UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

APLICATIVO PARA O RECONHECIMENTO DE OBJETOS EM IMAGENS UTILIZANDO A API CLOUD VISION DESTINADO A PESSOAS PORTADORAS DE DEFICIÊNCIA VISUAL

JEAN CARLOS FRANCO

BLUMENAU
2018

JEAN CARLOS FRANCO

APLICATIVO PARA O RECONHECIMENTO DE OBJETOS EM IMAGENS UTILIZANDO A API CLOUD VISION DESTINADO A PESSOAS PORTADORAS DE DEFICIÊNCIA VISUAL

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof(a). Roberto Heinzle, Doutor – Orientador

BLUMENAU
2018

APLICATIVO PARA RECONHECIMENTO DE OBJETOS EM IMAGENS DESTINADO A PESSOAS PORTADORAS DE DEFICIÊNCIA VISUAL

Por

JEAN CARLOS FRANCO

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof(a). Roberto Heinzle – Orientador, FURB
Membro:	Prof(a). Everaldo Artur Grahl – FURB
Membro:	Prof(a). Aurélio Faustino Hoppe – FURB

Blumenau, 11 de dezembro de 2018

Dedico este trabalho a minha esposa Ana, que sempre me incentivou, a minha falecida mãe Iloí por ser um exemplo de pessoa, ao prof. Roberto Heinzle e as pessoas com algum nível de deficiência visual.

AGRADECIMENTOS

A Deus, por nos conceder dom da vida.

A minha esposa, pelo apoio incondicional, pela paciência e compreensão durante os momentos de estudo durante.

Ao meus amigos e colegas de trabalho, que me apoiaram em alguns momentos. Em especial, ao Rodrigo Felipe Moritz Petters e Aristides Cândido Junior que me auxiliaram com dúvidas sobre a plataforma Android.

Ao meu orientador, Roberto Heinzle, por sua orientação, apoio e direcionamento para o desenvolvimento deste trabalho.

Aos professores Dalton Solano dos Reis e Mauro Mattos, por suas conversas e dicas sobre o tema do meu trabalho.

“If you don't know, the thing to do is not to get scared, but to learn.”

Ayn Rand

RESUMO

No Brasil existem aproximadamente 35 milhões de pessoas com algum nível de deficiência visual, deste montante mais de 6 milhões de pessoas possuem grande dificuldade para enxergar e cerca de 500 mil pessoas não enxergam nada, condição que pode gerar dificuldades na realização das tarefas diárias destas pessoas. Diante deste fato, este trabalho apresenta o desenvolvimento de um aplicativo móvel em forma de tecnologia assistiva para o reconhecimento de objetos em imagens, sendo destinado às pessoas com algum nível de deficiência visual. O aplicativo é destinado à plataforma Android, foi desenvolvido utilizando a linguagem de programação Java e faz uso da API Cloud Vision do Google para realizar a análise de uma imagem e apresentar ao usuário detalhes do seu conteúdo. Por ser destinado a pessoas com deficiência visual, o aplicativo apresenta ao usuário instruções por voz a cada etapa executada e descreve o conteúdo da imagem através da sintetização de voz. O aplicativo foi capaz de descrever de forma convincente cerca de 67% dos objetos e cenas analisadas, demonstrando o potencial da proposta como uma tecnologia assistiva e a assertividade da API Cloud Vision, entretanto os testes foram realizados em usuários com os olhos vendados, não foram aplicados em deficientes visuais. Os resultados alcançados foram satisfatórios e os objetivos do trabalho foram atingidos.

Palavras-chave: Deficiência visual. Reconhecimento de objetos. Tecnologia assistiva. Google Cloud Vision.

ABSTRACT

In Brazil, there are approximately 35 million people with some level of visual impairment. Of this amount, more than 6 million people have great difficulty to see and about 500 thousand people can't see anything, a condition that can generate difficulties in carrying out daily tasks. Because of this fact, this work presents the development of a mobile application in the form of assistive technology for the recognition of objects in images, being intended for people with some level of visual impairment. This application is intended for the Android platform and was developed using Java programming language. It also uses Google Cloud Vision API to perform analysis of an image and present to the user details of its content. Since it is intended for the visually impaired, the application presents the user with voice instructions at each step performed and describes the content of the image through speech synthesis. The application was able to convincingly describe about 67% of the objects and scenes analyzed, demonstrating the potential of the proposal of an assistive technology and the assertiveness of the Cloud Vision API. However, the tests were performed on users blindfolded, not on the visually impaired. The results achieved were satisfactory and the objectives of the work were achieved.

Keywords: Visual impairment. Object recognition. Assistive technology. Google Cloud Vision.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

API – Application Programming Interface (Interface de Programação de Aplicações)

ML – Machine Learning

TTS - Text-To-Speech

VC – Visão Computacional

SUMÁRIO

1 INTRODUÇÃO

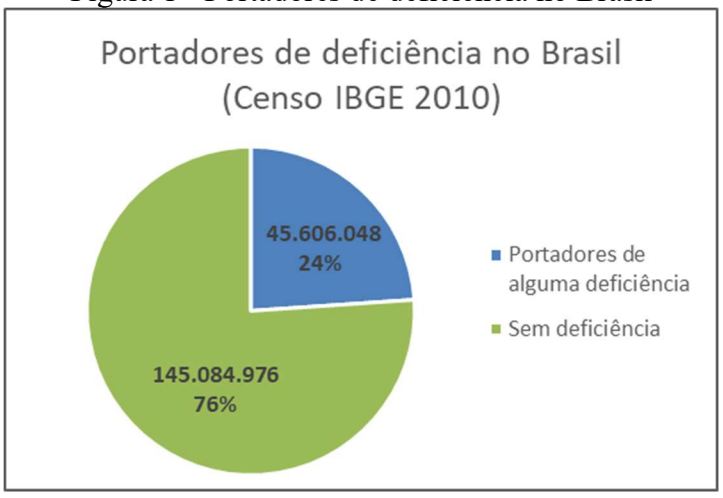
A legislação brasileira, através da lei Brasileira de Inclusão de Pessoa com Deficiência, apresenta a seguinte definição para portador de deficiência:

> Art. 2º Considera-se pessoa com deficiência aquela que tem impedimento de longo prazo de natureza física, mental, intelectual ou sensorial, o qual, em interação com uma ou mais barreiras, pode obstruir sua participação plena e efetiva na sociedade em igualdade de condições com as demais pessoas (BRASIL, 2015, p.1).

Esta legislação foi criada visando a inclusão social e a cidadania dos portadores de deficiência. Sua proposta é assegurar e promover, em condições de igualdade, os direitos e liberdades fundamentais do portador de deficiência. Dentre os assuntos tratados na lei, está o compromisso de disponibilizar, criar, facilitar e fomentar a utilização de serviços de tecnologia assistiva que maximizem a autonomia, a mobilidade e a qualidade de vida das pessoas portadoras de deficiência.

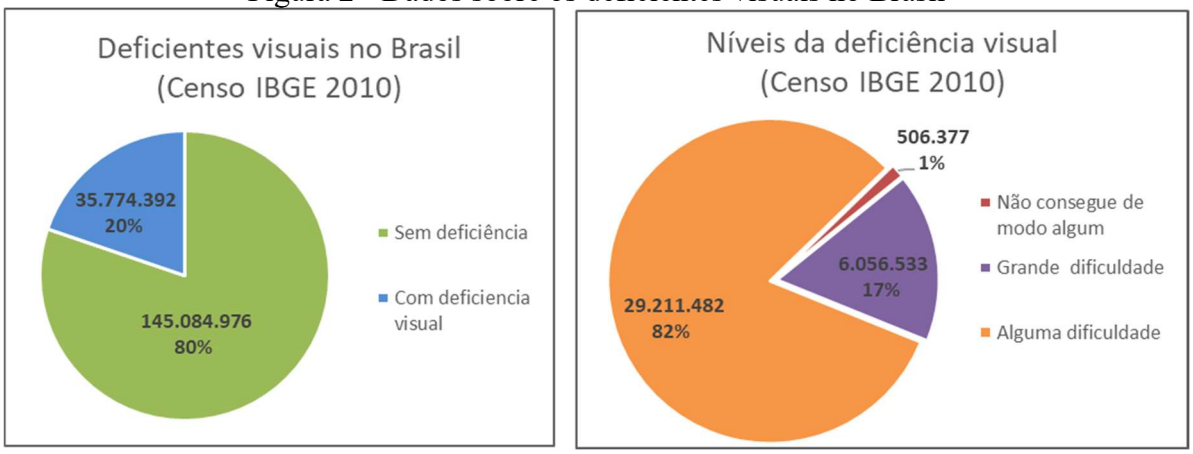
O censo demográfico do ano de 2010, realizado pelo Instituto Brasileiro de Geografia e Estatística (IGBE), aponta que no Brasil existem cerca de 45 milhões de pessoas com pelo menos algum tipo de deficiência, conforme demonstrado na Figura 1 este número equivale a 24% da população. O censo apurou as pessoas com deficiência visual, auditiva, motora e intelectual. A apuração também classifica cada tipo de deficiência em três níveis: não consegue de modo algum, grande dificuldade ou alguma dificuldade, a exceção é a deficiência mental/intelectual que não está classificada em níveis. A contagem de pessoas com algum nível de deficiência visual totaliza mais de 35 milhões de indivíduos e representa cerca de 20% da população. Dentre os deficientes visuais, cerca de 500 mil não conseguem enxergar nada e cerca de 6 milhões de pessoas possuem grande dificuldade (IBGE, 2018). Os detalhes do censo sobre os deficientes visuais são exibidos na Figura 2.

Figura 1 - Portadores de deficiência no Brasil



Fonte: Elaborado pelo autor com base nos dados de IBGE (2018).

Figura 2 - Dados sobre os deficientes visuais no Brasil



Fonte: Elaborado pelo autor com base nos dados de IBGE (2018).

Vanzin (2016) descreve que as tecnologias assistivas representam um conjunto de recursos oferecidos às pessoas com deficiência para que elas superem as barreiras que limitam o acesso a recursos aos quais tem direito, melhorando a sua qualidade de vida. Stora (2016) cita como exemplos de tecnologia assistiva uma bengala eletrônica para deficientes visuais, um pé articulado com amortecedor e antiderrapante, softwares como o Virtual Vision, que permitem pessoas com deficiência a utilizar computadores com o sistema operacional Windows.

Diante do exposto e da quantidade expressiva de pessoas com deficiência visual, este trabalho apresenta o desenvolvimento de um aplicativo móvel destinado aos portadores de deficiência visual capaz de realizar o reconhecimento de objetos em imagens através da API Cloud Vision com o intuito de oferecer algum auxílio em suas atividades no dia-a-dia. Através desta solução o usuário poderá capturar uma foto do ambiente aonde está e receber informações dos objetos contidos no ambiente como, por exemplo, animais e veículos.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um aplicativo móvel para a plataforma Android que permita realizar o reconhecimento de objetos em imagens utilizando a API Cloud Vision.

Os objetivos específicos são:

a) disponibilizar uma interface para aplicação móvel que permita a captura, através da câmera do celular, da imagem a ser analisada;
b) oferecer um mecanismo para reconhecer os objetos presentes na imagem capturada e apresentar o seu conteúdo ao usuário;
c) fornecer uma interface que permita a utilização do aplicativo por deficientes visuais através de gestos e com menus descritos por sintetização de voz.

1.2 ESTRUTURA

Este trabalho é composto por quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusão. O primeiro capítulo apresenta a introdução ao tema do trabalho e os seus objetivos. O segundo capítulo descreve a fundamentação teórica utiliza sobre os temas abordados no trabalho: computação gráfica, reconhecimento de imagens, síntese de voz e a API Cloud Vision do Google. No terceiro capítulo são demonstrados os detalhes da especificação do aplicativo e da sua implementação. Por fim, o quarto capítulo descreve as conclusões do trabalho realizado, os resultados obtidos e propõe sugestões para o desenvolvimento de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo é composto por três seções e tem como objetivo explorar os principais assuntos para a realização deste trabalho. A seção 4.1 aborda uma visão geral sobre a área de computação gráfica e o reconhecimento de imagens, enquanto a seção 4.2 trata sobre a sintetização de voz, que será utilizada para apresentar o conteúdo da imagem ao usuário. Por último, a seção 4.3 apresenta o serviço Cloud Vision do Google.

