UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES - CAMPUS DE ERECHIM
DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JOÃO VITOR VERONESE VIEIRA

DEAFEND: APLICATIVO PARA FACILITAÇÃO DA COMUNICAÇÃO COM LIBRAS

ERECHIM - RS
2020

JOÃO VITOR VERONESE VIEIRA

DEAFEND: APLICATIVO PARA FACILITAÇÃO DA COMUNICAÇÃO COM LIBRAS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel, Departamento de Engenharias e Ciência da Computação da Universidade Regional Integrada do Alto Uruguai e das Missões - Campus de Erechim.

Orientador: Prof. Dr. Guilherme Afonso Madalozzo

ERECHIM - RS
2020

Dedico este trabalho à minha mãe, Claire; meu pai, João Ari; e meu irmão, Lucas Emanuel, por me permitirem compartilhar essa jornada chamada “vida” com vocês e construírem o melhor ambiente que alguém poderia ter para viver.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus que, em Sua infinita bondade, esteve sempre me iluminando durante todas as aulas, tarefas e provas do curso, fornecendo força e inspiração para continuar. Agradeço pela Sua proteção em todos os caminhos desta vida e por me engrandecer com tantas bençãos ao longo da mesma, ofertando todas as condições necessárias (e muito mais) para chegar onde eu cheguei e conseguir concluir este bacharel.

Em seguida, meus mais sinceros (e inúteis, caso comparados à relevância que possuem nesse feito) agradecimentos às pessoas mais importantes da minha existência que, mais do que partilhar as pequenas conquistas, aguentaram todos os momentos de frustração que, naturalmente, fazem parte de uma jornada de 5 anos e, com toda paciência e alegria, me colocavam novamente nos "trilhos" e continuavam ouvindo-me falar sobre os códigos que não funcionavam. Obrigado pai, mãe e Lucas por serem meus exemplos na vida; me ensinarem o conceito de lealdade desde berço; demonstrarem, em todas situações, o significado das palavras *caráter* e *humildade*; e fazerem muito mais do que estava ao alcance de vocês, apenas para que eu me sentisse bem ou nunca me faltasse absolutamente nada. Amo vocês.

Em seguida, gostaria de agradecer aos meus colegas da Turma 2016 por cada momento que passamos juntos. Não há nada melhor, em meio a uma batalha, do que olhar em volta e notar que há bons soldados ao seu lado. Agradeço por tudo que ensinaram-me e o quanto me permitiram evoluir neste período. Destaco aqui o Kelwin Komka e o Vinicius Emanoel Andrade - companheiros de grupo em 99,9% dos trabalhos - por, muito mais do que dividirem comigo a carga dos exercícios realizados, fazerem este processo tornar-se interessante. Espero que a parceria construída com todos mantenha-se firme e apenas cresça daqui para frente.

Gostaria de dizer um amplo "muito obrigado" também aos professores que, com sua paciência e sabedoria, transmitiram-nos o conhecimento da melhor forma possível. É notável e digno de reconhecimento o esforço de cada um - cada qual com seu comprometimento - em fazer-nos absorver o conhecimento passado à cada aula. Graças aos vossos direcionamentos, caminhos foram vislumbrados e hoje tenho a oportunidade de, mais do que seguir carreira solo, saber que muitos tornaram-se verdadeiros amigos. Saliento a gratidão à colaboração do meu orientador, Prof. Dr. Guilherme, que me apoiou durante o período deste trabalho. Agrego aqui o recado a todos os funcionários da Universidade Regional Integrada do Alto Uruguai e das Missões que, quando possível, facilitaram nossa passagem pela instituição.

Por fim, sou grato a todos aqueles que, de uma forma ou de outra, contribuíram para a realização deste trabalho. Todos, cada qual da sua forma e com sua intensidade, foram peças fundamentais para este resultado e, sem suas contribuições, com certeza uma parte estaria faltando. Encerro este ciclo hoje com a sensação de dever cumprido e muito satisfeito, em todos aspectos. Obrigado!

Eu acredito demais na sorte. E tenho constatado que, quanto mais duro eu trabalho, mais sorte eu tenho.

(Thomas Jefferson)

RESUMO

Para aumentar a qualidade de vida dos membros de uma sociedade é, indiscutivelmente, necessária uma comunicação eficiente entre tais indivíduos. Desta forma, as recentes evoluções tecnológicas são bastante positivas, pois possibilitaram ainda mais rapidez na troca de dados entre as pessoas, transformando o aparelho celular em uma das mais importantes ferramentas contemporâneas. Contudo, enquanto os ouvintes - maioria absoluta da população - obtiveram excelentes novas formas de contato, a comunidade surda ainda enfrenta diversas barreiras para expressar suas ideias e assimilar novas informações. Interessante observar que, apesar das conquistas alcançadas ao longo da história, que serviram para amenizar os impactos estruturais na vida desses sujeitos, as maiores dificuldades dos surdos têm como causa um aspecto em comum que, habitualmente, deveria ser motivo de aproximação: a conversação. Fica claro que não existe um mecanismo legítimo de conexão entre a parcela significativa da população que possui algum tipo de problema auricular e os demais, que geralmente não tem familiaridade com língua de sinais. Logo, este trabalho criou, através de técnicas de processamento de linguagem natural (PLN), um aplicativo com o intuito de expandir a comunicabilidade dos surdos e servir como alternativa para um diálogo baseado na paridade entre Português e Libras. Além da aplicação móvel, que contém um manual de usuário disponível *online* e funcionalidades capazes de colaborar para que seus utilizadores (inclusive analfabetos) usufruam dos benefícios provenientes de uma integração social concreta, este projeto produziu um tutorial introdutório ao PLN, buscando fomentar o contato inicial de entusiastas com esta área da inteligência artificial.

Palavras-chave: Comunicação. Surdez. Processamento de Linguagem Natural. Aplicativo.

