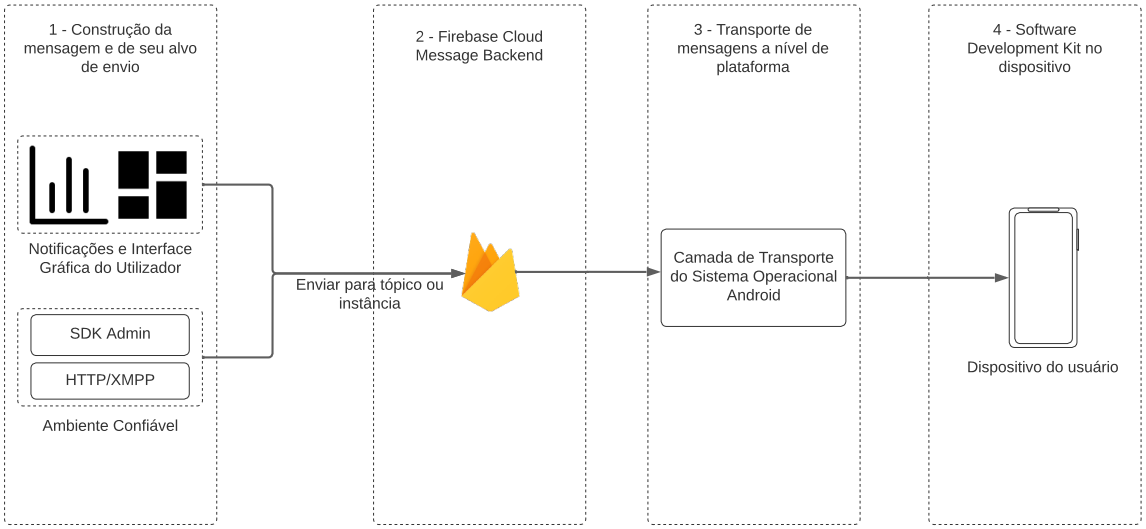
Por este motivo, para que não tenham mais do que duas pessoas na mesma sala concomitantemente, deve-se gerar uma chave de acesso única, que será compartilhada através de um meio de comunicação entre os dois usuários.

Tendo em vista esses aspectos, para a solução da sala com identificador único, foi utilizado o UID do usuário assistido, gerado pelo Firebase. Através desse UID, é separado do aplicativo a responsabilidade de gerar salas randômicas e, junto a isso, une ao identificador único já criado pelo Firebase a funcionalidade de gerar uma sala pessoal, pois apenas esse usuário possuirá este UID. Devido a essa chave ser pública e não ser utilizada para nenhuma operação dentro do banco de dados, não existem riscos ao compartilhá-la.

Para a parte do meio de transmissão, foi utilizado o FCM, cujo funcionamento pode ser visto através na Figura 11. Este, assim que ativado, permite que sejam encaminhados *push notifications* aos usuários desejados. O Firebase disponibiliza uma $API\ REST$ auxiliar, a qual permite, através do uso do *device_id* mencionado anteriormente, direcionar o *push notification* do usuário assistido diretamente para o usuário voluntário.

Dentro da $API\ REST$ fornecida para a notificação, é possível parametrizar seu conteúdo, dentre eles título da mensagem, descrição, ícone, cor e conteúdo. Para que o usuário voluntário receba a sala criada pelo usuário assistido, a chave de acesso da sala é passado via conteúdo da notificação. Assim que o voluntário clica na notificação, o aplicativo é aberto e com isso o conteúdo da notificação é extraído e o usuário é direcionado para a sala de vídeo conferência, na qual o usuário assistido aguarda a sua ajuda.

Figura 11 – Funcionamento do Firebase Cloud Messaging.



Fonte: (FIREBASE, 2022).

3.3.5 Criação de servidor

Para a integração de todos os serviços descritos previamente, fez-se necessário a criação de um servidor que suprisse as demandas necessárias. Por este motivo, foi desenvolvido um servidor em Javascript, utilizando o *framework* Next.js. Dentro do servidor, ficam as responsabilidades de gerar um novo *token* de acesso válido, para que os usuários possam entrar na sala da chamada de vídeo, e a responsabilidade de enviar a notificação para o voluntário quando o usuário assistido solicita.

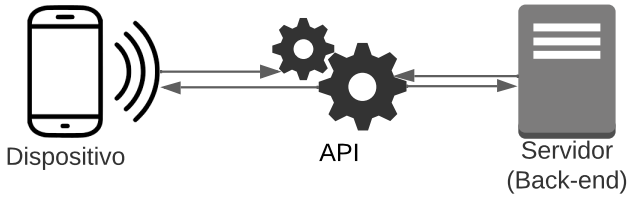
O intuito do servidor é intermediar as requisições feitas pelo aplicativo e separar a responsabilidade entre *front-end* e *back-end*. Essa separação é importante, pois as regras do processo de geração do *token* de acesso ou do *push notification* não devem estar centralizadas no aplicativo.

A divisão das atribuições ocorre em razão das funcionalidades não se limitarem a apenas uma plataforma. Caso em algum momento seja necessário realizar alterações na mensagem de notificação ou na geração do *token* de acesso, não será necessário lançar uma nova versão do aplicativo. Além disso, caso esse projeto possua uma versão para dispositivos iOS no futuro, essa responsabilidade também fica descentralizada. O funcionamento dessa estrutura pode ser visto na Figura 12.