2.1 COMPUTAÇÃO GRÁFICA E O RECONHECIMENTO DE IMAGENS (VISÃO COMPUTACIONAL)

A computação gráfica (CG), segundo Conci (2008, p. 3), é composta por pelo menos 03 grandes áreas: a síntese de imagens (SI), o processamento de imagens (PI) e a análise de imagens (AI). Pode-se definir uma imagem como sendo uma representação visual de objetos e uma imagem pode ser adquirida (fotos, filmes, etc.) ou criada (pinturas, desenhos, esculturas, etc.).

Conci (2008, p. 4) também detalha a abrangência de cada área da computação gráfica:

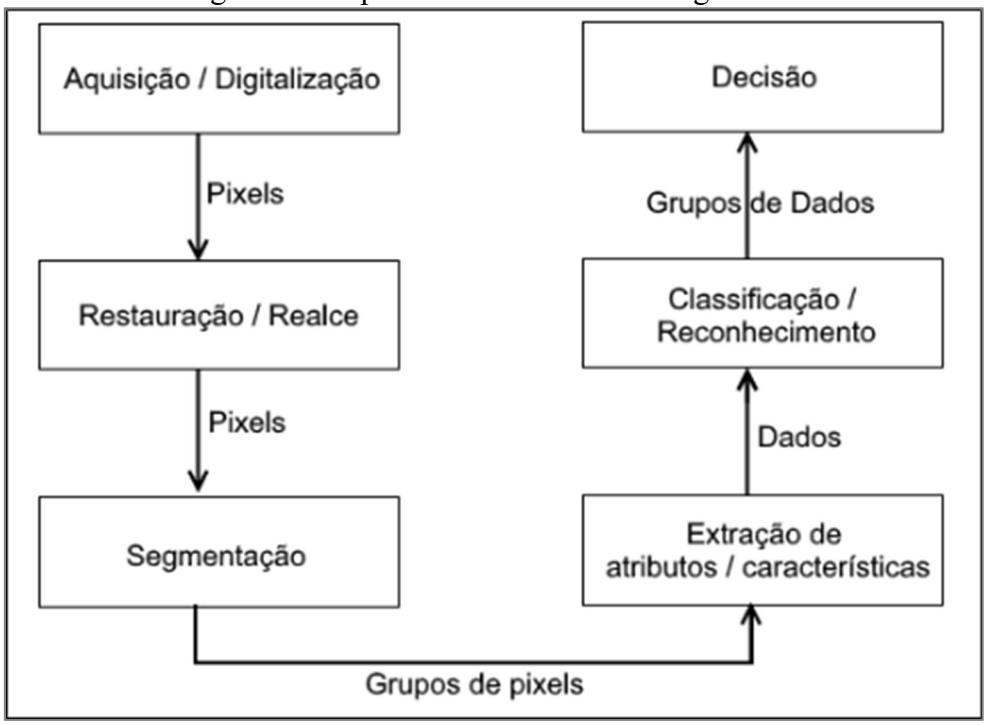
a) processamento de imagens: realiza a manipulação de imagens já existentes. Nesta área a imagem é tanto um dado de entrada como de saída. O processamento de imagens inclui tópicos como diminuição de ruídos, realce, restauração e recuperação de imagens;
b) síntese de imagens: trata da geração de imagens a partir de dados;
c) análise de imagens: trata do processo do mapeamento de parâmetros descritivos (em geral numéricos) que identifiquem informações importantes da imagem. Dentro da área de análise de imagens encontra-se a área de visão computacional (VC).

A visão computacional pode ser definida como a utilização de um conjunto de métodos e técnicas através dos quais os sistemas computacionais podem ser capazes de interpretar imagens (WANGENHEIM, 1998 apud HENSEL, 2012, p. 17). Os problemas tratados com visão computacional, segundo Caramori (2012), não envolvem apenas a busca de objetos em imagens, a visão computacional também é utilizada para buscar informações sobre os objetos encontrados, como por exemplo: a quantidade de pessoas que entraram e saíram em determinado estabelecimento ou a quantidade de frutos com má qualidade em um conjunto misto de frutos.

A Figura 3 apresenta uma sequência tradicional das etapas aplicadas nos sistemas de visão computacional descritas por Conci (2008) para a identificação e classificação de objetos.

O processo tem início a partir da captura/aquisição da imagem, realiza o realce dos detalhes de imagem de acordo com o interesse da análise, efetua a segmentação de pontos ou regiões da imagem (ex.: isolar fundo e objeto), aplica a extração dos atributos e características de cada segmento da imagem (ex.: total de objetos contidos, dimensões, geometria e textura). Após a obter os descritores e os segmentos da imagem inicia-se a etapa de classificação e reconhecimento: o processo de reconhecimento consiste em distinguir objetos na imagem agrupando os parâmetros obtidos de acordo com a sua semelhança para cada região de pixels, enquanto que o processo de classificação determina que o objeto pertence a determinado grupo com base em um banco de imagens. A etapa de decisão está relacionada a tomada de decisões com base nas informações obtidas e pode ser feita a partir de indagações simples relacionadas aos objetos presentes na imagem ou vinculada a algoritmos de inteligência artificial.

Figura 3 - Etapas de um sistema de VC genérico



Fonte: Adaptado de Conci (2008, p. 51).

2.2 SINTESE DE VOZ

A fala é um meio de comunicação com grande importância na interação humana. A fala é composta por uma sequência de sons que são gerados pelo sistema vocal sendo utilizada como uma ferramenta vital na interação entre os seres humanos. A importância da fala para a comunicação tem direcionado os cientistas a entender e imitar as variações dos sinais de voz. A arte de produzir sinais de fala por meios artificiais é conhecida por síntese de voz (RAHIM 1994, p. 1).

Goll (2016) descreve que tanto o reconhecimento quanto a síntese de voz são excelentes formas de imersão e acessibilidade das pessoas de forma geral, pois permitem que a interação com computadores seja feita de uma maneira mais natural e intuitiva. Também é um meio de permitir que pessoas com deficiência visual possam utilizar computadores de forma muito melhor.

A evolução da capacidade de processamento dos dispositivos eletrônicos tem possibilitado a utilização de tecnologias de voz. As tecnologias mais proeminentes atualmente são Text-To-Speech (TTS), que consiste na conversão de textos em síntese de voz, e Automatic Speech Recognition (ASR), onde ocorre o reconhecimento automático de voz para texto. (GOULART, 2016).

A tecnologia de TTS para a sintetização de voz pode ser incorporada em outros sistemas através de serviços de Aplication Programmig Interface (API), como por exemplo, a plataforma Google Cloud Platform (Google LLC, 2018a) que possui o serviço Cloud Text-to-Speech (Google LLC, 2018b) que fornece recursos de sintetização de voz natural em vários idiomas, incluindo o português. Este serviço auxilia na criação de interações realistas com os usuários em vários aplicativos e dispositivos.

Em seu artigo, Hashimoto (2011) destaca a importância da qualidade da fala sintetizada para os sistemas com diálogo falado. Descreve que a qualidade da fala sintetizada é uma caraterística importante porque os usuários não conseguem entender o retorno do sistema caso a qualidade da voz sintetizada seja baixa ou não seja natural. Por fim, conclui que a naturalidade e inteligibilidade de uma fala sintetizada é fortemente influenciada pela qualidade da formação das frases reproduzidas.

2.3 API CLOUD VISION DO GOOGLE

Hosseini (2017), descreve que as técnicas de `machine learning` (ML) estão sendo extensivamente implementadas em tarefas de visão computacional, principalmente na área de classificação visual, onde novos algoritmos relataram alcançar ou até mesmo superar o desempenho humano. Este sucesso dos algoritmos de ML geraram uma explosão na demanda por estes tipos de serviço. De olho nesta demanda, empresas como a Google, Amazon, Microsoft e BigML desenvolveram ferramentas de `machine learning` baseados na nuvem, que são ofertadas como serviço e buscam simplificar e ampliar a utilização dos algoritmos de `machine learning`. Esta abordagem permite que usuários e empresas possam se beneficiar rapidamente dos serviços de machine learning em suas aplicações sem precisar treinar ou hospedar seus próprios modelos.

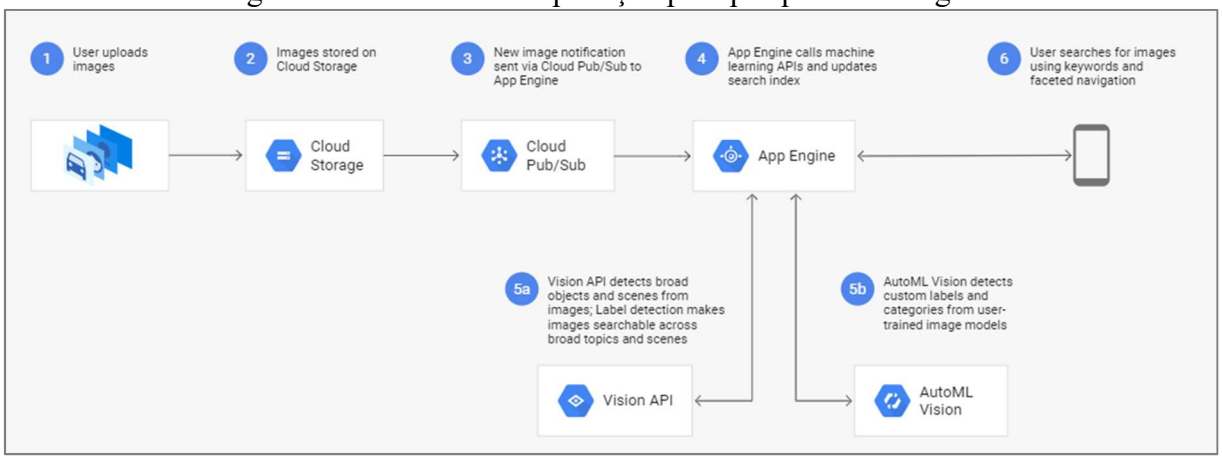
A API Cloud Vision é um serviço presente na plataforma Google Cloud, esta é a plataforma de computação na nuvem do Google que oferece várias funcionalidades, dentre elas: máquinas virtuais, ferramentas inteligentes de IOT (internet of things – internet das coisas), serviços de `machine learning` (aprendizado de máquina), armazenamento na nuvem, ferramentas grandes volumes de dados entre outros serviços (GOOGLE, 2018c).

O portal Cloud Vision, presente na plataforma Google Cloud, descreve que o serviço disponibilizado permite a análise avançada de imagens, informa que a plataforma oferece modelos pré-treinados por meio de uma API ou a capacidade de criar modelos personalizados. Por fim, descreve o funcionamento da API da seguinte forma:

> A API do Cloud Vision encapsula modelos avançados de aprendizado de máquina em uma API REST fácil de usar, o que permite aos desenvolvedores entender o conteúdo de imagens. Essa API rapidamente classifica as imagens em milhares de categorias, (por exemplo: “veleiro”), detecta objetos e rostos individuais e extrai palavras impressas contidas nas imagens. Você pode criar metadados no seu catálogo de imagens, moderar conteúdo ofensivo ou ativar novos cenários de marketing usando a análise de sentimento das imagens (GOOGLE, 2018d, p. 1).

Na sequência o portal Cloud Vision, apresenta como exemplo o fluxo de uma aplicação que realiza a pesquisa de imagens com base nos tópicos, cenas e categorias identificadas nas imagens analisadas. Este fluxo é demonstrado na Figura 4, nele o usuário efetua o upload das imagens a serem analisadas para o serviço Cloud Storage que aciona as APIs de machine learning que realizam a detecção dos objetos e categorias da imagem. Ao final, este processo permite que o usuário realize a busca por imagens através de palavras-chave através do celular.

Figura 4 - Fluxo de uma aplicação para pesquisa de imagens



Fonte: Google (2018d).

Em seu artigo, Hosseini (2017) confirma que a API Cloud Vision permite o retorno dos rótulos de uma imagem analisada, identifica os textos contidos na imagem e detecta os rostos dentro da imagem. Informa que a API permite determinar a probabilidade de a imagem conter conteúdo impróprio, incluindo conteúdo adulto, falsificado, médico ou de violência. Estes

recursos na forma de serviço, permitem a criação de uma séria de aplicações, tais como: classificação automática de imagens, retorno de texto de documentos escaneados, classificação de imagens em redes sociais, entre outros usos.