ABSTRACT

To increase the quality of life for the members of society, efficient communication between such individuals is, undoubtedly, necessary. With this in mind, the recent technological developments are quite positive, as they enable the exchange of data between people to become even faster, transforming the cell phone into one of the most important contemporary tools. However, while listeners - an absolute majority of the population - have obtained excellent new ways of contact, the deaf community still faces several barriers to express their ideas and assimilate new information. It is interesting to note that, despite the achievements attained throughout history, which served to soften the structural impacts on the lives of these individuals, the greatest difficulties of the deaf are caused by a common aspect that, usually, should be a reason for approximation: conversation. There is no legitimate connection mechanism between the significant portion of the population that has some type of auricular problem and the others, who are generally unfamiliar with sign language. Therefore, this work created, through natural language processing (NLP) techniques, an application with the aim of expands the communicability of the deaf and serves as an alternative for a dialogue based on parity between Portuguese and Libras (Brazilian Sign Language). In addition to the mobile application, which contains a user manual available online and features capable of collaborates so that its users (including illiterates) enjoy the benefits from a concrete social integration, this project produced an introductory tutorial to NLP, seeking to foster initial contact of enthusiasts with this area of artificial intelligence.

Keywords: Communication. Deafness. Natural Language Processing. App.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
ASL	American Sign Language
ASR	Automatic Speech Recognition
AWS	Amazon Web Services
BSL	British Sign Language
CPU	Central Process Unit
DL	Deep Learning
EAD	Ensino à Distância
ENEM	Exame Nacional do Ensino Médio
ER	Entidade-Relacionamento
EUA	Estados Unidos da América
FENEIS	Federação Nacional de Educação e Integração de Surdos
FSL	French Sign Language
GIF	Graphics Interchange Format
GT	Google Tradutor
IA	Inteligência Artificial
IDE	Integrated Development Environment
IoT	Internet of Things
INES	Instituto Nacional de Educação de Surdos
JSON	JavaScript Object Notation
LBI	Lei Brasileira de Inclusão
LIBRAS	Língua Brasileira de Sinais
LSB	Língua de Sinais Brasileira

LSCB	Língua de Sinais dos Centros Urbanos do Brasil
ML	Machine Learning
NLP	Natural Language Processing
OMG	Object Management Group
PC	Personal Computer
PLN	Processamento de Linguagem Natural
POC	Proof of Concept
POS	Part of Speech
RLE	Run-length encoding
SDK	Software Development Kit
SO	Sistema Operacional
SQL	Structured Query Language
SST	Speech to Text
TDD	Telecommunications Device For The Deaf
TI	Tecnologia da Informação
TTS	Text to Speech
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
UML	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

O poeta Vinicius de Moraes, em sua canção *Samba da Bênção*, diz que "A vida é a arte do encontro, embora haja tanto desencontro pela vida" e o ser humano, dentro de seus instintos naturais e evoluções temporais, faz dessa afirmação o resumo de sua existência em grupo. Em uma era marcada pela abundância dos meios de comunicação - tais como o rádio, a televisão, os computadores e os celulares - e pela valorização do conhecimento, é inevitável que as pessoas passem a interagir mais entre si, mesmo que indiretamente. E a verdade é que todos sentem a necessidade de partilhar suas concepções, bem como de ser compreendido pelos seus semelhantes. Foi em razão de tal carência que a espécia humana conquistou um dos principais fatores de sua vantagem evolutiva: a comunicação (MUNDO-E, 2018).

O conceito de comunicação origina-se do latim *communicare* e transmite a ideia de tornar algo comum, compartilhar, trocar opiniões. Desta forma, o ato de comunicar resulta na troca de mensagens que, por sua vez, envolve emissão e recebimento de informações. Comunicação é o fomento de significados comuns entre comunicador e intérprete, através do uso de signos e símbolos (PINHEIRO, 2005). Talvez por isso, nunca tenha sido tão importante para um profissional, por exemplo, evidenciar uma capacidade de expressão clara e satisfatória. Essa tese aplica-se para todos os contextos da sociedade, haja visto que a interação é a base de qualquer ofício. Logo, quanto mais eficiente for esta interlocução, maior serão os ganhos obtidos deste processo. Assim, a busca pelo máximo rendimento tem acelerado a inclusão gradual de soluções tecnológicas na rotina dos cidadãos.

À vista disso, as novas possibilidades mostram-se determinantes, já que é notória a quantidade de mídia - textos, imagens, vídeos e áudios - que é produzida, diariamente, através dos milhões de dispositivos que são utilizados para ampliar as formas de obter e compartilhar dados. Como resultado, cada indivíduo possui o equivalente a mais de meio milhão de livros armazenados no computador, simultâneo às informações retidas no celular ou na fita magnética do cartão de crédito. Ocorre que a junção de todos materiais, das mais diversas fontes, constitui um ecossistema que possui 300 *exabytes* (300.000.000.000.000.000.000 itens) de conteúdo produzido pelo próprio homem (LEVITIN, 2015).

O destaque da revolução digital que o mundo tem testemunhado é sua tendência de continuidade ou, quem sabe, de ampliação, uma vez que as informações surgem cada vez mais depressa em computadores cada vez mais potentes. Esse fluxo termina na ponta dos dedos das pessoas que, atualmente, dispõem de máquinas com poder de processamento maior do que as utilizadas no controle da Missão Apolo (OVERBYE, 2012, tradução do autor). Os impactos positivos dessa abrangência podem ser vistos no próprio Brasil que, conforme CGI.br (2019), saiu de 39% da população brasileira que usava a internet, em 2009, para 70%, em 2018, o que representa uma estimativa de 126,9 milhões de indivíduos com dez anos ou mais conectados à rede.