Através do servidor, consegue-se controlar e sintetizar para o aplicativo apenas as informações necessárias e úteis, como por exemplo a resposta do servidor com o novo *token* gerado ou se o envio da notificação foi bem sucedido. Toda a montagem de estrutura do *token* assim como a definição dos parâmetros enviados na notificação ficam com a centralizados na *API REST* desenvolvida. O fluxograma que demonstra o funcionamento

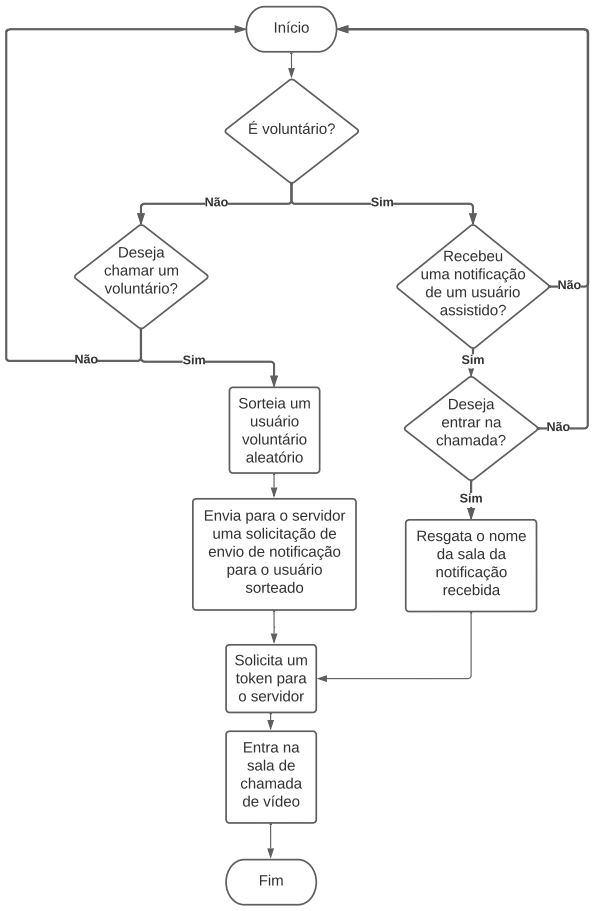
do servidor através do aplicativo pode ser visto na Figura 13, no qual o início se dá com a intenção de realizar a chamada vídeo e seu fluxo demonstra a interação do aplicativo com o servidor.

Figura 12 – Estrutura de comunicação entre o dispositivo, *API* e back-end.



Fonte: O Autor (2022).

Figura 13 – Fluxograma do funcionamento do aplicativo com o servidor desenvolvido.



Fonte: O Autor (2022).

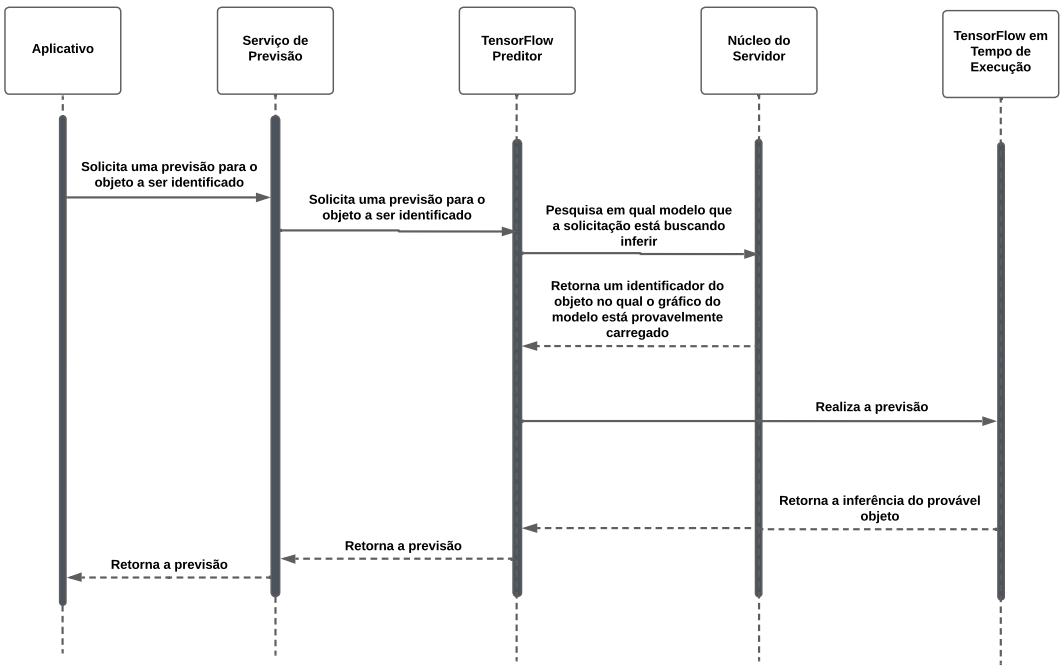
Destaca-se que, além do servidor ser uma importante peça para a segregação de

responsabilidades entre *front-end* e *back-end*, para a Agora o servidor é um intermediador necessário. Isso ocorre pois os *tokens* possuem um tempo de expiração, que para o aplicativo é de 1 hora, por este motivo e para que seja possível gerar múltiplas salas, cada uma com seu identificador único, é necessário gerar esses *tokens* dinamicamente.

3.3.6 Reconhecimento de objetos

Para o reconhecimento de objetos, foi utilizado o TensorFlow. Esta plataforma de código aberto para aprendizado de máquina possui como funcionamento o diagrama de sequência visto na Figura 14. Para sua atuação no modo embarcado, é necessário treinar, ou utilizar uma rede neural treinada, no formato *.tflite*. Esse formato conta com uma coleção de ferramentas e otimizações dos modelos para que seja possível roda-los em dispositivos móveis.

Figura 14 – Diagrama de sequência do funcionamento do TensorFlow.



Fonte: (TENSORFLOW, 2021).

Uma vez provido um modelo treinado, a ferramenta consegue interpretar seus dados salvos e com isso realizar predições sobre o possível objeto identificado. Assim que realizada a predição é retornado ao cliente qual foi o objeto identificado e qual é sua porcentagem de confiabilidade.

Para utilizar o TensorFlow foi necessário inserir sua biblioteca dentro do aplicativo, juntamente com um modelo base. O exemplar escolhido foi retirado da base de modelos

disponibilizados pela plataforma e é denominado *mobilenetv1.tflite*. Após sua implementação, a biblioteca é iniciada assim que o usuário abre a tela do aplicativo correspondente ao reconhecimento de objetos. A partir deste momento, a ferramenta segue seu diagrama de sequência, demonstrado na Figura 14, de maneira recursiva enquanto a tela estiver em execução.

A cada previsão realizada pelo TensorFlow, é preciso indicar para o usuário qual foi o objeto identificado. Deste modo, após obter este resultado, é criado um desenho retangular ao redor deste objeto, para sinalizar qual é o objeto em foco durante a predição. Junto a isso, o objeto previsto é informado ao usuário assistido pelo Talkback, o qual realiza a conversão de texto para fala.

4 RESULTADOS OBTIDOS

Nesse capítulo será apresentado e discutido o MVP desenvolvido, com todas as telas do aplicativo, divididos entre autenticação, usuário assistido e usuário voluntário.