2.4 TRABALHOS CORRELATOS

Neste capítulo são apresentados três trabalhos com características semelhantes aos principais objetivos do estudo proposto. O primeiro é um aplicativo para o reconhecimento de cartas de jogo tornando um jogo de mesa acessível para deficientes visuais (KRAEMER, 2017). O segundo é um aplicativo para sugestão de receitas a partir do reconhecimento de imagens em alimentos (HEINZLE, 2017) e o terceiro é um sistema para reconhecimento de expressões faciais (SEEFELD, 2016).

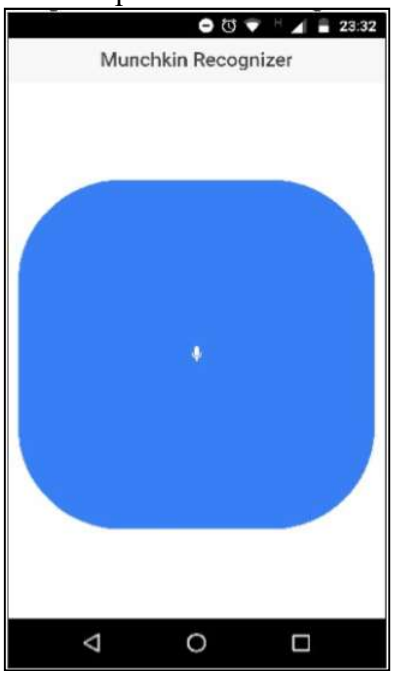
2.4.1 Tecnologia assistiva: tornando jogo de mesa acessível para cegos com auxílio de aplicativo móvel de reconhecimento de imagem

Kramer (2017) descreve que há uma grande preocupação no mundo e no Brasil em relação a inclusão social e a acessibilidade. Para tanto existe no Brasil um comitê para tratar assuntos de tecnologia e acessibilidade: o Comitê de Ajudas Técnicas (CAT), instituído em 16 de novembro de 2006.

Com o objetivo de auxiliar na acessibilidade de pessoas portadoras de deficiência visual aos jogos de mesa com cartas, Kramer (2017) desenvolveu um aplicativo para o reconhecimento dos dados presentes no jogo de mesa chamado Munchkin. Utilizando uma tecnologia que permite o reconhecimento óptico de caracteres (Optical Character Recognition - OCR), a solução permite identificar os textos e valores descritos nas cartas a partir da captura de uma imagem da carta através da câmera do celular. O aplicativo foi desenvolvido de forma híbrida para as plataformas Android e IOS através do framework Ionic.

A tela inicial do aplicativo é destinada ao reconhecimento de comandos de voz (Figura 5). Ao acionar o botão, o reconhecimento de voz fica ativo por 05 segundos e reconhece os comandos de voz `Tirar foto` e `Repetir`. Qualquer comando diferente é considerado inválido. Esta é a tela principal do aplicativo e é responsável pela execução das funcionalidades da solução.

Figura 5 - Tela para reconhecimento de voz



Fonte: Kramer (2017).

Ao identificar o comando `Tirar foto` a aplicação aciona a câmera do celular para a que o usuário capture a imagem de uma carta (Figura 6). Em seguida a solução efetua a análise da imagem e do conteúdo da carta retornando ao usuário, através de comandos de voz, os dados identificados. A Figura 7 apresenta uma tela do modo desenvolvedor da solução, contendo a descrição das informações capturadas da imagem. Esta tela foi criada para facilitar a mensuração dos resultados, visto que a resposta para o usuário final é através de comandos de voz.

Figura 6 - Capturando imagem da carta

Fonte: Kramer (2017).

Figura 7- Tela do modo desenvolvedor com os dados identificados

Fonte: Kramer (2017).

2.4.2 Cogncook: aplicativo para reconhecimento de imagens de alimentos e sugestão de receitas culinárias utilizando ingredientes disponíveis

Heinzle (2017) desenvolveu um aplicativo móvel que faz a sugestão de receitas culinárias a partir do reconhecimento de imagens de alimentos disponíveis. Segundo o autor, o

trabalho foi motivado pela constatação de uma das consequências da vida moderna: as pessoas têm pouco cuidado com a alimentação devido à falta de tempo. "A alimentação contemporânea, caracterizada pelo estilo de vida moderno, é marcada pela escassez de tempo para preparo e consumo de alimentos, pelo surgimento de alimentos prontos para o consumo, com novas técnicas de conservação e preparo, que agregam tempo e trabalho" (PINHEIRO, 2005 apud HEINZLE, 2017, p. 12). O aplicativo proposto tem como objetivo facilitar o acesso às pessoas a novas receitas culinárias utilizando os alimentos disponíveis em sua casa.

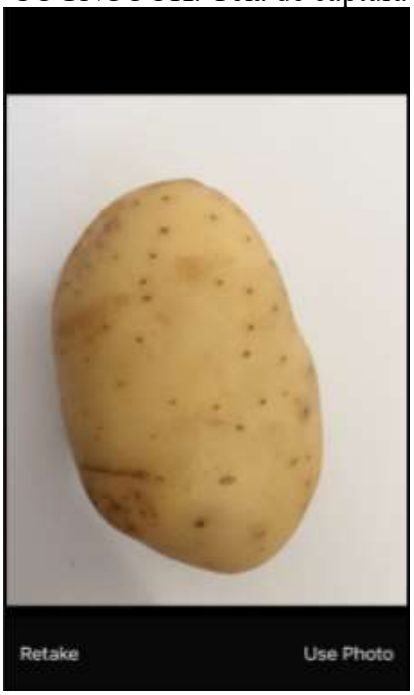
A solução desenvolvida por Heinzle (2017) efetua o reconhecimento dos alimentos presentes através de imagens, utilizando a ferramenta de visão computacional IBM Visual Recognition e realiza a sugestão das receitas com os alimentos identificados a partir do uso da API Spoonacular. O aplicativo possui uma tela para busca de receitas, que pode ser realizada a partir do preenchimento manual dos alimentos desejados ou do reconhecimento do alimento a partir de uma imagem (Figura 8).

Figura 8 – COGNCOOK: Tela para busca de receitas

Fonte: Heinzle (2017).

Caso o usuário opte pelo reconhecimento do alimento a partir de uma imagem, ele deve capturar a imagem (Figura 9) e aguardar o processamento. Caso o aplicativo retorne mais do que uma classificação para a imagem, é apresentada uma tela para que o usuário faça a seleção da opção correta.

Figura 9- COGNCOOK: Tela de captura alimento



Fonte: Heinzle (2017).

O usuário poderá adicionar vários ingredientes antes de realizar a busca das receitas. Ao realizar a busca de receitas, o aplicativo retorna para o usuário as receitas identificadas (Figura 10) e permite que o usuário consulte os detalhes de cada receita.

Figura 10 - COGNCOOK - receitas identificadas

Fonte: Heinzle(2017).

O resultado dos testes publicados por Heinzle (2017) informam que o reconhecimento dos alimentos nas imagens foi realizado com sucesso em 94,2% dos testes realizados. Já a busca de receitas a partir dos ingredientes apresentou uma taxa de sucesso de 96%.

2.4.3 E-Motiv: protótipo de sistema para reconhecimento de expressões faciais

Seefeld (2016) apresenta a criação de um protótipo para o reconhecimento de expressões faciais universais. O objetivo do trabalho é fornecer uma forma alternativa para a coleta de dados do usuário e que possa ser utilizada em outros sistemas de informação.

Ekman (1999 apud SEEFELD, 2016, p. 14) cita que o reconhecimento de emoções através de expressões, gestos ou da fala faz parte da interação social humana, mesmo que forma inconsciente. Descreve que o reconhecimento das emoções a partir das expressões faciais é considerada, dentre as formas citadas anteriormente, uma das formas mais primitivas e utilizadas pelo homem.

O E-Motiv utiliza técnicas de visão computacional e processamento de imagens digitais para realizar a detecção de faces em imagens e a extração de suas características. Para realizar a classificação das expressões faciais, o E-Motiv utiliza uma Rede Neural Artificial do tipo perceptron de multicamadas. Na Figura 11 Seefeld (2016) demonstra os contornos do rosto considerados pelo E-Motiv na análise de uma imagem, o traçado do exemplo é composto por 68 pontos faciais que serão utilizados para identificar a expressão facial.

Figura 11- Traçado obtido a partir dos pontos faciais

Fonte: Seefeld (2016).

O protótipo do E-Motiv foi capaz de reconhecer até duas expressões faciais, além da expressão neutra, com até 89% de taxa de acerto, comprovando que as técnicas aplicadas são viáveis para o reconhecimento de expressões.

3 DESENVOLVIMENTO

Este capítulo aborda os aspectos referentes à construção do aplicativo denominado Franco Vision, fundamentado nos conceitos apresentados no capítulo anterior. A seção 3.1 apresenta os requisitos funcionais e não funcionais. Na seção 3.2 encontra-se os diagramas de caso de uso e de classes, que descrevem o funcionamento do aplicativo e a sua especificação. A seção 3.3 descreve as técnicas e ferramentas utilizadas no desenvolvimento do aplicativo e apresenta os principais trechos de código. Por último, a seção 3.4 expõe os resultados auferidos após o desenvolvimento do aplicativo e apresenta a comparação com os trabalhos correlatos apresentados na seção 2.4.

3.1 REQUISITOS

Na sequência são apresentados os requisitos funcionais (RF) e os requisitos não funcionais (RNF) do trabalho desenvolvido:

a) permitir o cadastro de um novo usuário através de um e-mail e senha (RF);
b) permitir capturar uma imagem do ambiente (RF);
c) permitir selecionar uma imagem salva no dispositivo (RF);
d) transmitir e processar a imagem selecionada utilizando a API Google Vision (RF);
e) exibir o resultado da análise da imagem ao usuário (RF);
f) converter para comandos de voz o nome dos objetos identificados na análise da imagem (RF);
g) descrever através de comandos de voz as instruções para que o usuário possa utilizar o aplicativo (RF);
h) permitir o acionamento da câmera através de gestos na tela, utilizando a opção de duplo clique (RNF);
i) ser implementado para a plataforma Android utilizando a linguagem de programação Java (RNF);
j) utilizar a API Cloud Vision do Google para o reconhecimento de objetos na imagem (RNF);
k) utilizar a plataforma Firebase do Google para realizar a autenticação e cadastro do usuário (RNF).

3.2 ESPECIFICAÇÃO

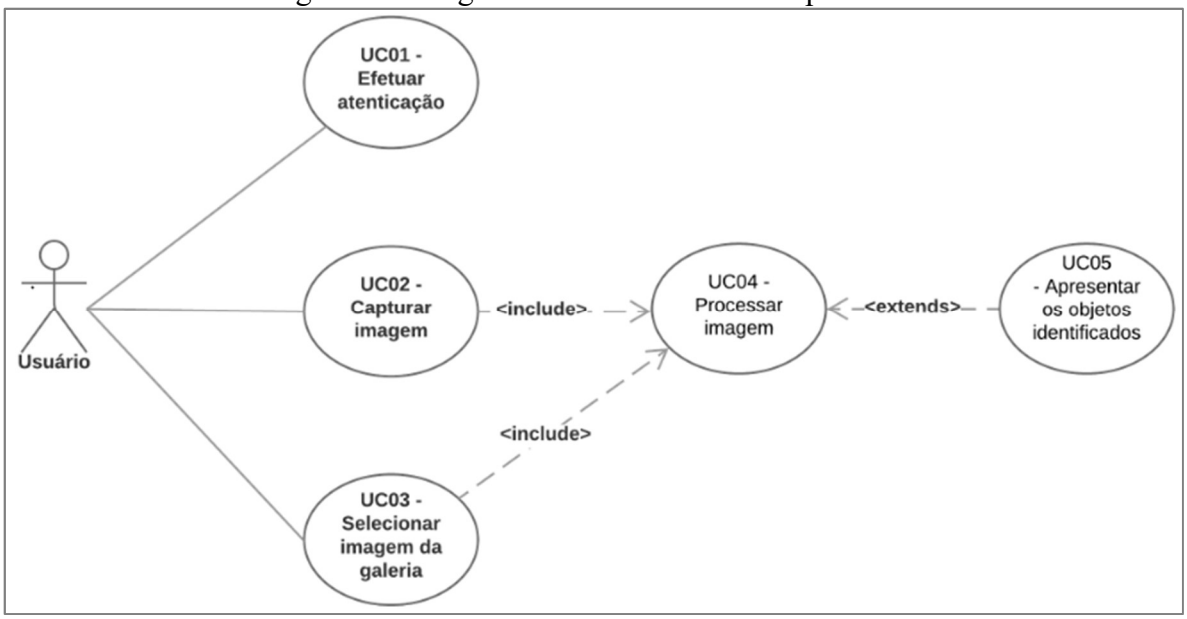
Esta seção apresenta a especificação do aplicativo e suas funcionalidades, sua modelagem foi realizada utilizando a abordagem orientada a objetos através da ferramenta Draw.io, utilizando-se a Unified Modeling Language (UML). No desenvolvimento deste trabalho foram criados os diagramas de caso de uso e de classes. No final desta seção é apresentada a arquitetura do aplicativo.