Nesse cenário de ampliação do alcance da tecnologia, encontra-se a comunidade surda[1] que, apenas no Brasil, soma cerca de 10,7 milhões de pessoas (GANDRA, 2019). Os integrantes deste grupo, contudo, não desfrutam plenamente dos benefícios desta expansão, pois ainda enfrentam dificuldades para assimilar novos conhecimentos ou expor suas ideias em um simples diálogo (inclusive entre dois surdos). Isso porque, historicamente, todos aqueles que diferem-se da maioria tendem a apresentar dificuldade de expressão e encontrar uma barreira de inclusão muito grande em todos os aspectos de sua vida. Tais características tornam-se ainda mais expressivas quando, por motivos genéticos, ambientais ou demográficos, esses indivíduos possuem alguma patologia que afeta sua habilidade comunicativa (DUARTE et al., 2013).

Lamentavelmente, apesar de todas as evoluções da humanidade e da miscigenação de raças, crenças e culturas, é curioso observar como, na imensa maioria dos registros históricos, a comunidade surda dificilmente é notada ou referenciada. Utilizando-se desta linha de pensamento, Strobel (2008) reforça este anonimato historial e explica o que poderia ser a causa deste panorama. Ocorre que, apesar da presença do povo surdo ser tão antiga quanto à humanidade, nos diferentes momentos históricos nem sempre essas pessoas foram respeitadas em suas diferenças ou mesmo reconhecidas como seres humanos (STROBEL, 2008).

Dessa forma, nascer surdo significava estar obstruído de almejar uma carreira profissional de sucesso ou, simplesmente, receber uma formação acadêmica adequada. Felizmente, é possível dizer que hoje em dia já existem diversas leis, instituições e ferramentas que transformaram essa conjuntura, proporcionando uma nova ótica de vivência para essa parcela da população. Algumas personalidades reforçam essa opinião quando alcançam reconhecimento mundial, caso de um dos músicos mais aclamados da história, Ludwig van Beethoven, que desde os 28 anos de idade já ouvia muito mal ao ponto de, na estreia de sua obra mais famosa, não conseguir escutar o aplauso do público (GLOBO, 2017); e de Bill Clinton, diagnosticado com perda moderada da audição e que, apesar disso, foi eleito duas vezes para o cargo de presidente dos Estados Unidos da América (CRISTIANO, 2020).

Um dos principais pilares para viabilizar essa tonificação e reconhecimento da comunidade surda no Brasil foi a Língua Brasileira de Sinais (Libras) que, de acordo com a Lei nº 10.436/02, é uma "forma de comunicação e expressão, em que o sistema linguístico de natureza visual-motora, com estrutura gramatical própria, constituem um sistema linguístico de transmissão de ideias e fatos, oriundos de comunidades de pessoas surdas do Brasil" (BRASIL, 2002). O problema, contudo, é que a absoluta maioria dos conhecedores desta língua são os próprios surdos, situação que denota pouco envolvimento do público em geral com o tema.

A falta de interesse dos ouvintes em aprender uma língua gestual possui algumas causas. Uma delas é que, em virtude da ausência de contato ao longo da vida, muitos indivíduos

[1] A diferença dos termos *deficiência auditiva* e *surdez*, segundo Aragon e Santos (2015), é que "a definição de deficiência auditiva considera que a pessoa com alguma limitação ou impedimento auditivo tem uma incapacidade, enquanto a definição de surdez considera o sujeito surdo como aquele que tem apenas uma diferença linguística e, consequentemente, uma diferença cultural". Portanto, no decorrer deste documento, será adotado o termo *surdez* e relacionados, pois considera-se que o surdo possui toda capacidade de aprender e desenvolver-se.

sequer conhecem a existência da Libras - ou acreditam que é apenas uma forma de mímica para representar a língua portuguesa. Também, é possível identificar um certo egoísmo da população. Esse sentimento é colossal e comanda o mundo, tanto que, se fosse dado a um indivíduo o direito de escolher entre a própria aniquilação e a do mundo, é temerário constatar para qual lado a maioria inclinaria-se (SCHOPENHAUER, 2001). Desse modo, sendo que todo ecossistema - profissional e pessoal - que cerca o ser ouvinte é favorável à sua condição, possivelmente as exceções que poderiam motivá-lo a estudar essa nova língua seriam a existência de um ente próximo com determinado grau de surdez ou uma vocação trabalhista, tal como aspiração à docência.

Todavia, os participantes do Exame Nacional do Ensino Médio (ENEM) 2017 - que contou com 7.603.290 milhões de inscritos - precisaram fazer suas redações sobre o tema "Desafios para a formação educacional de surdos no Brasil" (INEP, 2017), ou seja, mais um fato que atesta a existência de um engajamento social crescente acerca deste assunto. Concomitantemente, a tecnologia aumenta sua relevância com os adventos do Aprendizado de Máquina (*Machine Learning*), Aprendizagem Profunda (*Deep Learning*), Internet das Coisas (*Internet of Things*) e demais ramos de pesquisas recentes em tecnologia da informação (TI). Dessa maneira, é possível constatar que existem recursos suficientemente capazes de compor aplicações que podem suprir a lacuna de conversação que atinge a comunidade surda, favorecendo a mudança de percepção das pessoas para com esta esfera do diálogo.

Mais do que conter, é possível reparar essa brecha. Isso porque um sistema que estimule o contato entre interessados e fluentes na língua brasileira de sinais, por exemplo, auxiliaria na transposição desse obstáculo de comunicação, à imagem do que faz o *Google Tradutor* (GT) - que interliga nações distintas, independente de sua língua nativa. Apoiando-se nos mesmos princípios, criou-se o aplicativo **DeafEnd**. Essa palavra não existe e foi formada para transmitir duas essências:

- **Fim da surdez:** a palavra *deaf*, em Inglês, significa "surdo", enquanto *end* significa "fim". Dessa forma, a junção desses dois termos busca fazer uma metáfora como uma forma de encerrar a exclusão de um sujeito com carências auditivas, já que o aplicativo possui reconhecimento de fala.
- **Defesa:** como este é um protótipo focado em Libras, a pronúncia */dé-fend/* remete ao verbo da língua portuguesa "defender" e enfatiza todas as medidas que vem sendo tomadas para que seja possível ampliar a inclusão desses cidadãos na sociedade.