4.1 AUTENTICAÇÃO

A tela inicial do aplicativo, exibida ao abri-lo pela primeira vez, pode ser vista na Figura 15. Esta tela proporciona de uma maneira simples uma apresentação sobre o objetivo da aplicação, assim como as possíveis alternativas do usuário, caso já possua uma conta ou caso deseje realizar o cadastro.

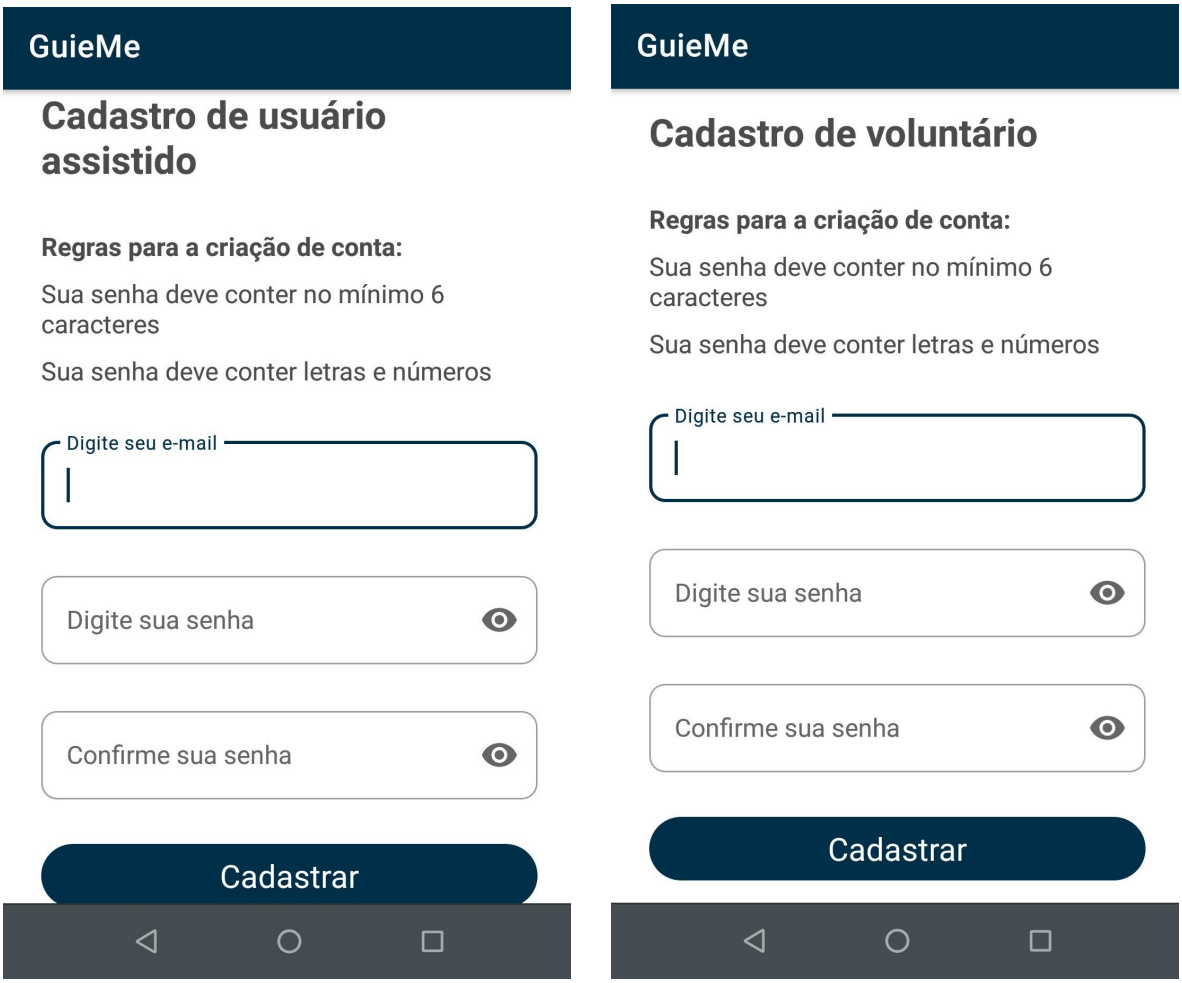
Caso o usuário deseje se cadastrar, ele deve escolher entre se cadastrar como voluntário, usuário que receberá as notificações de chamadas e que auxiliará o usuário com deficiência visual, ou se cadastrar como usuário assistido, o qual poderá solicitar a ajuda de um voluntário ou utilizar do modelo de reconhecimento de objetos.

Figura 15 – Tela inicial do aplicativo.

Fonte: O Autor (2022).

Caso o usuário escolha a opção de se cadastrar, a tela apresentada é idêntica tanto para o usuário assistido, quanto para o usuário voluntário, modificando apenas o texto mostrado na tela, como demonstrado na Figura 16. Apesar da tela ser idêntica para os dois cenários, o aplicativo recebe da primeira tela apresentada um parâmetro que identifica qual foi o tipo de cadastro selecionado. Isso ocorre para que a partir do momento que o usuário realize o cadastro com sucesso, seja armazenado no Real Time Database a preferência do usuário escolhida no cadastro.

Figura 16 – Tela de cadastro do aplicativo.