3.2.1 Diagrama de caso de uso

A Figura 12 apresenta o diagrama de casos de uso do aplicativo, desenvolvido a partir do levantamento dos requisitos da aplicação e representa as funcionalidades disponíveis ao usuário da aplicação. Como pode ser observado, são sete casos de uso e apenas um único setor.

O caso de uso `UC01 - Efetuar Autenticação` é responsável pela autenticação dos usuários existentes e pelo cadastro de novos usuários. No caso de uso `UC02 - Capturar Imagem` o usuário pode capturar a imagem a ser analisada através da câmera do celular, o acionamento da câmera pode ocorrer através do comando de duplo clique na tela ou através de um botão. O caso de uso `UC03 – Selecionar imagem` da galeria permite que o usuário selecione uma imagem da galeria, caso ele não deseje utilizar a câmera do celular. No caso de uso `UC04 – Processar imagem` o aplicativo faz a requisição de análise do conteúdo da imagem. No caso de uso `UC05 – Apresentar os objetos identificados` são listados em forma de texto os objetos e categorias identificadas na imagem, o aplicativo também realiza a descrição por sintetização de voz dos objetos e categorias identificadas. O acionamento do caso de uso `UC05 – Apresentar os objetos identificados` somente ocorrerá caso o processo de análise da imagem, realizado no caso de uso `UC04 – Processar imagem`, seja realizado com sucesso e retorne dados sobre o conteúdo da imagem.

Figura 12- Diagrama de casos de uso do aplicativo

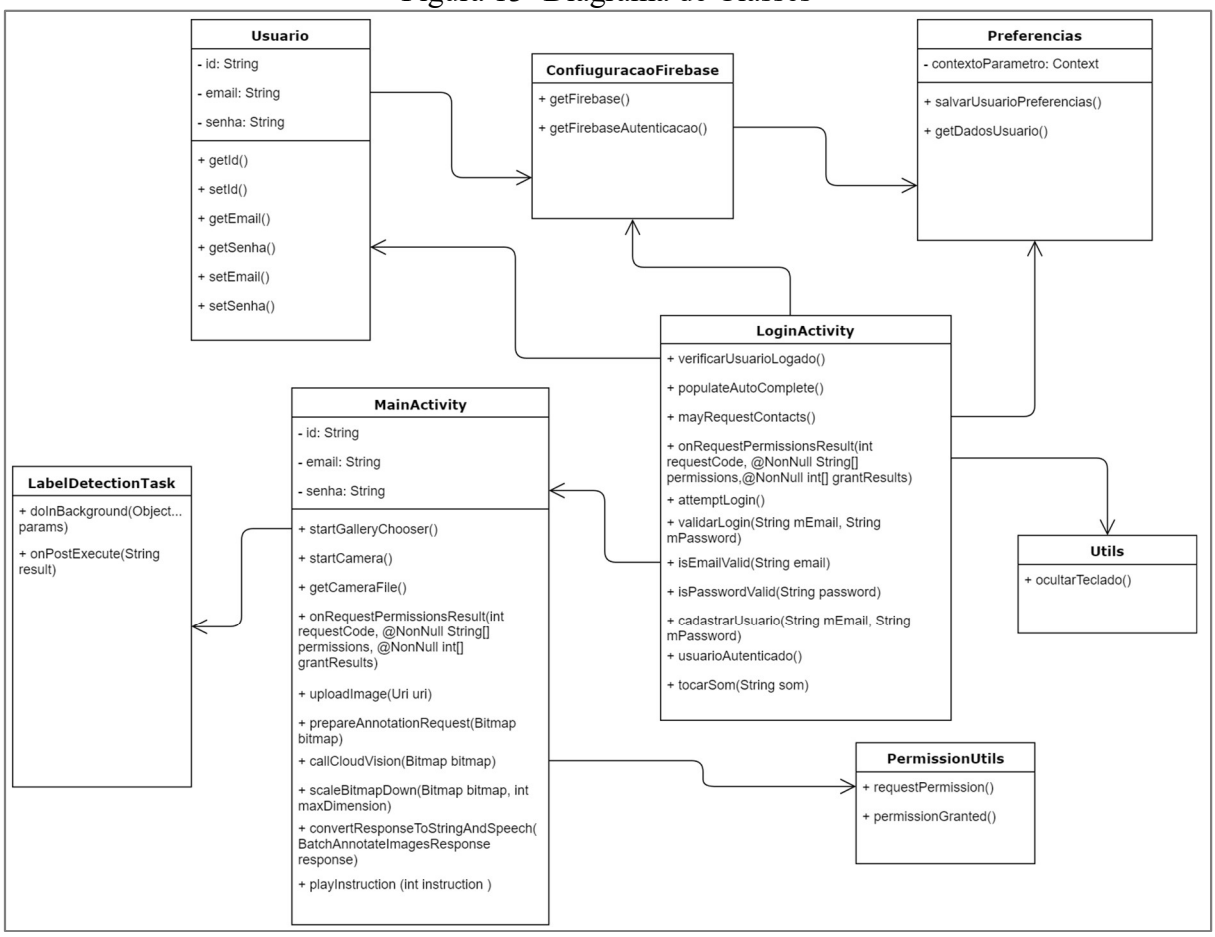


Fonte: Elaborado pelo autor.

3.2.2 Diagrama de Classes

O diagrama de classes permite visualizar como as classes do aplicativo estão estruturadas, assim como o seu relacionamento. Esta seção descreve as classes do aplicativo Franco Vision, conforme a Figura 13.

Figura 13- Diagrama de Classes



Fonte: Elaborado pelo Autor

A classe `LoginActivity` é acionada no início da aplicação e é responsável pela autenticação e cadastro dos usuários do aplicativo. O método `verificarUsuarioLogado` valida se há um usuário autenticado no aplicativo, caso exista, direciona o usuário para a tela principal do aplicativo acionando a classe `MainActivity` e pulando a etapa de preenchimento do e-mail do usuário e senha. O método `cadastrarUsuario` realiza o cadastro de um novo usuário caso o e-mail informado não exista na base de usuários, o cadastro do usuário é realizado através da classe `Usuario`.

A classe `ConfiguracaoFirebase` é responsável pela conexão do aplicativo com a plataforma Firebase. A plataforma Firebase é utilizada para registrar os usuários cadastrados na aplicação e realizar a autenticação dos usuários na abertura do aplicativo. A classe `Preferencias` é responsável por armazenar no celular os dados do último usuário autenticado no aplicativo, estes dados são utilizados para evitar que o usuário precise se autentificar a cada acesso ao aplicativo. As classes `Utils` é uma classe de apoio e auxilia na exibição do teclado na tela de autenticação, assim como a classe e `PermissionUtils` que realiza a requisição das

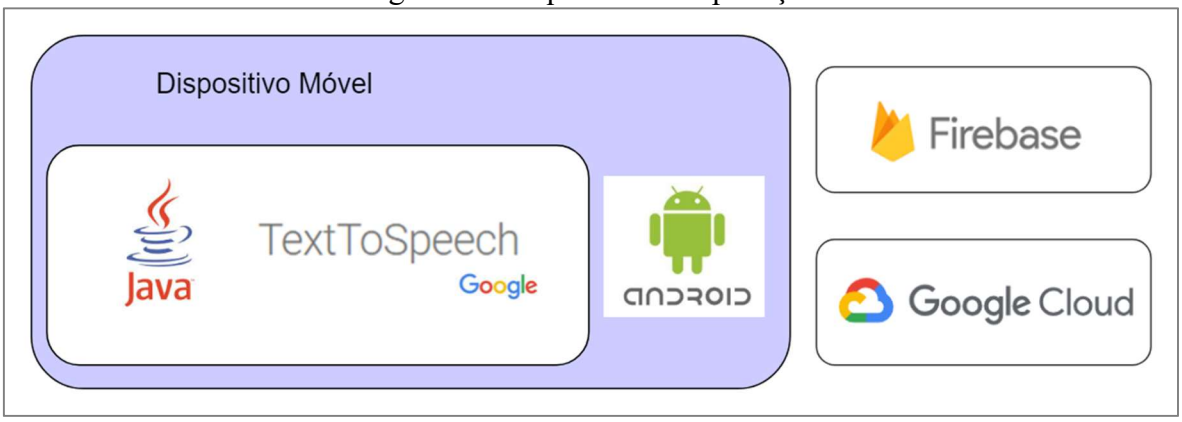
permissões de acesso aos recursos do Android, como por exemplo, o acesso à câmera e ao armazenamento do dispositivo.

A classe `MainActivity` é a classe principal da aplicação, ela é responsável por chamar o método `startCamera` que permite o usuário realizar a captura da imagem a ser analisada. Em seguida a classe obtém a imagem capturada pelo usuário através do método `getCameraFile`, efetua os preparativos para o processamento da imagem através dos métodos `uploadImage`, `scaleBitmapDown e prepareAnnotationRequest`. A análise da imagem é acionada através do método `callCloudVision`, este método efetua a chamada da classe `LabelDetectionTask` que realizada a análise da imagem de forma assíncrona. O método `convertResponseToStringAndSpeech` da classe `MainActivity` é acionado no retorno da análise da imagem e é responsável por exibir em forma de texto os objetos identificados e por efetuar a descrição dos objetos utilizando a sintetização de voz.

3.2.3 Arquitetura da aplicação

Para a elaboração deste trabalho foi utilizada a linguagem Java, a biblioteca TextToSpeech do Google e dois serviços de processamento na nuvem do Google: as plataformas Firebase e Google Cloud. A Figura 14 apresenta a arquitetura do aplicativo e a sua integração com as plataformas Firebase e Google Cloud do Google, o desenho da arquitetura foi elaborado através da ferramenta Draw.io.

Figura 14- Arquitetura da aplicação



Fonte: Elaborado pelo autor.

A plataforma Firebase do Google é utilizada como um banco de dados hospedado na nuvem para o armazenamento de todos os usuários cadastrados na aplicação, a aplicação também utiliza o serviço de autenticação do Firebase para realizar a autenticação dos usuários cadastrados. A plataforma Google Cloud (2018d) disponibiliza vários serviços de computação na nuvem, dentre os serviços presentes nesta plataforma está o Cloud Vision que disponibiliza funcionalidades de processamento de imagem utilizado na construção do aplicativo.

3.3 IMPLEMENTAÇÃO

Nesta seção são detalhadas as técnicas e ferramentas utilizadas no desenvolvimento deste trabalho, por fim é detalhada a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento deste aplicativo foi utilizada a linguagem de programação Java na versão 10. O ambiente de desenvolvimento utilizado foi Android Studio na versão 3.2.1, ferramenta oficial do Google para a criação de aplicativos para a plataforma Android. Além da interface de desenvolvimento, o ambiente do Android Studio permite o acesso à SDK do Android que disponibiliza várias APIs que facilitam o acesso aos recursos do dispositivo, como o acesso à câmera, a galeria de imagens e a sintetização de voz. O versionamento do código fonte durante o projeto foi realizado utilizando o sistema de controle de versão Git, através da plataforma de hospedagem Bitbucket.

A autenticação e o cadastro de usuários foram implementados utilizando os serviços `Authentication` e `Realtime Database` disponibilizados pela API da plataforma Firebase do Google. A utilização dos serviços do Firebase exige a criação de uma chave de acesso, que pode ser cadastrada no site da plataforma (GOOGLE, 2018e). O serviço de autenticação é disponibilizado de forma gratuita pela plataforma, enquanto que o plano gratuito do serviço de banco de dados possui algumas restrições, limitando o número de conexões simultâneas a 100 usuários e o armazenamento máximo a 1 GB.