Assim sendo, todas as atividades executadas no presente trabalho tinham como objetivo construir uma solução tecnológica que aumente a comunicabilidade da população surda, viabilizando novas alternativas para um diálogo baseado na língua de sinais. Para tanto, criou-se um aplicativo móvel que possui, entre outras, a funcionalidade de reconhecer frases da língua portuguesa e representá-las gestualmente, através de imagens animadas (*GIFs*) de Libras. A transposição entre línguas é realizada por meio de técnicas de processamento de linguagem natural (*natural language processing*), um ramo da inteligência artificial que ajuda computadores

a entender, interpretar e manipular a linguagem humana. O PLN é fruto da união interdisciplinar de áreas como ciência da computação e linguística computacional, que buscam associar a comunicação humana e o entendimento dos computadores (SAS, 2019).

Tendo em vista que a biblioteca utilizada para usufruir destas técnicas é incompatível com dispositivos móveis - e também buscando evitar o *download* de complexos modelos de redes neurais, que demandariam certo espaço de armazenamento dos aparelhos para executarem localmente - construiu-se uma Interface de Programação de Aplicações (*Application Programming Interface*), que são conjuntos de instruções e padrões de programação que recebem e processam dados em um formato padronizado baseado em requisições *HTTP* (ANTUNES, 2019). Desse modo, o aplicativo consegue enviar as sentenças reconhecidas para serem processadas pela API realizando simples requisições via internet.

Já os dados que compõe as telas da aplicação móvel foram armazenados em um banco de dados *NoSQL*, que são criados para modelos de conteúdo específicos e têm esquemas flexíveis muito úteis na criação de aplicativos modernos. Esse tipo de sistema é amplamente reconhecido por sua facilidade de integração, simplicidade e performance em escala (AWS, 2019). E o aplicativo, de fato, foi construído através de um *framework*[2] de desenvolvimento híbrido, caracterizado por permitir que *apps* sejam produzidos utilizando uma única linguagem de programação - geralmente compatível com a *web* - e aproveitem da mesma base de código para gerar aplicações tanto para aparelhos com sistema operacional iOS, quanto para Android (MALINOSQUI, 2019). Reunindo todos estes métodos, este projeto visou utilizar esta aplicação móvel como uma Prova de Conceito (*Proof of Concept*) que verifique se é possível facilitar e, até mesmo, aprimorar a comunicação de sujeitos surdos com os demais indivíduos - independente do seu conhecimento.

1.1 Organização do trabalho

O presente documento está distribuído em 6 capítulos, incluindo esta Introdução. O Capítulo 2 caracteriza-se como o suporte teórico do trabalho, uma vez que descreve conceitos essenciais como a surdez, artefatos tecnológicos, metodologias de desenvolvimento móvel, inteligência artificial e bibliotecas externas. Nele estão contidos todos os conhecimentos que nortearam as atividades práticas de arquitetura e codificação. O Capítulo 3, por sua vez, apresenta as aplicações móveis que, além de semelhantes ao produto construído, são consideradas casos de sucesso no mesmo ramo de atuação.

Já o Capítulo 4 aborda a sistemática realizada para desenvolver o aplicativo - desde o levantamento de requisitos até a implementação. Finalmente, o Capítulo 5 transcreve quais foram os resultados obtidos e as contribuições geradas pela realização deste projeto, enquanto o Capítulo 6 finaliza esta monografia expondo as considerações finais do Autor.

[2]É um pacote de códigos prontos que podem ser utilizados no desenvolvimento. A proposta de uso dessa ferramenta é aplicar funcionalidades, comandos e estruturas já prontas para garantir qualidade no projeto e produtividade (SOUZA, 2019).

2 REFERENCIAL TEÓRICO

Após ter introduzido o presente trabalho, expondo a problemática que o mesmo se propõe a resolver e a forma como essa solução deverá ser criada, este Capítulo propõe construir uma base teórica acerca deste tema que, apesar de estar inserido diariamente na vida das pessoas, poucas são as que realmente o conhecem em detalhes. Tendo em vista que, assim como expressa Werneck (2006), "Um dos maiores obstáculos ao desenvolvimento do conhecimento humano advém da imprecisão dos termos utilizados na constituição dos saberes", é muito importante que, ao dissertar-se sobre qualquer assunto, o mesmo seja deveras fundamentado e, por conseguinte, compreendido pelos interessados.

Assim sendo, este Capítulo será dedicado a expor a teoria histórica e os conceitos relacionados à surdez. Também, será desenvolvido um conhecimento importante sobre a parte técnica do presente projeto, dissertando sobre as tecnologias existentes e os serviços utilizados em sua arquitetura. Esta abordagem tem como intenção reforçar tudo que foi exposto até o momento e propiciar ao leitor deste documento uma compreensão completa, ainda que superficial, do contexto em que uma pessoa surda está inserida, as dificuldades que ela enfrenta atualmente e as ações realizadas com o objetivo de transpor tais barreiras, além de todo processo de construção do aplicativo. Todo conteúdo exposto nesta Seção é um misto da leitura de artigos, trabalhos de conclusão de curso, livros e *websites*, somados à visão do Autor deste trabalho.

2.1 Surdez

A deficiência auditiva é a dificuldade de ouvir e, surdez, a impossibilidade de ouvir. Há diversas categorias de privação auditiva quanto à sua intensidade, tal como leve, moderada, severa e profunda e é importante destacar que nem todo surdo é mudo e, por isso, não é correto o termo "surdo-mudo" (JUSTIÇA, 2009). Esta objeção na audição é resultado de uma alteração no sistema e/ou nas vias auditivas, que acaba reduzindo ou impedindo o acesso aos estímulos sonoros. A gravidade também depende de questões como a localização (ouvido médio, interno, unilateral, bilateral etc.) e do momento da perda (antes ou depois da aquisição da linguagem) (NUNES et al., 2015).