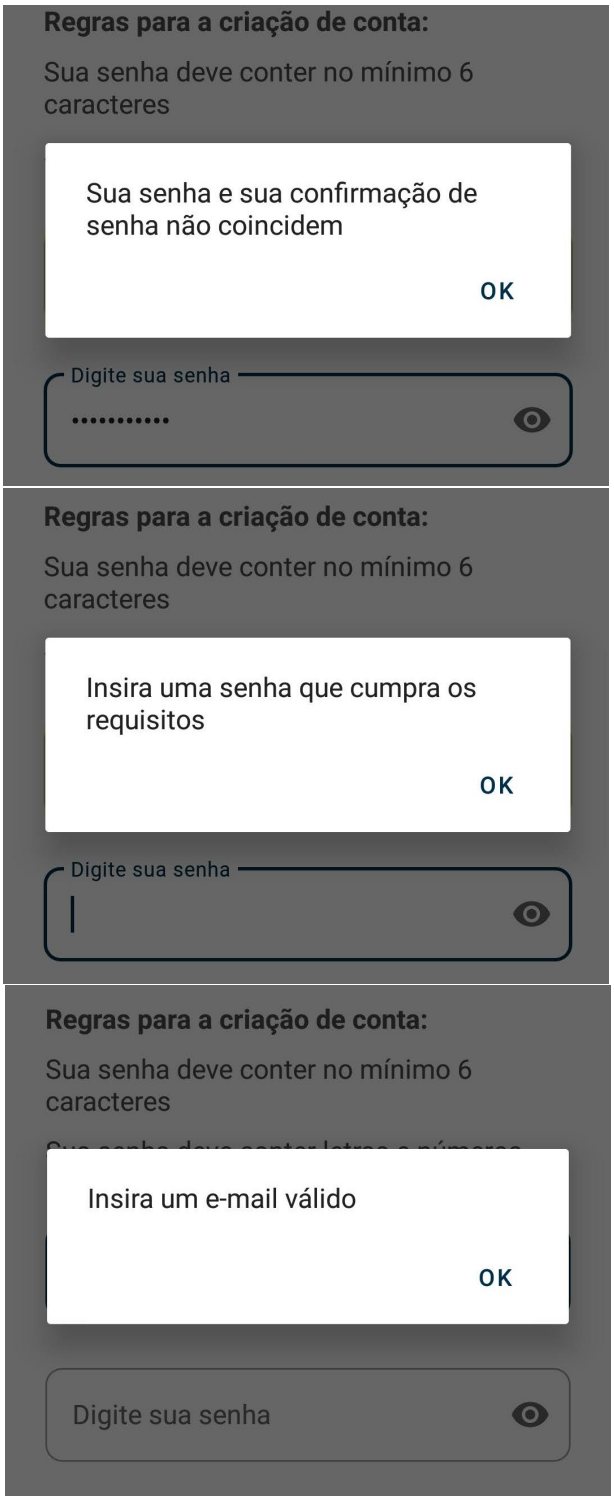
Fonte: O Autor (2022).

Entretanto, é possível que o usuário, durante o cadastro, não tenha preenchido os requisitos necessários para a criação de conta, ou até tente cadastrar um e-mail já registrado. Para esses cenários, existem tratamentos de erros e os casos de uso de autenticação, que informam ao usuário qual foi o erro ocorrido e o que ele deve fazer para corrigi-lo. Através da Figura 17 é possível observar quais são as possíveis mensagens de erros.

Assim que o cliente cumprir todos os requisitos citados, ele realizará com sucesso o cadastro no aplicativo. Em seguida, ele será novamente direcionado para a tela inicial do aplicativo. Após esse processo, como este já está cadastrado, poderá escolher a opção

"Já possuo conta" para poder realizar seu *login*.

Figura 17 – Telas com mensagem de erro.



Fonte: O Autor (2022).

A tela de de *login* do aplicativo pode ser vista na Figura 18. Nesta tela, é preciso realizar o preenchimento do e-mail e senha previamente cadastrados. Devido a isso, nela também existe o mesmo fluxo de validação das informações se a conta e a senha são válidos.

Caso as informações não estejam corretas, serão exibidas mensagens de erro seguindo o mesmo modelo demonstrado acima.

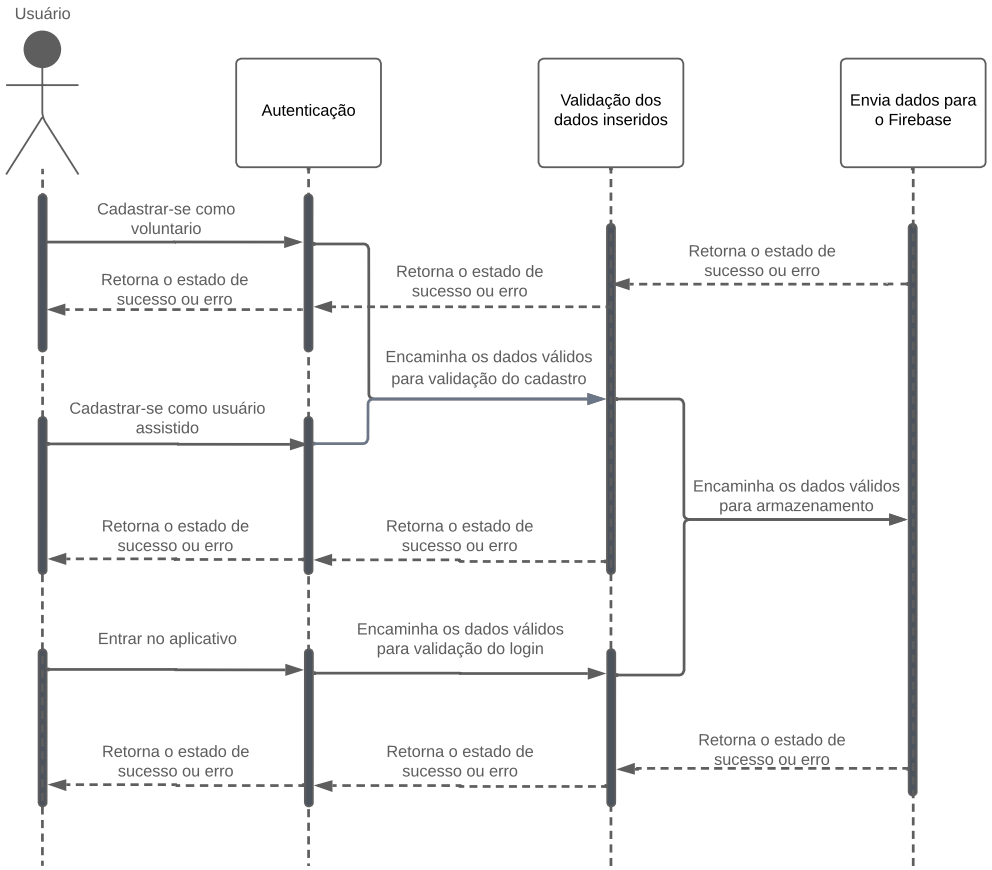
Juntamente com a validação de e-mail e senha, nesta tela também ocorre a validação do fluxo que será visto pelo usuário que está entrando no aplicativo. Isso ocorre pois assim que autenticado, o cliente pode ser direcionado ou para o fluxo de usuário assistido ou voluntário, telas que serão detalhadas nas próximas seções.