Para o reconhecimento dos objetos e categorias em uma imagem, utilizou-se a API Cloud Vision do Google. Este serviço também exige uma chave em sua utilização, que pode ser criada através da plataforma Google Cloud (GOOGLE, 2018c). O plano gratuito permite a execução de 1.000 operações de detecção de rótulos por mês, a partir deste volume há uma cobrança de uma taxa para cada milhar adicional.

Na sintetização de voz dos resultados da análise da imagem foi utilizada a classe `TextToSpeech` (TTS), disponibilizada de forma nativa pela SDK do Android. Esta API permite sintetização de um texto em comandos de voz para ser executada de forma imediata ou permite a criação de um arquivo de áudio. Os comandos de voz apresentados a cada etapa no aplicativo foram previamente gravados e são executados através da classe `MediaPlayer`, também disponibilizada pela SDK do Android.

3.3.2 Autenticação e cadastro de usuários

Ao utilizar o aplicativo pela primeira vez o usuário é direcionado para uma tela de autenticação e precisa realizar um cadastro simples, este processo de está implementado na classe `LoginActivity`. A primeira instrução executada pela classe `LoginActivity` é se conectar à plataforma Firebase, esta conexão é necessária para que o aplicativo obtenha a referência da instância do Firebase a ser utilizada nos métodos de autenticação e referência para a banco de dados que contém a base de usuários. A conexão com o Firebase é realizada através da classe `ConfiguracaoFirebase` e as credenciais para conexão do aplicativo com o Firebase são configuradas no arquivo `google-services.json`, presente no projeto.

Após estabelecer a conexão com o Firebase, o aplicativo permite que o usuário informe um e-mail e senha para se autenticar. Caso este e-mail não exista no cadastro de usuários, ele é cadastrado e o usuário é autenticado. Antes de validar os dados da autenticação no Firebase, o aplicativo efetua algumas pré-validações através do método `attemptLogin` para determinar se o usuário preencheu as informações de forma correta, como por exemplo, se preencheu o e-mail com o caractere "@" e se a senha possui pelo menos quatro dígitos. Caso as informações estejam corretas, a aplicação aciona o método `validarLogin` apresentado no Quadro 1 responsável validar se o usuário informado já está cadastrado ou se é um novo cadastro.

Quadro 1- Validar login do usuário

```
1  private void validarLogin(String mEmail, String mPassword) {
2     autenticacao = ConfiguracaoFirebase.getFirebaseAutenticacao();
3     autenticacao.signInWithEmailAndPassword(mEmail, mPassword)
4           .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
5              @Override
6              public void onComplete(@NonNull Task<AuthResult> task) {
7
8                 if (task.isSuccessful()) {
9                    usuarioAutenticado(); //abre a tela principal do app
10                } else {
11                   String erroExcessao = "";
12
13                   // lança a excessão retornada pelo firebase e trata a mensagem com base na
14 excessão
15                   try {
16                      throw task.getException(); // throw = lançar - no caso exceção
17
18                   } catch (FirebaseAuthInvalidUserException e) {
19                      // Se não existir a conta cadastrada, tenta cadastrar
20                      cadastrarUsuario(mEmail, mPassword);
21                      // erroExcessao = "E-mail não possui uma conta vinculada!";
22
```

```
23                  } catch (FirebaseAuthInvalidCredentialsException e) {
24                      erroExcessao = "Senha inválida para o e-mail informado!";
25                  } catch (Exception e) {
26                      erroExcessao = "ao efetuar o login.";
27                      e.printStackTrace();
28                  }
29
30                  if (!erroExcessao.equals("")) {
31                      showProgress(false);
32                      Toast.makeText(LoginActivity.this, "Erro: " + erroExcessao
33                              , Toast.LENGTH_LONG).show();
34                  }
35              }
36          }
37      });
38  }
```

Fonte: Elaborado pelo autor.

No método `validarLogin` a variável `autenticação` recebe a referência da instância do Firebase ao qual precisa se comunicar, na sequência tenta realizar a autenticação do usuário através da interface `signInWithEmailAndPassword` disponibilizada pela API do Firebase. O método `addOnCompleteListener` monitora o resultado da autenticação do usuário, caso a autenticação ocorra com sucesso (`task.isSuccessful`) a aplicação marca o usuário como autenticado e o direciona para a tela principal do aplicativo através do método `usuarioAutenticado`. O método possui comportamentos diferentes para os casos de falha na autenticação, caso o usuário não exista na base de dados (exceção `FirebaseAuthInvalidUserException`) o aplicativo aciona o método `cadastrarUsuario` para cadastrar o usuário, se a senha é inválida o aplicativo notifica que a senha é inválida e caso o Firebase retorne um erro não previsto, a descrição do erro é apresentada ao usuário.

O cadastro de novos usuários é criado através do método `cadastrarUsuario` apresentado no Quadro 2. Este método utiliza a classe `Usuario` para criar uma nova instância do objeto, preenche as informações do usuário (e-mail e senha) através dos métodos `usuario.setEmail` e `usuario.setSenha`. Realiza o cadastro do usuário no serviço de autenticação do Firebase através do método `createUserWithEmailAndPassword`, caso o usuário tenha sido registrado com sucesso o usuário é registrado no banco de dados presente no Firebase através do método `salvar` da classe `Usuario`.

Quadro 2 - Cadastrar novo usuário

```
1   private void cadastrarUsuario(String mEmail, String mPassword) {
2
3       Usuario usuario;
4       usuario = new Usuario();
```

```
5       usuario.setEmail(mEmail);
6       usuario.setSenha(mPassword);
7
8       // executa o cadastro do usuario e valida o resultado do cadastro
9       autenticacao.createUserWithEmailAndPassword(
10              usuario.getEmail(),
11              usuario.getSenha()
12      ).addOnCompleteListener(LoginActivity.this, new OnCompleteListener<AuthResult>() {
13          @Override
14          public void onComplete(@NonNull Task<AuthResult> task) {
15
16              if (task.isSuccessful()) {
17                  //Recuperar o id de usuario gerado pelo firebase
18                  FirebaseUser usuarioFirebase = task.getResult().getUser();
19                  usuario.setId(usuarioFirebase.getUid());
20                  usuario.salvar();
21
22                  Preferencias preferencias = new Preferencias(LoginActivity.this);
23                  preferencias.salvarUsuarioPreferencias(usuario.getEmail(), usuario.getId(),
24  usuario.getSenha());
25
26                  showProgress(false);
27                  // direciona para a tela principal do app
28                  usuarioAutenticado();
29              } else {
30
31                  showProgress(false);
32                  Toast.makeText(LoginActivity.this, "Erro ao cadastrar usuario: " +
33  task.getException().toString()
34                          , Toast.LENGTH_LONG).show();
35              }
36          }
37      });
38  }
```

Fonte: Elaborado pelo autor.

3.3.3 Reconhecimento de objetos e categorias nas imagens

O código fonte responsável pelo reconhecimento de objetos e categorias através da API Cloud Vision está localizado na classe `MainActivity` e foi criado utilizando como base o código fonte de exemplo disponibilizado pelo Google no Github (GOOGLE, 2018f). O aplicativo permite analisar uma imagem capturada através da câmera do dispositivo através do método `startCamera` ou através de uma imagem já existente no dispositivo através do método `startGalleryChooser`. O retorno de ambos os métodos contém a imagem a ser analisada e é tratado no método `onActivityResult`, este método aciona o método `uploadImage` listado no Quadro 3. Este método faz o redimensionamento da imagem para economizar banda através

do método `scaleBitmapDown`, chama o método `callCloudVision` que irá efetivamente acionar o processo de análise do conteúdo da imagem e exibe a imagem em análise na tela através do comando `mMainImage.setImageBitmap(bitmap)`.

Quadro 3 - Método uploadImage

```
1  public void uploadImage(Uri uri) {
2     if (uri != null) {
3        try {
4           // scale the image to save on bandwidth
5           Bitmap bitmap =
6                 scaleBitmapDown(
7                       MediaStore.Images.Media.getBitmap(getContentResolver(), uri),
8                       MAX_DIMENSION);
9
10          callCloudVision(bitmap);
11          mMainImage.setImageBitmap(bitmap);
12
13       } catch (IOException e) {
14          Log.d(TAG, "Image picking failed because " + e.getMessage());
15          Toast.makeText(this, R.string.image_picker_error, Toast.LENGTH_LONG).show();
16       }
17    } else {
18       Log.d(TAG, "Image picker gave us a null image.");
19       Toast.makeText(this, R.string.image_picker_error, Toast.LENGTH_LONG).show();
20    }
21 }
```

Fonte: Elaborado pelo autor.

O método `callCloudVision` listado no Quadro 4 é responsável por solicitar a análise do conteúdo da imagem. Ele altera o texto exibido na tela informando que a imagem está sendo processada, executa uma instrução por voz através do método `playInstruction` informando o início da análise e solicitando que o usuário aguarde. Na sequência executa o método `prepareAnnotationRequest` que configura o composição da requisição que será enviada à API do Cloud Vision e é listado no Quadro 5, a requisição para a API do Cloud Vision é efetuada com o parâmetro `LABEL_DETECTION`, que determina qual análise será aplicada a imagem, no caso deste trabalho a identificação de objetos e categorias. Por fim executa o método `LabelDetectionTask` para realizar a análise da imagem. As chamadas das APIs do Cloud Vision são efetuadas através da biblioteca `google-api-services-vision`, disponível para a plataforma Android, esta biblioteca simplifica as chamadas para as API do Cloud Vision.

Quadro 4 - Método callCloudVision

```
1  //Realiza a chamada da API para reconhecimento da imagem
2  private void callCloudVision(final Bitmap) {
3     // Altera o texto para processando
```

```
4     mImageDetails.setText(R.string.loading_message);
5
6     playInstruction(INSTRUCAO_ANALISE_EM_PROCESSAMENTO);
7
8     // Executa a tarefa de analise da imagem em background
9     try {
10       AsyncTask<Object, Void, String> labelDetectionTask = new LableDetectionTask(this,
11  prepareAnnotationRequest(bitmap));
12       labelDetectionTask.execute();
13    } catch (IOException e) {
14       Log.d(TAG, "failed to make API request because of other IOException " +
15             e.getMessage());
16    }
17 }
```

Fonte: Elaborado pelo autor.

Quadro 5 - Método prepareAnnotationRequest

```
1  private Vision.Images.Annotate prepareAnnotationRequest(Bitmap) throws IOException {
2     HttpTransport httpTransport = AndroidHttp.newCompatibleTransport();
3     JsonFactory jsonFactory = GsonFactory.getDefaultInstance();
4
5     VisionRequestInitializer requestInitializer =
6           new VisionRequestInitializer(CLOUD_VISION_API_KEY) {
7              @Override
8              protected void initializeVisionRequest(VisionRequest<?> visionRequest)
9                    throws IOException {
10                super.initializeVisionRequest(visionRequest);
11
12                String packageName = getPackageName();
13                visionRequest.getRequestHeaders().set(ANDROID_PACKAGE_HEADER,
14 packageName);
15
16                String sig = PackageManagerUtils.getSignature(getPackageManager(),
17 packageName);
18
19                visionRequest.getRequestHeaders().set(ANDROID  CERT  HEADER, sig);
20             }
21          };
22
23    Vision.Builder builder = new Vision.Builder(httpTransport, jsonFactory, null);
24    builder.setVisionRequestInitializer(requestInitializer);
25
26    Vision vision = builder.build();
27    BatchAnnotateImagesRequest batchAnnotateImagesRequest =
28          new BatchAnnotateImagesRequest();
29    batchAnnotateImagesRequest.setRequests(new ArrayList<AnnotateImageRequest>() {{
30       AnnotateImageRequest annotateImageRequest = new AnnotateImageRequest();
31
32       // Add the image
33       Image base64EncodedImage = new Image();
```

```
34          // Convert the bitmap to a JPEG
35          ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
36          bitmap.compress(Bitmap.CompressFormat.JPEG, 90, byteArrayOutputStream);
37          byte[] imageBytes = byteArrayOutputStream.toByteArray();
38
39          // Base64 encode the JPEG
40          base64EncodedImage.encodeContent(imageBytes);
41          annotateImageRequest.setImage(base64EncodedImage);
42
43          // add the features
44          annotateImageRequest.setFeatures(new ArrayList<Feature>() {{
45              Feature labelDetection = new Feature();
46              labelDetection.setType("LABEL_DETECTION");
47              labelDetection.setMaxResults(MAX_LABEL_RESULTS);
48              add(labelDetection);
49          }});
50
51          // Add the list of one thing to the request
52          add(annotateImageRequest);
53      }});
54
55      Vision.Images.Annotate annotateRequest =
56              vision.images().annotate(batchAnnotateImagesRequest);
57      // Due to a bug: requests to Vision API containing large images fail when GZipped.
58      annotateRequest.setDisableGZipContent(true);
59      Log.d(TAG, "created Cloud Vision request object, sending request");
60
61      return annotateRequest;
62  }
```

Fonte: Elaborado pelo autor.