Nunes et al. (2015) ainda explica que, por considerar tantas variáveis, o título "surdo" torna-se extremamente abrangente e pode abordar casos completamente distintos, tais como: uma criança que nasce com surdez profunda nos dois ouvidos, filha de pais surdos e bastante ativos dentro da comunidade; ou um jovem com uma perda leve - após um acidente na adolescência - filho de pais ouvintes e que nunca tiveram contato com outros surdos. Tal discrepância entre contextos atesta como pode ser difícil zelar por questões educacionais, culturais e até mesmo de cidadania desses indivíduos.

As dificuldades para conduzir este tema mostram-se tão complexas que, apesar do au-

xílio da tecnologia e da robusta base de conhecimento desenvolvida acerca do assunto, ainda podem ser encontradas nos dias de hoje. Sacks (2010) descreveu em seu livro - publicado em 1986 e traduzido em 2010 - que somos notavelmente ignorantes a respeito da surdez. De fato, desde o princípio, pessoas com incapacidades (de diversos tipos, não apenas auditiva) são rotuladas pela sociedade com estereótipos extremistas, sendo vistos desde semideuses que comunicavam-se diretamente com as divindades, até como o castigo resultante de uma vida de pecados de seus progenitores (DUARTE et al., 2013).

Apesar das causas dessa forma de abordagem - culturais, preconceituosas ou mesmo intrínsecas ao âmbito vivido - a consequência é que, até pouco tempo atrás, a população ainda apresentava um conceito bastante excludente com relação aos surdos. Esse estigma, por muitas vezes, assemelhava-se aos da Antiguidade, onde a maioria dos povos considerava qualquer indivíduo que apresentava certa fragilidade como um "peso ao Estado" e, em diversas vezes, acabava exterminando-os - tal como em Esparta, onde os recém-nascidos que manifestassem algum tipo de deficiência eram arremessados do alto do Taigeto[1] (DILLI, 2010).

Felizmente, as novas e emergentes tecnologias têm sido aliadas no processo cujas palavras-chave são o conforto, a segurança e a flexibilidade, sustentando o desenvolvimento de produtos, ambientes e serviços que objetivam atender as necessidades de pessoas de todas as idades, habilidade e tamanhos (SONZA et al., 2013). Além disso, a prosperidade da humanidade possibilitou uma melhoria significativa nas pesquisas científicas, meio que esclarece, cada vez mais, as causas e possíveis tratamentos de problemas auditivos. Apesar de ser evidente o longo caminho que existe até alcançar-se uma inclusão completa dessa comunidade, estas evoluções são alentadoras, pois impulsionam a mudança de um paradigma, peça-chave na busca de uma qualidade de vida maior para todos.

2.1.1 Evolução histórica

Nota-se, com o conteúdo do Subcapítulo 2.1, que os surdos vivenciaram períodos árduos ao longo da história e que este cenário vem sendo transformado de forma gradual. Para que seja compreensível a evolução dos últimos tempos, é importante abordar os pontos principais quanto à evolução histórica das línguas de sinais e às conquistas da comunidade surda, foco principal desta Seção. Inicialmente, Morais et al. (2018) explica que não há registro histórico que aponte onde e quando a forma básica do que viriam a ser as línguas de sinais surgiu e, por isso, não há como definir com exatidão o ponto de início da utilização dessa prática de comunicação. No entanto, é de se concordar que, possivelmente, pessoas surdas sempre existiram e, em virtude disso, algum meio de diálogo rústico era utilizado pelos primeiros povos.

Ainda assim, é possível destacar dois fatos que acredita-se serem os primeiros sinais do reconhecimento e adoção das línguas visuais-espaciais como forma de conversação, principalmente no âmbito educacional: o primeiro deles aconteceu durante a Idade Contemporânea,

[1] Abismo de mais de 2.400 metros de altitude, próximo de Esparta.

quando Ponce de León, baseado em datilologia, escrita e oralização, construiu um trabalho para a educação dos surdos que tornou-se referência aos educadores desta área; e o segundo, que data de 1799, foi a fundação do Instituto Nacional de Surdos em Paris, por Charles-Michel de l'Epée, conhecido como a primeira escola de surdos do mundo (MORAIS et al., 2018).

Tais eventos estão diretamente ligados com a evolução da Libras - sintetizada no Quadro 1 - já que a mesma foi criada com base na língua de sinais francesa, quando o francês Eduard Hernest Huet (ex-aluno do Instituto de Paris) veio para o Brasil e impulsionou o início da educação de surdos durante o segundo império.

Quadro 1 – Eventos marcantes na evolução da língua de sinais no Brasil.