Figura 18 – Tela de Login.

Fonte: O Autor (2022).

Em virtude de existirem dois fluxos, assim que verificada a autenticação do usuário, é buscado no Real Time Database qual é a preferência do usuário escolhida no momento do cadastro, para que com esse dado ele seja direcionado para o seu fluxo adequado. O diagrama de interação que representa o processo de autenticação pode ser visto na Figura 19.

Figura 19 – Diagrama de interação da autenticação.



Fonte: O Autor (2022).

Através deste diagrama, é possível identificar que após o clique para se cadastrar ou entrar no aplicativo, culmina em diversas ações de validações a partir dele. Tendo em vista essa quantidade de validações, percebe-se a importância de não enviar requisições para o servidor enquanto os dados preenchidos não estiverem adequados com os requisitos mínimos para criação ou autenticação de conta. No cenário em que as requisições fossem feitas diretamente para o servidor, o procedimento de autenticação seria mais demorado e custoso, visto que sempre dependeria do retorno da chamada.

Outro ponto que o diagrama evidencia, é a importância de um tratamento de erros de fácil entendimento para o usuário. Dado que existem várias validações durante o processo autenticação, é de suma importância devolver para o usuário uma mensagem adequada para que ele compreenda qual ação deve realizar para se autenticar no aplicativo.

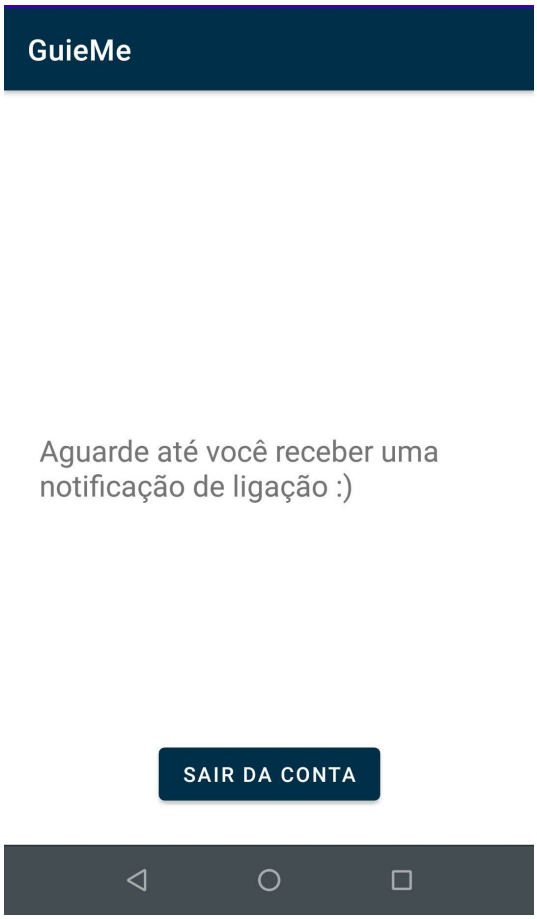
4.2 ÁREA AUTENTICADA

A área autenticada, corresponde a toda a área na qual só é possível ter acesso após realizar o *login*. Assim que feito o cadastro e a autenticação o usuário será direcionado para um dos fluxos detalhados a seguir.

4.2.1 Usuário voluntário

Para o cenário de que o usuário autenticado seja um usuário voluntário, será apresentada a tela demonstrada na Figura 20.

Figura 20 – Tela principal do usuário voluntário.

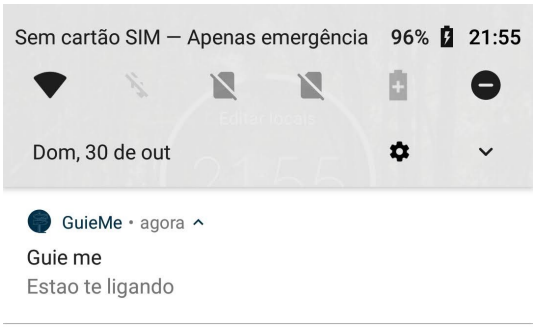


Fonte: O Autor (2022).

Nesse cenário, a interface e as ações do cliente são bastante limitadas, apenas fornecendo uma mensagem de que ele poderá receber uma notificação de ligação e fornecendo um botão para se desconectar do aplicativo. Como o aplicativo possui *login* persistente, caso o usuário não clique no botão de sair de conta, ao abrir a aplicação ele será direcionado diretamente para essa tela.

Nesse cenário, assim que autenticado, é registrado no Real Time Database o identificador único do dispositivo, fornecendo com isso um canal para que ele possa receber uma notificação a qualquer instante. No caso do aplicativo estar fechado ou em plano de fundo, se ele receber uma notificação ela aparecerá conforme a Figura 21, caso contrário, se o aplicativo estiver aberto na tela principal, ele será diretamente direcionado para a vídeo chamada sem ser notificado previamente.

Figura 21 – Modelo de notificação recebida pelo usuário voluntário.



Fonte: O Autor (2022).

4.2.2 Usuário assistido

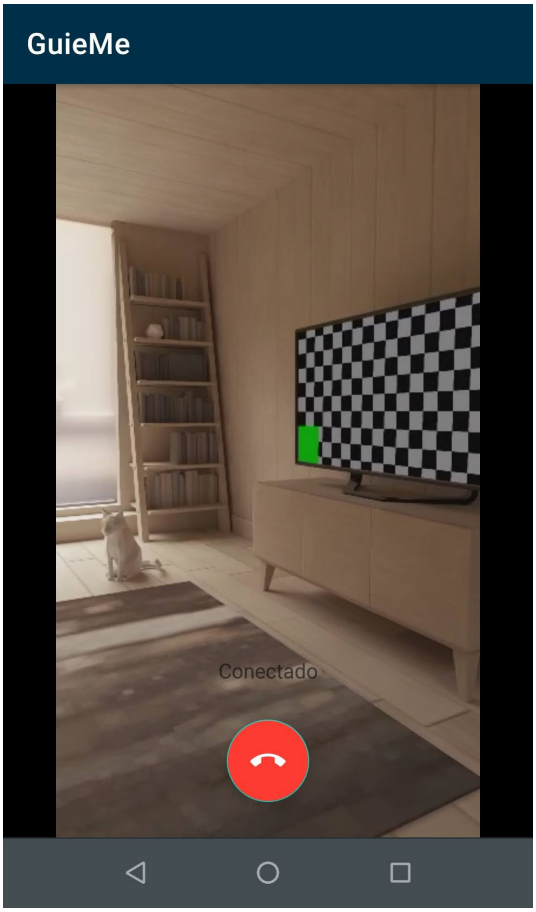
Assim que um usuário assistido é autenticado, é apresentada a tela de início demonstrada na Figura 22.