A análise da imagem é efetuada de forma assíncrona através da classe `LabelDetectionTask` listada no Quadro 6. Ela é responsável por acionar a API do Cloud Vision e aguardar o retorno do resultado da análise, após receber o retorno o aplicativo aciona o método `convertResponseToStringAndSpeech` que recebe a resposta da API e a interpreta. Ao final o aplicativo exibe o texto dos objetos identificados através do método `onPostExecute`.

Quadro 6 - Classe LableDetectionTask

```
1   private static class LableDetectionTask extends AsyncTask<Object, Void, String> {
2       private final WeakReference<MainActivity> mActivityWeakReference;
3       private Vision.Images.Annotate mRequest;
4
5       LableDetectionTask(MainActivity activity, Vision.Images.Annotate annotate) {
6           mActivityWeakReference = new WeakReference<>(activity);
7           mRequest = annotate;
8       }
9
10      @Override
```

```
11      protected String doInBackground(Object... params) {
12          try {
13              Log.d(TAG, "created Cloud Vision request object, sending request");
14              // Realiza a chamada em background para a API do google
15              BatchAnnotateImagesResponse response = mRequest.execute();
16              return convertResponseToStringAndSpeech(response);
17
18          } catch (GoogleJsonResponseException e) {
19              Log.d(TAG, "failed to make API request because " + e.getContent());
20          } catch (IOException e) {
21              Log.d(TAG, "failed to make API request because of other IOException " +
22                      e.getMessage());
23          }
24          return "Cloud Vision API request failed. Check logs for details.";
25      }
26
27      protected void onPostExecute(String result) {
28          MainActivity activity = mActivityWeakReference.get();
29          if (activity != null && !activity.isFinishing()) {
30              TextView imageDetail = activity.findViewById(R.id.image_details);
31              imageDetail.setText(result);
32          }
33      }
34  }
```

Fonte: Elaborado pelo autor.

O método `convertResponseToStringAndSpeech` apresentado no Quadro 7 é responsável por interpretar o resultado da análise de imagem retornada pela API Cloud Vision. O aplicativo recebe como resultado a descrição em forma de texto dos objetos e categorias identificados na imagem, cada item do resultado também recebe uma pontuação de confiança indicando a probabilidade de o item descrito estar presente na imagem, a pontuação dos itens varia de 0 (nenhuma confiança) a 1 (muito alta). Estes dados são tratados e exibidos na tela ao usuário. Este método também realiza a sintetização do resultado da análise através de comandos de voz e para isto utiliza o método `speak` da biblioteca `TTS` (TextToSpeech) presente na SDK do Android.

Quadro 7 - Método convertResponseToStringAndSpeech

```
1   //conversão do resultado para texto e fala
2   private static String convertResponseToStringAndSpeech(BatchAnnotateImagesResponse
3   response) {
4       StringBuilder message = new StringBuilder("Itens encontrados na imagem:\n\n");
5
6       String resultadoAnalise;
7
8       List<EntityAnnotation> labels = response.getResponses().get(0).getLabelAnnotations();
9       if (labels != null) {
```

```
10
11          resultadoAnalise = "Items found in image: ";
12          tts.speak(resultadoAnalise, TextToSpeech.QUEUE_ADD, null, "TESTE");
13
14          for (EntityAnnotation label : labels) {
15             //Formata a string com os itens encontrados na imagem
16             message.append(String.format(Locale.US, "%.3f: %s", label.getScore(),
17    label.getDescription() ));
18             message.append("\n");
19
20             //Texto a ser sintetizado
21             tts.speak(label.getDescription(), TextToSpeech.QUEUE_ADD, null, "TESTE");
22
23          }
24       } else {
25          message.append("Nenhum objeto encontrado.");
26          resultadoAnalise = "No objects found.";
27          tts.speak(resultadoAnalise, TextToSpeech.QUEUE_ADD, null, "TESTE");
28       }
29       return message.toString();
30    }
```

Fonte: Elaborado pelo autor.

3.3.4 Instruções de uso do aplicativo

Na abertura e durante a utilização do aplicativo, o usuário recebe várias instruções de voz. Estas instruções são necessárias para situar o usuário portador de deficiência visual do local do aplicativo onde ele se encontra e da ação que está sendo realizada. Estas instruções foram previamente gravadas em um arquivo de áudio e são apresentadas ao usuário através do método `playInstruction` apresentado no Quadro 8. Este método é acionado em vários pontos da classe MainActivity e recebe como parâmetro qual instrução deve ser executada. O áudio é executado através da biblioteca `MediaPlayer` presente na SDK do Android.

Quadro 8 - Método playInstruction

```
1   private void playInstruction (int instruction ){
2
3      //Trata a instrução recebida e executada o arquivo correspondente (pré-gravado)
4      switch ( instruction ) {
5         case INSTRUCAO_INICIAL:
6            mediaPlayer = MediaPlayer.create(MainActivity.this,
7   R.raw.abertura_de_duplo_clique);
8            break;
9         case INSTRUCAO_CAMERA_ACIONADA:
10           mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.camera_ativada);
11           break;
12        case INSTRUCAO_ANALISE_EM_PROCESSAMENTO:
13           mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.analisando_imagem);
```

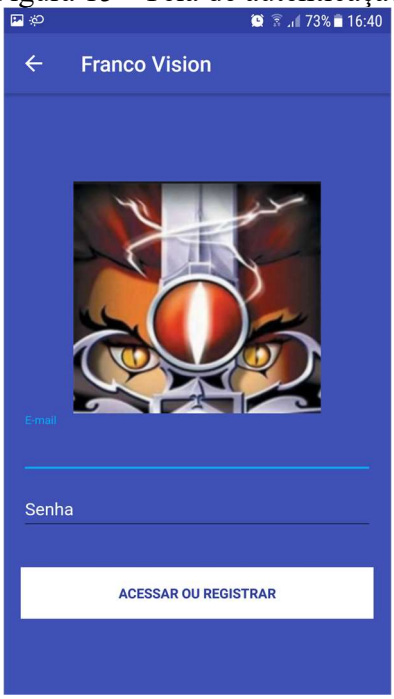
```
14              break;
15          case INSTRUCAO_ANALISE_ERRO:
16              mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.analise_erro);
17              break;
18          case INSTRUCAO_ANALISE_SUCESSO:
19              mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.analise_sucesso);
20              break;
21          case INSTRUCAO_CAPTURAR_NOVA_IMAGEM:
22              mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.nova_imagem);
23              break;
24          case INSTRUCAO_AGRADECIMENTOS:
25              mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.agradecimentos);
26              break;
27      }
28
29      //libera os recursos da memória após a execução
30      mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
31          @Override
32          public void onCompletion(MediaPlayer mp) {
33              mp.release();
34              mp = null;
35          }
36      });
37      //Executa o som
38      mediaPlayer.start();
39  }
```

Fonte: Elaborado pelo autor.

3.3.5 Operacionalidade da implementação

A Figura 15 apresenta a tela de autenticação, que é apresentada ao usuário na primeira utilização do aplicativo. Ao abrir a tela o usuário é instruído através de comandos de voz que está utilizando o aplicativo Franco Vision e recebe a orientação para que solicite ajuda para realizar o cadastro inicial. O cadastro do usuário é realizado através do seu endereço de e-mail e uma senha, caso o usuário já possua cadastro ele deve informar os dados para se autenticar. A etapa de autenticação precisa ser realizada apenas uma vez e nos próximos acessos o usuário não verá a tela de autenticação.

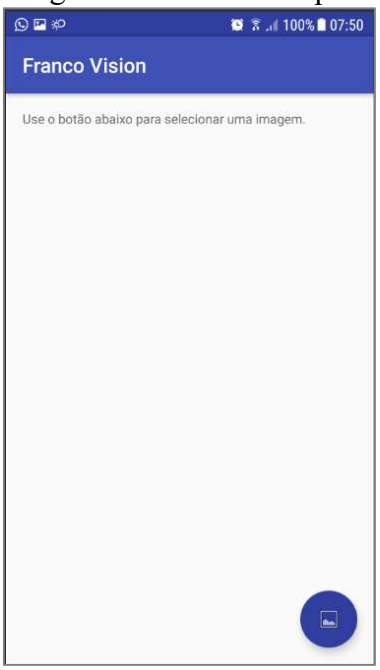
Figura 15 - Tela de autenticação



Fonte: Elaborado pelo autor.

Após a etapa de autenticação, o usuário é direcionado à tela principal do aplicativo demonstrada pela Figura 16. Nesta tela o usuário é orientado através de comandos de voz a dar duplo clique na tela para ativar a câmera e capturar uma foto.

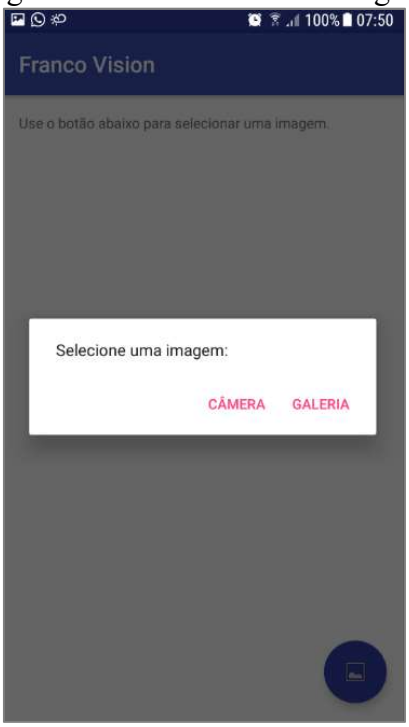
Figura 16 - Tela Principal



Fonte: Elaborado pelo autor.

Nesta tela o usuário também tem opção para a ativar a câmera de forma manual ou selecionar uma foto da galeria. Estas opções são acessíveis através do botão flutuante da tela demonstrado na Figura 17.

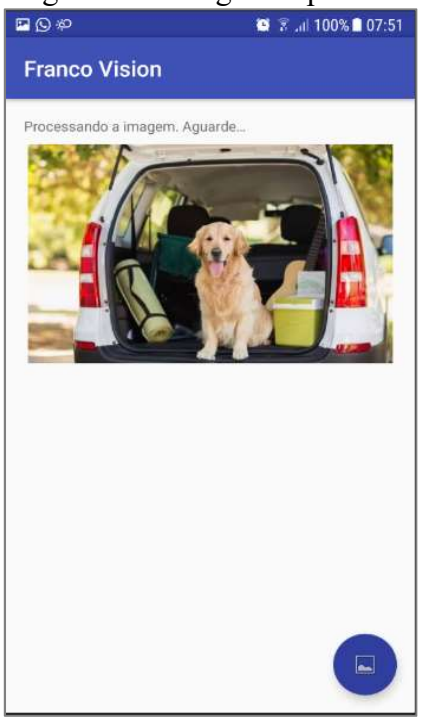
Figura 17 - Selecione uma imagem



Fonte: Elaborado pelo autor.

Após a captura de uma foto ou da seleção de uma imagem na galeria, a imagem é exibida no centro da tela e o aplicativo informa através de comandos de voz para que o usuário aguarde o processo de análise da imagem. Esta etapa é exibida a seguir na Figura 18.

Figura 18 - Imagem capturada



Fonte: Elaborado pelo autor.