Ano	**Acontecimento**
1857	Fundada a primeira escola de surdos do Brasil, conhecida hoje como Instituto Nacional de Educação de Surdos (INES).
1960	W. Stokoe faz o primeiro estudo linguístico sobre língua de sinais americana (ASL).
1978	O III Congresso Internacional (Gallaudet) divulga ideias de comunicação total (português sinalizado), influenciando diversos países.
1980	Entre a década de 70 e a atual, iniciam movimentos da comunidade surda exigindo mais direitos (surge o termo ***deaf power***).
1987	Criada a Federação Nacional de Educação e Integração de Surdos - FENEIS.
1991	Libras é reconhecida oficialmente pelo governo do estado de Minas Gerais.
1994	Declaração de Salamanca: documento reconhecido mundialmente que trata dos princípios, políticas e práticas em educação especial.
1995	É criado um comitê de luta pela oficialização da língua de sinais (Libras).
2002	Libras é oficializada pela Lei nº. 10.436 e ganha o *status* de língua. Na prática, esta Lei implica que instituições públicas devem ofertar acessibilidade em línguas de sinais e permite que os sistemas educacionais optem por ofertar educação bilíngue.
2005	O Decreto nº. 5.626 - que regulamenta e Lei nº. 10.436 - considera pessoa surda aquela que, por ter perda auditiva, compreende e interage com o mundo por meio de experiências visuais; sanciona Libras como disciplina curricular obrigatória nos cursos de formação de professores; expõe os requisitos para a formação de professores e instrutores de Libras; cria o PROLIBRAS; dentre outras garantias.
2006	O MEC cria os primeiros cursos de nível superior em Letras/Libras.
2010	A Lei nº. 12.319 regulamenta a profissão de tradutor e intérprete de Libras.
2011	O Decreto nº. 7.611 dispõe sobre a educação especial e também foca nas diretrizes para a elaboração de materiais didáticos nesse contexto.
2014	No dia 24 de abril (data da publicação da Lei nº. 10.436/02), é celebrado o dia nacional da Libras, oficializado pela Lei nº. 13.055.
2017	Estudantes surdos puderam, pela primeira vez, ter acesso a vídeos com as questões do Enem traduzidas para Libras.
2018	O curso EAD de pedagogia bilíngue venceu o Reimagine Education 2018, prêmio que é considerado o "Oscar" da educação mundial; além de conquistar o primeiro lugar na categoria "Educação híbrida".

Fonte: Adaptado de Morais et al. (2018)

Com a evolução apresentada nesta linha do tempo, é importante destacar que, apesar

das grandes dificuldades superadas ao longo do tempo e de algumas barreiras que ainda podem ser encontradas nos dias atuais, o acolhimento da sociedade para com os surdos melhorou apreciavelmente. Portanto, é fundamental olhar para o passado, realçando todas as conquistas alcançadas ao longo da história, buscando vislumbrar um futuro que contenha acessibilidade completa e duradoura para qualquer indivíduo.

2.1.2 Acessibilidade e legislação

Em paralelo ao progresso tecnológico e aos crescentes debates acerca de igualdade e direitos individuais, o tópico *acessibilidade* vem ganhando bastante notoriedade, tanto mundial, quanto nacionalmente. E essa importância cada vez maior é fundamental para o desenvolvimento conjunto dos cidadãos, tendo em vista que acessibilidade, em sua essência, significa incluir a pessoa com deficiência na participação de atividades como o uso de produtos, serviços e informações (INES, 2020). Levando em consideração esta definição, pode-se dizer que os primeiros feitos envolvendo acessibilidade específica para a comunidade de surdos foram os telefones TDD e os relógios despertadores vibratórios (BOTELHO, 2015).

Porém, nem sempre a acessibilidade foi vista como uma característica essencial para uma nação bem-sucedida. Em razão disso, movimentos como o "Setembro Azul" tiveram de ser criados para dar maior visibilidade à causa e conscientizar a população sobre os desafios enfrentados pelos sujeitos surdos (STENO, 2018). É animador perceber que esse cenário vem sendo, aos poucos, transformado. O Brasil, por exemplo, já dispõe de diversas soluções que traduzem conteúdos digitais para Libras - conforme será detalhado no Capítulo 3.

Todavia, apenas a existência de ferramentas inovadoras não garante o acesso de todos os necessitados a essa espécie de recurso. Era preciso, da mesma forma, transformar convenções de acessibilidade em leis e assegurar que os direitos dos surdos seriam preservados. Em concordância com os principais eventos destacados no Quadro 1, Botelho (2015) destaca as duas maiores referências brasileiras de ações de inclusão no âmbito legal como sendo a publicação da Lei 10.436/02 e do Decreto 5.626/05, que reconhecem a Libras como meio de comunicação natural e legítimo dos surdos e assegura-lhes o direito de usá-la perante a sociedade.

Ademais, convém mencionar a Lei nº. 10.098, pioneira na legislação totalmente voltada à acessibilidade; o decreto nº. 5296 que, além de fortalecer as principais ideias da Lei anterior, resgatou as normas técnicas da ABNT como parâmetros de acessibilidade (todas descritas no manual 9050); o estatuto da pessoa com deficiência - também conhecido como LBI (Lei Brasileira de Inclusão); bem como todas as convenções na área da saúde, portarias na área da educação e resoluções na área de trânsito que, cada qual em seu domínio, regulamentam a prática destas atividades de uma forma mais acessível (COELHO, 2018).

Com isso, é satisfatório notar que as atividades de acessibilidade não limitam-se à construção de ferramentas ou utensílios customizados e, de fato, já abrangem áreas mais complexas e abstratas, que impactam diretamente a qualidade de vida das pessoas com algum tipo de dificuldade. Essa combinação entre teoria e prática é essencial dado que, quando verdadeiramente

aplicadas, ambas contribuem de forma idêntica na construção de um ambiente agradável para todos, independente de suas condições.

2.1.3 Libras

Esta Seção do trabalho designa-se, exclusivamente, a apresentar uma visão geral sobre as regras que norteiam a criação de frases na língua de sinais brasileira. Convém ressaltar, entretanto, que devido à sua complexidade - e por não ser o foco do presente projeto - pontos importantes, mas extremamente específicos, não serão abordados, de modo que a essência do texto mantenha-se em conteúdos decisivos para o planejamento do aplicativo. Iniciando pela nomenclatura, é possível encontrar algumas opções dentro da literatura especializada: Língua de Sinais dos Centros Urbanos do Brasil (LSCB); Língua de Sinais Brasileira (LSB); Língua Brasileira de Sinais (LIBRAS ou Libras). Entretanto, a jurisdição federal, com a Lei de Libras, oficializa a terminologia como Língua Brasileira de Sinais (DONATO; DINIZ, 2010). De acordo com o exposto, é interessante salientar que optou-se por utilizar o termo "*Libras*" ao longo deste documento.