Figura 22 – Tela principal do usuário assistido.

Fonte: O Autor (2022).

Nesta tela, o usuário consegue escolher quais das funcionalidades melhor irá suprir suas necessidades para o momento, suas opções são ligar para voluntário, reconhecimento de objeto ou sair da conta. Assim que selecionada a opção de chamada de voluntário, o usuário é direcionado para uma sala de chamada de vídeo e aguardará até que um voluntário clique na notificação de chamada. A tela da chamada de vídeo pode ser vista na Figura 23.

Figura 23 – Tela com a chamada de vídeo.

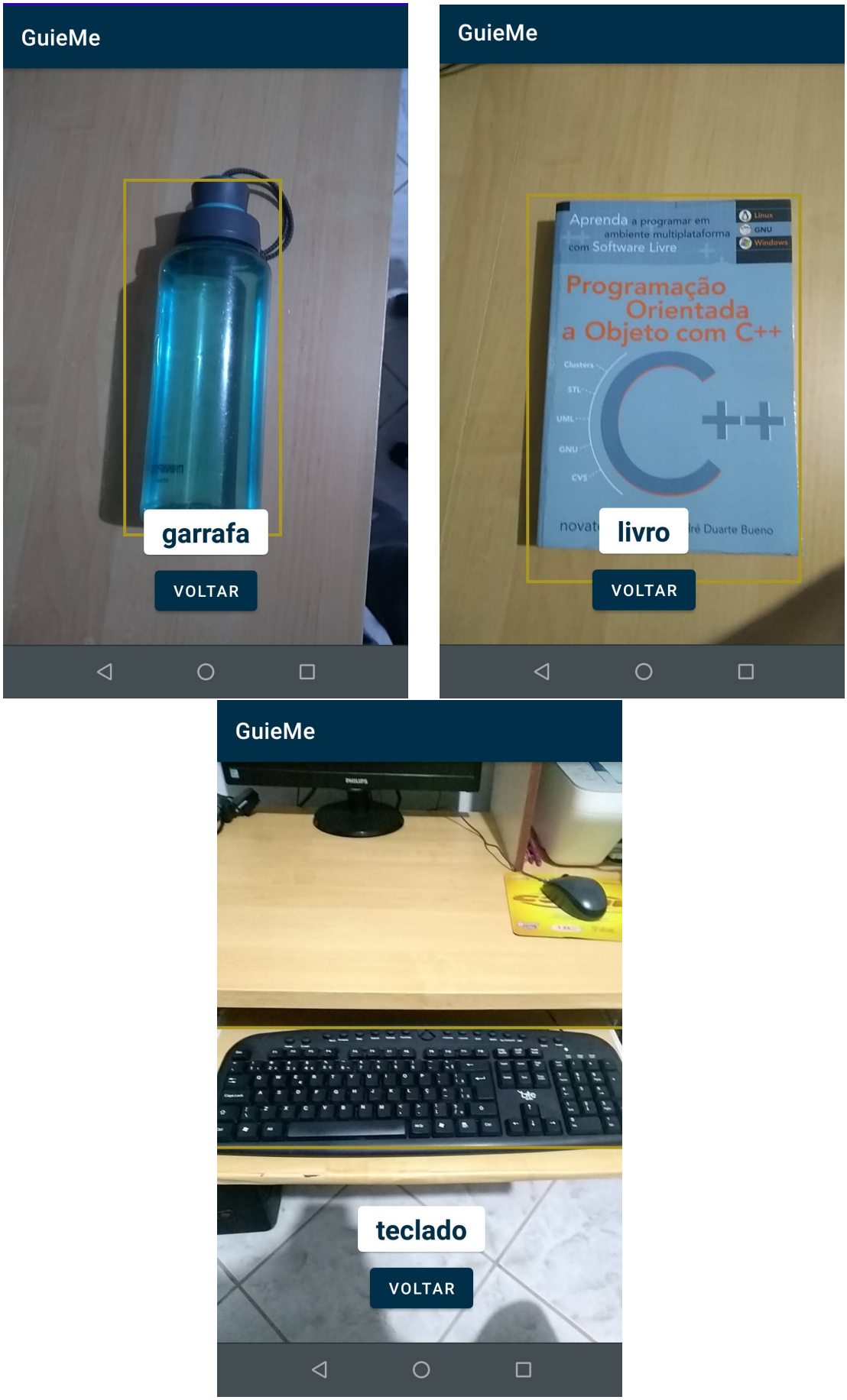


Fonte: O Autor (2022).

Dentro dessa tela, o usuário é notificado através de um texto logo acima do botão quando ele está conectado a um voluntário. Assim que estabelecida a conexão, ele poderá movimentar o seu dispositivo e solicitar a ajuda desejada ao voluntário. Logo que todas as necessidades que levaram à realização da chamada sejam sanadas, ambos os usuários podem realizar a finalização da chamada clicando no botão vermelho da tela. Após a finalização da chamada, ambos os usuário retornam para a tela anterior.

Neste caso, o usuário pode selecionar neste momento a opção de reconhecimento de objeto. Assim que selecionada a tela de reconhecimento de objeto, o usuário é direcionado para a tela demonstrada na Figura 24.

Figura 24 – Tela com reconhecimento de objeto.



Fonte: O Autor (2022).

Nesta tela o modelo pré-treinado MobileNet é colocado em execução, junto a isso enquanto a câmera está em execução, o sistema envia para o modelo treinado o que está sendo detectado através da câmera do usuário. Assim que a imagem é avaliada pelo modelo, é retornado para o sistema o resultado dessa predição, com isso é retornado o nome do objeto reconhecido e seu nível de confiabilidade.