Ao finalizar a análise da imagem o aplicativo descreve para o usuário, através da sintetização de voz, os objetos e categorias identificadas. O aplicativo também apresenta em

forma de texto os itens identificados, detalhando a pontuação de confiança obtida cada item. A pontuação pode variar de 0 (nenhuma confiança) a 1 (muito alta) e o resultado de uma análise pode ser visto na Figura 19.

Figura 19 - Resultado da análise

Fonte: Elaborado pelo autor.

A Figura 20 apresenta em destaque os itens identificados na análise da imagem e a sua pontuação. A sequência para exibição dos itens identificados respeita a pontuação recebida por cada item, sendo que os itens com maior pontuação são listados primeiro.

Figura 20 - Pontuação dos elementos

Fonte: Elaborado pelo autor.

Após receber o resultado da análise, o usuário pode efetuar a análise de novas imagens acionando a câmera através do duplo clique ou do botão flutuante.

3.4 RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os testes realizados com o aplicativo, também é apresentada uma comparação deste trabalho com os trabalhos correlatos. Os testes do aplicativo foram realizados na plataforma Android utilizando o dispositivo Samsung A5 2016.

3.4.1 Testes da análise de imagens

Para avaliar a eficiência do reconhecimento de objetos e categorias em imagens, o aplicativo foi submetido a testes com 30 imagens que fazem parte do cotidiano de uma pessoa. Nos testes foram considerados imagens com um único objeto assim como cenas com paisagens. O Quadro 9 apresenta a descrição das imagens capturadas, o resultado obtido para cada teste, a avaliação dada para o teste classificando o resultado como relevante ou não, por fim apresenta o total de itens relevantes para o teste, dado que o resultado da análise pode conter várias classificações da imagem.

Para a descrição do resultado da análise foram considerados apenas os itens com pontuação igual ou superior a 0.7, ou seja, apenas os itens classificados com uma probabilidade de estarem contidos na imagem de pelo menos 70%. Os itens considerados como relevantes, foram destacados em negrito na coluna que contém o resultado da análise. Na execução dos

testes, foram notados que alguns fatores podem reduzir a taxa de acerto da análise, como por exemplo a distância do objeto capturado e imagens desfocadas. As imagens do aplicativo contendo o resultado de cada teste realizado estão disponíveis para consulta no Google Drive (FRANCO, 2018).

Para considerar se o resultado da análise do conteúdo da imagem é relevante, utilizou-se como critério o auxílio dado ao usuário em reconhecer o objeto e os detalhes da sua composição. O objeto não precisa ser exatamente descrito para que o resultado seja considerado relevante. Caso uma das classificações com pontuação maior do que 0.7 aplicadas à imagem transmita ao usuário algo que possa identificar o objeto ou a cena analisada, o resultado é considerado como relevante.

Quadro 9 - Resultado reconhecimento de imagens

Conteúdo da imagem	**Resultado da análise**	**Resultado relevante**	**Itens relevantes**
Vaso de flor amarela	**Yellow**, **flower**, **cut flowers**, floristry, **flower arranging, petal,**	Sim	5/6
Extintor de incêndio	Product, Product	Não	0/2
Motocicleta com um baú de carga	Product, **motor vehicle**, **bicycle accessory**, **bicycle**, **vehicle**, **wheel**	Sim	6/7
Carro em uma garagem	**Motor vehicle, vehicle, transport, mode of transport, car, automotive exterior, automotive tire, automotive** wheel system, automotive design, **tire**	Sim	9/11
Rua contendo casas, vegetação rasteira e árvores ao fundo	**Tree, sky, building, plant, facade**	Sim	5/5
Vaso com uma planta	**Plant, leaf, flowerpot,** flora, **houseplant**	Sim	4/5
Relógio de pulso com pulseira marrom	**Watch accessory, buckle, strap,** belt**, watch strap, brown**, belt buckle, Product, **watch**	Sim	6/9
Armário contendo um forno elétrico e um forno de micro-ondas, ambos brancos	**Home appliance, kitchen appliance,** clothes dryer, major appliance, **eletronics, microwave oven**	Sim	5/7
Xícara branca com textos escritos	Product	Não	0/1
Xícara branca com textos escritos e um prato branco	Nenhum item com pontuação acima de 0,7	Não	0/0
Garrafa térmica vermelha	Red	Não	0/1
Par de tênis de cor predominante branca e detalhes em vermelho e preto	**Footwear, White, shoe, sneakers**	Sim	4/4

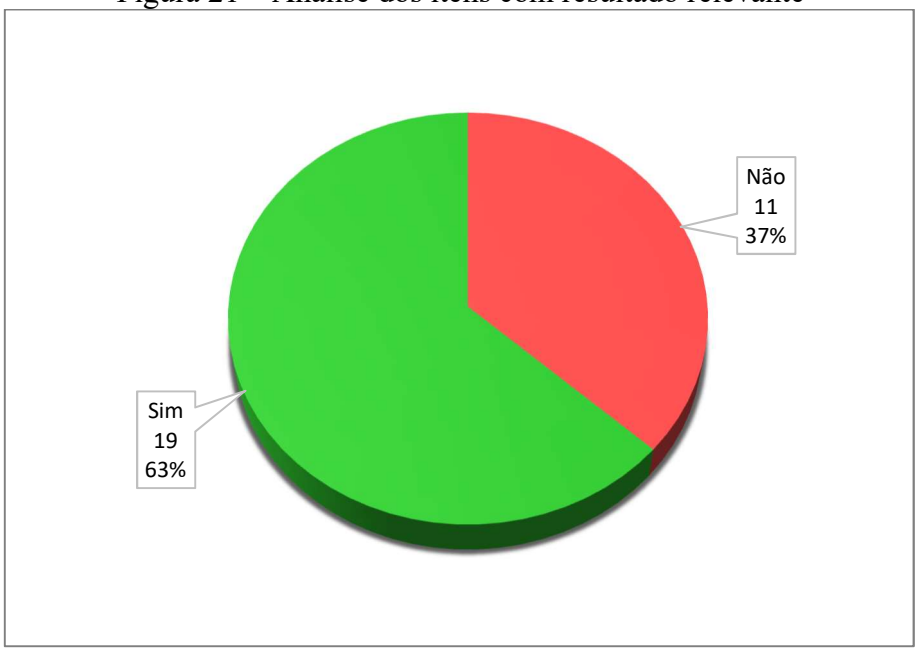
Ovo sobre a mesa	Orange	Não	0/1
Maçã vermelha (cerca 80 cm de distância da câmera)	Nenhum item com pontuação acima de 0,7	Não	0/0
Maçã vermelha (cerca 10 cm de distância da câmera)	**Fruit**, **apple**, produce	Sim	2/3
Limão Taiti (cerca 80 cm de distância da câmera)	Green	Não	0/1
Limão Taiti (cerca 10 cm de distância da câmera)	**Lime**, produce, **persian lime, fruit, key lime, citrus, citron**	Sim	6/7
Calculadora cientifica Casio	**Eletronics, technology, eletronic device**	Sim	3/3
Dois vasos decorativos com um fundo de madeira	Nenhum item com pontuação acima de 0,7	Não	0/0
Bancada com um televisor LCD, um controle remoto e um modem	**Screen, tecnology, display device, television, computer monitor, lcd tv, television set, led backlit display,** multimídia**, eletronic device**	Sim	9/10
Cadeira de madeira de cor clara, com estofamento floral	**Furniture**, shelf, shelving, table	Sim	1/4
Mala de viagem preta com a alça levantada	**Product**	Não	0/1
Par de brincos dourando com detalhe em branco	**Body jewelry, jewellery, silver, metal**	Sim	4/4
Porta de madeira na cor marrom com detalhes (trabalhada)	Furniture, wood stain	Não	0/2
Chaveiro na cor prata, com quatro chaves também na cor prata	Nenhum item com pontuação acima de 0,7	Não	0/0
Bandeira do Brasil (através da galeria de imagens)	**Green**, font, **flag**	Sim	2/3
Imagem do centro de Blumenau, contendo carros, pessoas e um ponto turístico	**Car, town, landmark, city, sky, street, downtown**	Sim	7/7
Fechada de um prédio contendo outros prédios ao redor e árvores	**Metropolitan area, tower block, skyscraper, condominium, urban area, building, city, metropolis, residential área,** mixed use	Sim	9/10

Caminhão	**Transport, motor vehicle, vehicle, truck, comercial vehicle, mode of transport, trailer truck, freight transport**	Sim	8/8
Vista de uma praia contendo uma palmeira, o mar e o céu de fundo	Tropics**, sky**, caribbean**, sea, vegetation, shore, palm tree, arecales, coastal and oceanic landforms, beach**	Sim	8/10

Fonte: Elaborado pelo autor.

Conforme demonstrado na Figura 21, do total de trinta testes realizados, dezenove apresentaram resultado considerado relevante, o equivalente a 67% dos testes. Enquanto que os itens com resultado considerado irrelevante totalizam 11 testes, o equivalente a 33% dos cenários.

Figura 21 – Análise dos itens com resultado relevante



Fonte: Elaborado pelo autor.

3.4.2 Testes de usabilidade

Os testes de usabilidade foram aplicados em dois usuários com idades entre 25 e 30 anos com o intuito validar a utilização do aplicativo nas tarefas do cotidiano. Não foi possível aplicar o teste do aplicativo em pessoas com deficiência visual, por isso os usuários receberam a instrução para utilizar o aplicativo de olhos vendados e o objetivo de reconhecer objetos ao seu redor.

Em relação a interface da aplicação, o aplicativo recebeu elogios dos usuários por apresentar uma interface simples e de fácil uso. As instruções por comando de voz orientam o usuário sobre a sua localização dentro do aplicativo e comunicam o que está ocorrendo, os usuários informaram que esta funcionalidade é essencial para a utilização do aplicativo. O

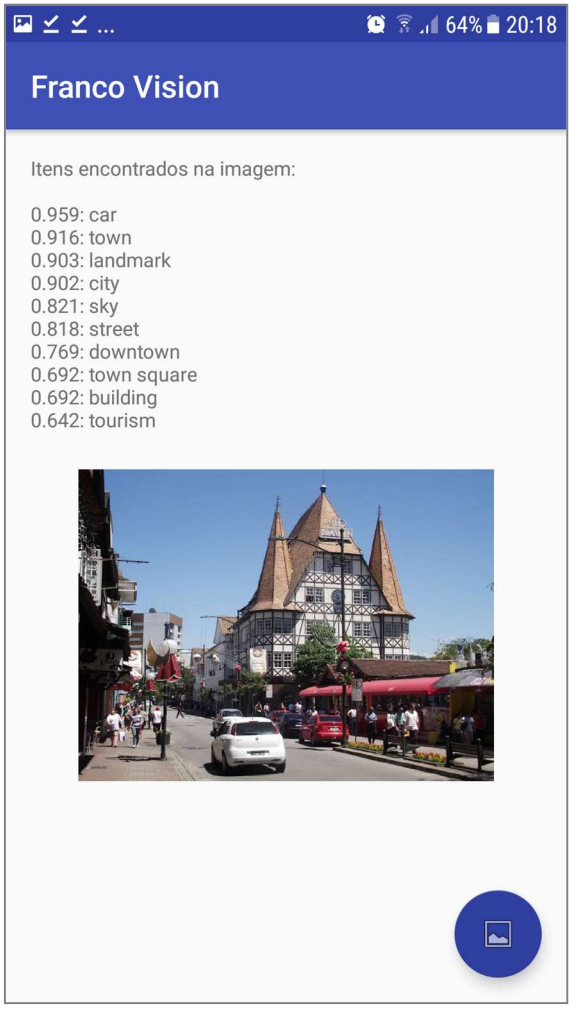
acionamento da câmera através do duplo clique também foi elogiado, pois facilita o acesso a função, visto que um usuário com deficiência visual teria dificuldade de acessar a câmera através de um botão.

Os usuários reportaram dúvidas sobre como capturar a foto após acionar a câmera. Este ponto do aplicativo apresenta comportamentos diferentes conforme o modelo do dispositivo pois a imagem é capturada através do aplicativo de câmera padrão do aparelho. Em alguns aparelhos a foto pode ser capturada dando duplo clique em qualquer lugar da tela, em outros dispositivos é preciso clicar em um botão. No dispositivo Samsung A5 2016, foi configurado o aplicativo de câmera do celular para capturar a foto através do comando de voz `capturar`. Após a explicação deste ponto, os usuários conseguiram capturar a foto normalmente.