Antes mesmo de detalhar as especificidades da Libras, vale dizer que, na maioria do mundo, há pelo menos uma língua de sinais usada amplamente pelos sujeitos surdos de cada país - diferente da língua falada utilizada na mesma área geográfica. Isto se dá pois essas línguas são independentes das línguas orais, uma vez que foram produzidas dentro das comunidades surdas. Sendo assim, a Língua de Sinais Americana (ASL) é diferente da Língua de Sinais Britânica (BSL), que difere da Língua de Sinais Francesa (FSL) (STROBEL; FERNANDES, 1998). Ainda que certas línguas tenham sofrido forte influência de alguma outra - caso da própria Libras, que possui exemplos bastante similares com a ASL e a FSL - este esclarecimento inicial cumpre a função de clarificar a inexistência de uma língua de sinais única e universal.

Conforme explanado na segunda linha do Quadro 1, os estudos linguísticos pioneiros sobre língua de sinais iniciaram com William Stokoe em 1960. Tal episódio marcou a história da comunicação humana, dado que este autor apresentou uma análise descritiva da língua de sinais americana em uma época na qual todos os estudos desta área concentravam-se nas análises de línguas faladas (QUADROS; PIZZIO; REZENDE, 2009). Já no Brasil, confirme Donato e Diniz (2010), os estudos sobre as línguas de sinais iniciaram na década de 1980, por Ferreira-Brito e Felipe, seguidas por Karnopp e Quadros. Quanto à constituição, segundo Saussure (2006 apud DONATO; DINIZ, 2010), a língua é constituída de signos e esses, por sua vez, são constituídos de significante e significado. O *significante* é o conceito e o *significado* é a representação mental que possui-se, ou seja, essa representação ocorre através de sons nas línguas orais e de imagens nas línguas de sinais.

Devido a isso - cada qual em seu contexto, obviamente - é possível identificar certas semelhanças entre as línguas, já que todas são estruturadas a partir de unidades mínimas que formam unidades mais complexas, ou seja, todas possuem os seguintes níveis linguísticos: o fonológico, o morfológico, o sintático, o semântico e o pragmático (ALMEIDA, 2013). Mesmo

assim, cada língua apresenta uma ordem básica das palavras, de acordo com suas variações linguísticas. Isso quer dizer que, individualmente, existe uma ordem dominante, tendo em vista a formalização da estrutura das frases e a distinção de sentenças corretas pelos indivíduos que a dominam. No caso da Libras, existe uma ordem básica que determina o domínio do aspecto formal da língua pelo sinalizador, que é sujeito-verbo-objeto (SVO) (DONATO; DINIZ, 2010).

Mesmo com a ressalva sobre a flexibilidade da ordem das frases em Libras, as autoras Felipe (1989 apud QUADROS; KARNOPP, 2004) e Ferreira-Brito e Langevin (1995 apud QUADROS; KARNOPP, 2004) puderam observar que parece haver uma ordenação mais básica e frequente que as demais. E, de encontro com o que foi dito anteriormente, essa ordem é sujeito-verbo-objeto (QUADROS; KARNOPP, 2004). Para confirmar esta tese, Quadros e Karnopp (2004) levantaram 9 evidências que indicam isso - por questões de objetividade, apenas duas serão citadas como exemplo:

- **Todas as sentenças SVO são gramaticais:**
  - JOÃO GOSTAR MARIA
  - *O João gosta da Maria*
- **As ordens OSV e SOV ocorrem somente quando há alguma coisa a mais na sentença, como a concordância e as marcas não-manuais:**
  - [MARIA]tópico JOÃO GOSTAR
  - *A Maria, o João gosta.*

Para complementar o entendimento, Donato e Diniz (2010) explicam que, na língua brasileira de sinais, não pronunciar o sujeito e o objeto é possível, basta observar o contexto sintático em que estes elementos são recuperáveis. Além disso, os autores fornecem alguns exemplos de derivações possíveis para sentenças SVO em Libras, que podem ser visualizados no Quadro 2.

Quadro 2 – Derivações da estrutura SVO em Libras.

Estrutura	**Libras**	**Português**
SV	ELE ELA DORMIR	*ele e ela dormiram*
SOV	PAULO ELE CARLOS BATER	*Carlos bateu em Paulo*
OSV	CARRO JOÃO COMPRAR	*João comprou um carro*

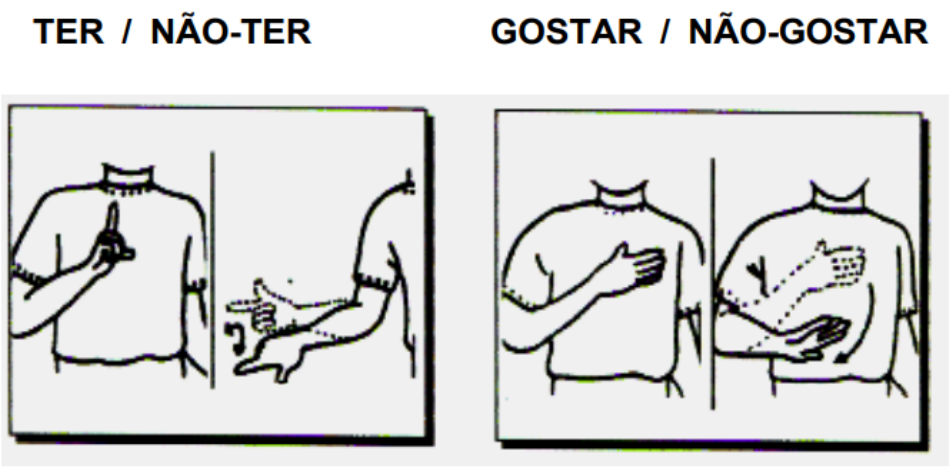
Fonte: Donato e Diniz (2010)