Com o objetivo de realizar tratativas que transformem os dados para uma maneira mais amigável para o usuário, é reconhecido apenas um objeto de cada vez, sendo esse o objeto que está mais em destaque na câmera. Além disso, devido ao modelo ser pré-treinado, seus objetos detectados estão todos em inglês, devido a esse ponto foi necessário criar um dicionário local, no qual é enviado o nome do objeto identificado em inglês e seu nome é traduzido para o português. Caso o objeto não tenha uma tradução ele retorna como objeto não identificado. Ressalta-se que a cada segundo o Talkback lê em voz alta o objeto reconhecido, independentemente dele ter variado ou não.

Por fim, caso o usuário deseje retornar para a tela anterior ele pode clicar no botão de voltar na tela. Com isso, a última ação possível para esse usuário é a de deslogar do aplicativo. Da mesma maneira que o usuário voluntário, caso este não deslogue da aplicação, ao abri-lá ele já será redirecionado para a tela de início de seu fluxo.

4.3 ANÁLISE DOS RESULTADOS

Tendo em vista a integração das tecnologias realizada, assim como o resultado do aplicativo obtido, conclui-se que a solução proposta atingiu os requisitos desejados, assim como a finalidade inicial de desenvolver um MVP. Dentre seus pontos, a parte mais complexa de se realizar foi a integração com a plataforma da Agora, utilizada para realizar videochamadas, visto que para seu uso, foi necessário a implementação de um servidor auxiliar para utilizá-la de maneira escalável. Além disso, outra parte de elevada complexidade foi a implementação das notificações direcionadas ao dispositivo. Essa complexidade ocorre, pois para sua implementação foi necessário tanto o consumo da *API REST* do Firebase, como também cuidados em relação a informações sensíveis do usuário a exemplo do identificador do dispositivo.

Dentre os pontos fortes, nota-se que a aplicação é eficaz na parte de autenticação, visto que ela possibilita o cadastro de múltiplas pessoas de uma maneira intuitiva para o usuário. Além disso, ela também propõe que o usuário assistido possa escolher entre as funcionalidades desejadas para o momento, sendo elas uma interação *online* através da vídeo chamada ou apenas o método *offline* utilizando o reconhecimento de objetos.

Outro ponto positivo da aplicação, além das diversas tecnologias utilizadas, é em relação ao aplicativo ser objetivo em relação aos seus recursos e acessível, possibilitando que qualquer usuário com deficiência visual, após conectado, possa utilizar dos recursos com apenas um clique.

Entretanto, ao analisar pontos fracos que necessitariam de evolução, tem-se que o

modelo utilizado para o reconhecimento de objetos é muito simples para o cotidiano das pessoas, devido a isso ele possui uma alta quantidade de objetos cadastrados, contudo possui uma baixa variação deles, ou seja seriam necessários treinamentos de diversos modelos de mesa para que fosse possível reconhecer variados tipos de mesas.

Outro ponto de melhoria identificado, é em relação a distribuição de notificação para os usuários voluntários. Em função do banco de dados ser $NoSQL$ e ter a funcionalidade de um MVP, o usuário a ser notificado é resgatado do banco de dados por meio de um sorteio, sem passar por um filtro em relação a quantidade de notificações já recebidas. Por esta razão, não existe uma distribuição normal entre todos os usuários, acarretando no não recebimento de notificações enquanto o usuário não for sorteado. Além disso, resgatar esta tabela a cada envio de notificação também gera um problema para a escalabilidade nas ligações, pois conforme a tabela aumenta, torna-se inviável resgatá-la completamente para a realização do sorteio.

5 CONCLUSÃO

A motivação deste trabalho de conclusão de curso se deu devido à baixa quantidade de aplicativos acessíveis no mercado, visto o grande número de pessoas deficientes visuais. Entende-se que um dos principais objetivos da tecnologia é o de facilitar o cotidiano das pessoas, auxiliando em tarefas e otimizando o seu tempo. Devido a isso, tecnologias acessíveis e de uso simplificado são necessárias para melhorar a qualidade de vida dos seus usuários.

Durante o desenvolvimento, foi possível estudar sobre a acessibilidade em aplicativos nativos Android, assim como, modelos de arquitetura, tecnologias para realizar vídeo chamadas com a Agora e modelos de reconhecimento de objetos com o TensorFlow. Além disso, foi construído tanto *back-end*, quanto o *front-end*, o primeiro sendo desenvolvido em Next.js, uma tecnologia desenvolvida em JavaScript, e o segundo sendo desenvolvido em Kotlin. Alinhado com esses conhecimentos técnicos utilizados, foram aplicados diversos conhecimentos adquiridos durante o curso, como metodologias para o desenvolvimento ágil, visto em matérias como administração e conhecimentos em diversas áreas da programação, desde orientação a objetos, protocolos de telecomunicação e visão computacional.

Para a parte de autenticação, banco de dados e notificação foi utilizado o Firebase, o qual funcionou como uma tecnologia centralizadora das informações do usuário, através dele foi obtido o UID do usuário, assim como armazenado suas preferências. O servidor, auxiliou para a geração de *tokens* dinâmicos que possibilitasse a entrada nas vídeos chamadas e as notificações proporcionaram que através do UID do usuário assistido, fosse possível que o voluntário se conectasse na mesma sala de conferência.

Dessa maneira, constata-se que o objetivo da construção de um MVP acessível foi atingido com sucesso, tendo em vista esse aspecto, a próxima etapa de validação do projeto ocorrerá através de de validações com pessoas deficientes visuais, com o intuito de identificar possíveis melhorias e com isso aprimorar a solução proposta.

5.1 SUGESTÕES PARA TRABALHOS FUTUROS

Em relação a trabalhos futuros, uma melhoria no modelo de reconhecimento de objetos utilizados pode ser elencado como grande relevância. Devido a simplicidade do modelo pré-treinado utilizado, assim como devido ao modelo ter sido catalogado inicialmente em inglês, torna-se de importante a evolução do projeto com uma solução robusta em reconhecimento de diferentes objetos utilizados no cotidiano, em português e com alta assertividade.

Além deste ponto, outra melhoria a ser considerada é a criação de uma metodologia que realize a distribuição dos usuários voluntários notificados em uma curva equalitária de chamadas recebidas. Isso auxiliaria para que todos os usuários tivessem uma quantidade semelhante de chamados, não ocupando muito do seu tempo enquanto proporciona com

que todos ajudem. Junto a isso, o desenvolvimento de um método de avaliação da ajuda prestada pelo voluntário, traria uma melhora para o projeto no longo prazo, o que traria como resultado melhor atendimento aos usuários assistidos e escalabilidade do projeto.