Após a captura da foto, o aplicativo solicita a confirmação da imagem antes de realizar a análise do seu conteúdo. Esta confirmação não é adequada ao público alvo do aplicativo, pois o botão de confirmação fica na parte superior da tela, sendo de difícil acesso a pessoas com deficiência visual. Este é um comportamento padrão da biblioteca da câmera disponibilizada pelo SDK do Android e não pode ser removido. Ao estudar opções de contorno para esta questão, identifiquei que a única solução disponível é implementar dentro do aplicativo a funcionalidade de câmera ao invés de utilizar a biblioteca de câmera disponibilizada pelo Android. Devido ao cronograma restrito, a melhoria da câmera deverá ser implementada em uma segunda etapa e foi listada como uma extensão do trabalho atual.

Os usuários reportaram que a qualidade do reconhecimento dos objetos é satisfatória, sendo que a aplicação reconheceu corretamente a maioria das imagens capturadas. Como o resultado da análise retorna objetos e categorias, um usuário reportou que em alguns casos o resultado fica genérico, como por exemplo ao capturar um vaso o resultado da análise retornou apenas a categoria produto. Já para os casos de imagens com paisagens a apresentação de categorias foi citada como relevante pois ajuda a descrever o contexto, o usuário citou como exemplo a análise de uma foto do centro de Blumenau demonstrada na Figura 22, onde o aplicativo descreveu a existência de uma cidade, um carro, um ponto turístico e uma rua.

Figura 22 - Análise da foto do centro de Blumenau



Fonte: Elaborado pelo autor.

Os usuários comentaram que o resultado da análise da imagem deveria ser traduzido para português, esta melhoria irá facilitar a utilização e adoção do aplicativo por mais usuários. De forma geral, os usuários aprovaram a aplicação e informaram que ela tem relevância e potencial para ajudar as pessoas com deficiência visual a superar algumas dificuldades do seu dia a dia.

3.4.3 Comparação com os trabalhos correlatos

Esta seção realiza uma comparação do aplicativo desenvolvido com os trabalhos correlatos descritos na seção 2.4. O Quadro 10 apresenta os trabalhos correlatos, o domínio ao qual cada trabalho se dedica, efetua a comparação entre as principais características de cada um, identifica os trabalhos que são tecnologias assistivas voltadas para a inclusão social e apresenta as plataformas em que cada trabalho está disponível.

Quadro 10 - Comparativo entre trabalhos correlatos

Características / trabalhos relacionados	Kramer (2017)	Heinzle (2017)	Seefeld (2016)	Franco (2018)
Domínio	Reconhecimento dos textos em cartas de jogo para deficientes visuais	Reconhecimento de alimentos e sugestão de receitas	Reconhecimento de expressões faciais	Reconhecimento de objetos para deficientes visuais
Reconhecimento de imagem	Sim	Sim	Sim	Sim
Reconhecimento de texto	Sim	Não	Não	Não
Sintetização de voz	Sim	Não	Não	Sim
Utiliza Internet	Sim	Sim	Não	Sim
Tecnologia assistiva	Sim	Não	Não	Sim
Plataforma suportada	Android/IOS	Android/IOS	Windows (Desktop)	Android

Fonte: Elaborado pelo autor.

Através do quadro, é possível observar que todos os projetos possuem como principal característica o reconhecimento de informações em imagens e o retorno dos dados obtidos ao usuário. O projeto de Kramer (2017) realiza a identificação dos textos presentes em uma carta de jogo (destinado ao jogo Munchkin), o projeto de Heinzle (2017) é destinado ao reconhecimento de alimentos, o projeto de Seefeld (2016) realiza o reconhecimento de expressões faciais e o Franco Vision realiza a identificação de objetos e categorias em imagens. Somente o trabalho de Kramer (2017) e o Franco Vision podem ser classificados como uma tecnologia assistiva, sendo destinados às pessoas portadoras de deficiência visual. Para permitir a acessibilidade destas pessoas o trabalho de Kramer (2017) utiliza a sintetização de voz para detalhar os comandos da aplicação, como também para apresentar a descrição do conteúdo da carta do jogo ao usuário. Enquanto que o Franco Vision utiliza a sintetização de voz para detalhar o conteúdo de uma imagem.

As soluções analisadas utilizam a conexão com a internet para acessar os serviços de análise das imagens, com exceção do trabalho de Seefeld (2016) que realiza a análise da imagem de forma local através da biblioteca Dlib (DLIB, 2016). O protótipo desenvolvido por Seefeld (2016) é uma aplicação desktop para a plataforma Windows e realiza a análise de imagens já existentes, os projetos de Kramer (2017) e Heinzle (2017) foram desenvolvidos de forma híbrida para plataformas móveis destinadas aos ambientes Android e IOS, enquanto que o Franco Vision é exclusivo para dispositivos móveis com a plataforma Android.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um aplicativo móvel utilizando a linguagem de programação Java para a plataforma Android com o intuito de auxiliar no cotidiano das pessoas com deficiência visual efetuando a análise do conteúdo de imagens através da API Cloud Vision do Google. Com este aplicativo o usuário é capaz de capturar uma imagem da paisagem a sua frente e ouvir através da sintetização de comandos de voz os objetos e categorias identificadas na imagem. Os resultados alcançados foram satisfatórios, dado que 67% dos testes realizados foram considerados com resultado relevante e os usuários que testaram a aplicação relataram que a proposta do aplicativo possui aplicação ao cotidiano, tendo assim potencial para auxiliar as pessoas com deficiência visual.

O objetivo principal do projeto, que era o de criar um aplicativo que auxilie as pessoas com deficiência visual a identificar objetos em uma imagem, foi atingido, entretanto não foi possível validar a utilização da aplicação com deficientes visuais, os testes foram realizados em dois usuários com os olhos vendados. Quanto aos objetivos específicos, que eram permitir a captura de uma imagem através da câmera do celular, reconhecer os objetos presentes na imagem, apresentar o conteúdo da análise ao usuário e dispor de uma interface de utilização adequada às necessidades dos deficientes visuais, também foram atingidos. A aplicação desenvolvida permite a captura de uma imagem através da câmera do celular, analisa o conteúdo da imagem através da API Cloud Vision, apresenta o resultado da análise através da sintetização de voz, descreve por voz cada etapa executada no aplicativo e permite que o usuário acione a câmera do dispositivo através do gesto de duplo clique na tela.

Foram identificadas algumas situações que podem influenciar no resultado do conteúdo da imagem reduzindo a sua acuracidade, como por exemplo a distância do objeto analisado em relação à câmera. Nos testes realizados com uma maçã a uma distância de 80 cm da câmera do celular a API não reconheceu o objeto. Em um segundo teste com a mesma maçã a uma distância de 10 cm da câmera o objeto foi reconhecido com perfeição. Outros fatores que podem influenciar o resultado da análise são a iluminação do ambiente e a nitidez da imagem capturada.

Para que o aplicativo seja utilizado no dia a dia por pessoas com deficiência visual são necessários alguns ajustes de usabilidade, como por exemplo a remoção da etapa de confirmação após a captura de uma imagem e a tradução do resultado da análise para português. Como limitação destaco o fato de a API Cloud Vision ser paga, o que implicaria na cobrança de uma mensalidade dos usuários caso o aplicativo seja utilizado em larga escala.

Por fim, este trabalho deixa uma contribuição social, ao se preocupar e sugerir uma solução aos portadores de deficiência visual com o intuito de identificar objetos e categorias em uma imagem. Quanto à contribuição cientifica, este trabalho apresenta um estudo de caso para a aplicação da API Cloud Vision no reconhecimento de objetos em imagens, além da fundamentação teórica apurada sobre os temas relacionados ao trabalho.

4.1 EXTENSÕES

Para trabalhos futuros, sugere-se as seguintes extensões:

a) melhorar a utilização da câmera do celular, removendo a confirmação da imagem capturada;
b) tradução do resultado da análise da imagem para português, facilitando assim o entendimento pelo usuário;
c) evolução do modelo de análise da imagem para permitir que o usuário possa determinar o tipo de análise a ser executada conforme o contexto. Análise de objetos para casos em que o usuário está focando um objeto em específico e análise de objetos mais categorias para os casos de análise de uma paisagem;
d) criação de uma API REST própria que permita realizar a análise da imagem, tornando a solução mais acessível devido a redução do custo da análise;
e) adição de outras funcionalidades voltadas para a acessibilidade, como informar a localização atual do usuário, ler o conteúdo de placas ou identificar cores.

REFERÊNCIAS

BRASIL. Lei nº 13146, de 6 de julho de 2015. **Lei Brasileira de Inclusão da Pessoa Com Deficiência**. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm>. Acesso em: 02 abr. 2018

CARAMORI, Marco Aurélio. **Protótipo de software para leitura labial**. 2012. 67 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2012.

CONCI, Aura; AZEVEDO, Eduardo; LETA, Fabiana R. **Computação Gráfica**. Rio de Janeiro: Elsevier, 2008.

FRANCO, Jean Carlos Franco. **Franco Vision**, [S.l.], 2018. Disponível em: <https://drive.google.com/open?id=10WlfN3qaPy8BQVBRCkb84OqGj19S5ua9>. Acesso em 28 nov.2018.

GOLL, Luís Henrique. **Viki:** inteligência virtual com interação por voz aplicada à internet das coisas. 2016. 59 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2016.

GOOGLE LLC. **Cloud text-to-speech.** [S.l.], 2018a. Disponível em: <https://cloud.google.com/>. Acesso em 04 jun. 2018.

_____. **Cloud text-to-speech.** [S.l.], 2018b. Disponível em: <https://cloud.google.com/text-to-speech>. Acesso em 04 jun. 2018.

_____. **Google Cloud.** [S.l.], 2018c. Disponível em: <https://cloud.google.com>. Acesso em 10 nov. 2018.

_____. **Google Vision.** [S.l.], 2018d. Disponível em: < https://cloud.google.com/vision>. Acesso em 15 nov. 2018.

_____. **Firebase.** [S.l.], 2018e. Disponível em: < https://firebase.google.com>. Acesso em 17 nov. 2018.

_____. **Google Cloud Sample Code.** [S.l.], 2018f. Disponível em: <https://github.com/GoogleCloudPlatform/cloud-vision>. Acesso em 25 ago. 2018.

GOULART, Gielez Feldhaus. **Aplicativo para auxiliar crianças autistas no desenvolvimento e aquisição da linguagem**. 2016. 50 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2016.

HASHIMOTO, Kei et al. An analysis of machine translation and speech synthesis in speech-to-speech translation system. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP)*,,* 2011, Prague. **Proceedings...** Prague: IEEE, 2011. p. 5108-5111.

HEINZLE, Rodrigo. **Cogncook: aplicativo para reconhecimento de imagens de alimentos e sugestão de receitas culinárias utilizando ingredientes disponíveis.** 2017. 50 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2017.

HENSEL, André Luís. **Analisador de imagens de alvos de competições para a plataforma Android**. 2012. 69 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2012.

HOSSEINI, Hossein; XIAO, Baicein; POOVENDRAN, Radha. Google's Cloud Vision API Is Not Robust To Noise**.** In: IEEE INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS, 16, 2017, Cancun. **Proceedings...** Cancun: IEEE*,* 2017. p. 101-105

IBGE. **Censo demográfico 2010**. [S.1], 2018. Disponível em: <ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/xls/Brasil_xls.zip>. Acesso em 05 nov. 2018.

KRAEMER, Ronan G. **Tecnologia assistiva: tornando jogo de mesa acessível para cegos com auxílio de aplicativo móvel de reconhecimento de imagem.** 2017. 63 f. TCC (Graduação) - Curso de Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2017.

RAHIM, Mazin G. **Artificial neural networks for speech analysis/synthesis**. London: Chapman e Hall, 1994.

SEEFELD, William L. **E-motiv:** protótipo de sistema para reconhecimento de expressões faciais. 2016. 59 f. TCC (Graduação) - Curso de Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2016.

STORA, Ana Paula et al. **Base de dados em tecnologias assistivas para pessoas com deficiência visual badatec.** Curitiba: Appris, 2016.

VANZIN, Tarcisio. Prefácio. In: STORA, Ana Paula et al. **Base de dados em tecnologias assistivas para pessoas com deficiência visual badatec.** Curitiba: Appris, 2016.