Concomitantemente a essas melhorias, outra implementação relevante seria o teste com pessoas deficientes visuais. Através deste teste, seria possível medir a eficácia do projeto desenvolvido, assim como suas limitações e identificação dos principais pontos de melhorias.

REFERÊNCIAS

ABADI, Martín *et al.* **TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems**. [*S.l.*], 2016. Disponível em: https://research.google/pubs/pub45166/.

AGGARWAL, Charu C. **Neural Networks and Deep Learning**: A Textbook. 1. ed. [*S.l.*]: Springer, 2018. P. 1.

AGORA. **Video Calling**. California, 2022. Disponível em: https://docs.agora.io/en/video-calling/overview/product-overview?platform=android. Acesso em: 16 out. 2022.

ALEXANDRU, Dumbravan. **Clean Android Architecture**: Take a layered approach to writing clean, testable, and decoupled Android applications. 1. ed. [*S.l.*]: Packt Publishing, 2022. P. 233–240.

ANDERSON, David J. **Kanban: Successful Evolutionary Change for your Technology Business: Successful Evolutionary Change for your Technology Business**. 7. ed. [*S.l.*]: Blue Hole Press, 2010. P. 28–31.

BARI, Neha; KAMBLE, Nilesh; TAMHANKAR, Parnavi. **Android Based Object Recognition and Motion Detection to Aid Visually Impaired**. [*S.l.*], 2014. Disponível em: http://warse.org/pdfs/2014/ijacst063102014.pdf.

BIGDATACORP. **Acessibilidade em apps com mais de 10 milhões de downloads no Brasil**. São Paulo, 2020. Disponível em: https://bigdatacorp.com.br/estudo-acessibilidade-nos-aplicativos-com-mais-de-10-milhoes-de-downloads-no-brasil/#:~:text=O%5C%20levantamento%5C%20apurou%5C%20que%5C%2C%5C%20dentre,interface%5C%20visual%5C%20que%5C%20poderiam%5C%20apresentar.. Acesso em: 10 nov. 2022.

COHEN, Joseph Paul. **BlindTool – A mobile app that gives a "sense of vision"to the blind with deep learning**. [*S.l.*], 2015. Disponível em: https://github.com/ieee8023/blindtool. Acesso em: 6 nov. 2022.

DA SILVA, Sandra Rúbia; PEREIRA, Camila Rodrigues. **O CONSUMO DE SMARTPHONE ENTRE JOVENS DE CAMADAS POPULARES**. [*S.l.*], 2015. Disponível em: http://revistazcultural.pacc.ufrj.br/wp-content/uploads/2015/05/O-

consumo-de-smartphone-entre-jovens-de-camadas-populares-_-Revista-Z-Cultural.pdf. Acesso em: 10 nov. 2022.

DEVELOPERS, Google. **Desenvolver apps Android com o Kotlin**. [*S.l.*], 2022. Disponível em: https://developer.android.com/kotlin?hl=pt-br. Acesso em: 10 nov. 2022.

EVANS, Eric. **Domain-Driven Design: Tackling Complexity in the Heart of Software**. 1. ed. [*S.l.*]: Addison-Wesley Professional, 2003. P. 40–44.

EYES, Be My. **See the world together**: Be My Eyes connects people needing sighted support with volunteers and companies through live video around the world. [*S.l.*], 2023. Disponível em: https://www.bemyeyes.com/. Acesso em: 10 fev. 2023.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Tese (Doutorado) – University of California.

FIREBASE. **Visão geral da arquitetura do FCM**. [*S.l.*], 2022. Disponível em: https://firebase.google.com/docs/cloud-messaging/fcm-architecture?hl=pt-br. Acesso em: 15 out. 2022.

FOWLER, Martin. **BoundedContext**. [*S.l.*], 2014. Disponível em: https://martinfowler.com/bliki/BoundedContext.html. Acesso em: 15 out. 2022.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Data reafirma os direitos das pessoas com deficiência visual**. Brasil, 2010. Disponível em: http://portal.mec.gov.br/component/tags/tag/deficiencia-visual#:~:text=Desse%5C%20total%5C%2C%5C%206%5C%2C5%5C%20milh%5C%C3%5C%B5es,enxergar%5C%20(3%5C%2C2%5C%25). Acesso em: 19 nov. 2022.

JACOBSON, Daniel; BRAIL, Greg; WOODS, Dan. **APIs: A Strategy Guide**: Creating Channels with Application Programming Interfaces. 1. ed. [*S.l.*]: O'Reilly Media, 2011. P. 4.

LI, Fei-Fei. **ImageNet**. [*S.l.*], 2022. Disponível em: https://www.image-net.org/. Acesso em: 10 nov. 2022.

MARTIN, Robert C. **Arquitetura limpa: O guia do artesão para estrutura e design de software**. 1. ed. [*S.l.*]: Alta Books, 2019. P. 195–200.

MARTIN, Robert C. **Design Principles and Design Patterns**. [*S.l.*], 2000. Disponível em: https://web.archive.org/web/20150906155800/http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf.

MARTINEZ, Matias; GOIS MATEUS, Bruno. Why Did Developers Migrate Android Applications From Java to Kotlin? **IEEE Transactions on Software Engineering**, v. 48, n. 11, p. 4521–4534, 2022.

MICROSOFT. **The MVVM Pattern**. Washington, 2012. Disponível em: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/hh848246(v=pandp.10)?redirectedfrom=MSDN. Acesso em: 22 out. 2022.

PAUTASSO, Cesare; WILDE, Erik. **Web Services Foundations**: RESTful Web Services: Principles, Patterns and Emerging Technologies. [*S.l.*]: Springer, 2013. P. 31–51. ISBN 978-1-4614-7517-0.

SHEN, Darren Tan Yung. **Mobile Application for The Visually Impaired**. [*S.l.*], 2020. Disponível em: http://eprints.utar.edu.my/3948/1/16ACB03924_FYP.pdf. Acesso em: 6 nov. 2022.

TENSORFLOW. **Vida útil de uma solicitação de inferência de veiculação do TensorFlow**. [*S.l.*], 2021. Disponível em: https://www.tensorflow.org/tfx/serving/performance?hl=pt-br#sequence_diagram. Acesso em: 18 fev. 2023.