Outro aspecto da língua que envolve tanto a fonologia quanto a sintaxe é a *prosódia*. Este tópico que, na língua portuguesa, envolve pontos como ritmo, aumento e queda do *pitch* da voz, entonação, pausas e volume, tem seu equivalente nas línguas de sinais com expressões faciais e posturas corporais (QUADROS; PIZZIO; REZENDE, 2009). Este tipo de característica evidencia-se no que refere-se à composição de frases, sendo que é necessário atentar-se aos movimentos que serão realizados nas frases (STROBEL; FERNANDES, 1998):

- **Afirmativas:** a expressão facial é neutra;

- **Interrogativas:** sobrancelhas franzidas e um ligeiro movimento da cabeça, inclinando-se para cima;
- **Exclamativas:** sobrancelhas levantadas e um leve movimento da cabeça inclinando-se para cima e para baixo;
- **Negativas:** são um pouco mais complexas em razão de suas variedades e podem ser executadas através de três processos:
    1. incorporando-se um sinal de negação diferente do afirmativo:

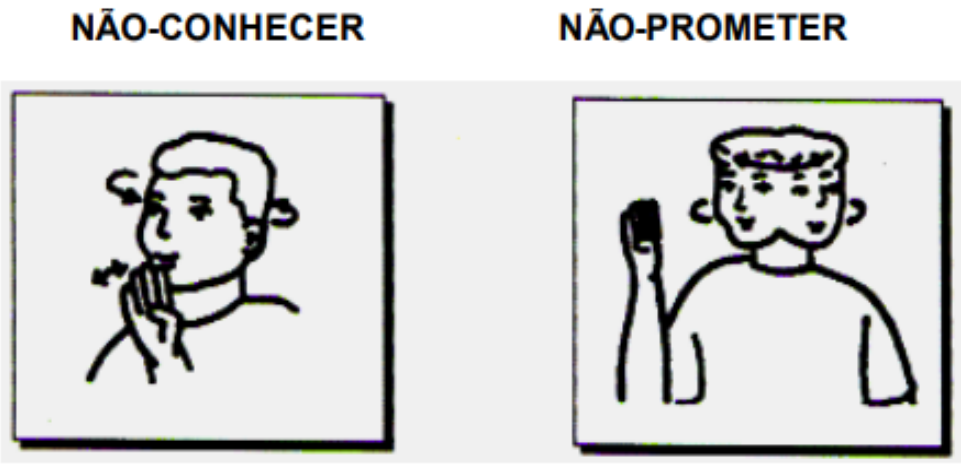
Figura 1 – Frases negativas: Processo Um



Fonte: Strobel e Fernandes (1998)

2. realizando-se um movimento negativo com a cabeça, simultaneamente à ação que está sendo negada:

Figura 2 – Frases negativas: Processo Dois



Fonte: Strobel e Fernandes (1998)

3. acrescida do sinal NÃO (com o dedo indicador) à frase afirmativa. Em algumas ocasiões, podem ser utilizados dois tipos de negação ao mesmo tempo (movimentando

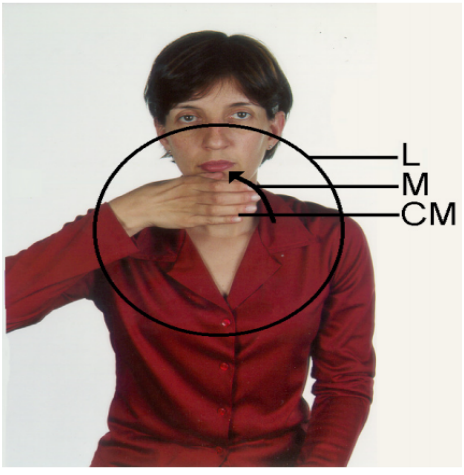
a cabeça além da negação gestual, por exemplo):

Figura 3 – Frases negativas: Processo Três

Fonte: Strobel e Fernandes (1998)

Possivelmente, para qualquer cidadão que nunca teve contato com este tipo de língua gestu-visual-espacial, tantas informações seja algo espantoso. E apesar de todas informações já expostas nesta Seção, ainda falta abordar, mesmo que brevemente, os atributos que viabilizam a verdadeira diversidade da Libras. Isso porque os sinais são formados a partir da combinação do movimento das mãos com um determinado formato, em um determinado lugar (que pode ser uma parte do corpo ou um espaço em frente ao corpo). Todas estas variações e articulações das mãos, que podem, inclusive, ser comparadas aos fonemas e aos morfemas, são chamadas de parâmetros (ALMEIDA, 2013). A Figura 4 identifica os três parâmetros considerados principais:

Figura 4 – Sinais: Parâmetros principais



Fonte: Quadros e Karnopp (2004)

Contudo, Honora e Frizanco (2009) complementam a Figura 4 e explicam que, para

produzir um sinal correto na língua brasileira de sinais, pode ser necessário utilizar até cinco parâmetros que, segundo os autores, são:

- **Configuração das mãos (CM):** são as formas em que as mãos são dispostas para a execução do sinal. É a representação de como estará a mão de dominância (direita para destros, esquerda para canhotos) no momento do sinal - existem sinais que podem utilizar as duas mãos.
- **Ponto de articulação ou Locação (PA/L):** é o local onde incide a mão configurada para a execução do sinal. Pode ser alguma parte do corpo ou um espaço neutro, tanto vertical (ao lado do corpo), quanto horizontal (na frente do corpo).
- **Movimento (M):** alguns sinais têm movimento, outros são estáticos. Este atributo indica o deslocamento da mão no espaço durante a execução do sinal.
- **Orientação ou Direcionalidade (O/D):** é a direção que o sinal terá para ser executado.
- **Expressão facial e/ou não-manuais (EF/ENM):** diversos sinais necessitam de um complemento facial e até corporal para fazer com que sejam compreendidos. Tais expressões são as adaptações feitas, geralmente através do rosto, para dar vida e entendimento ao sinal executado.

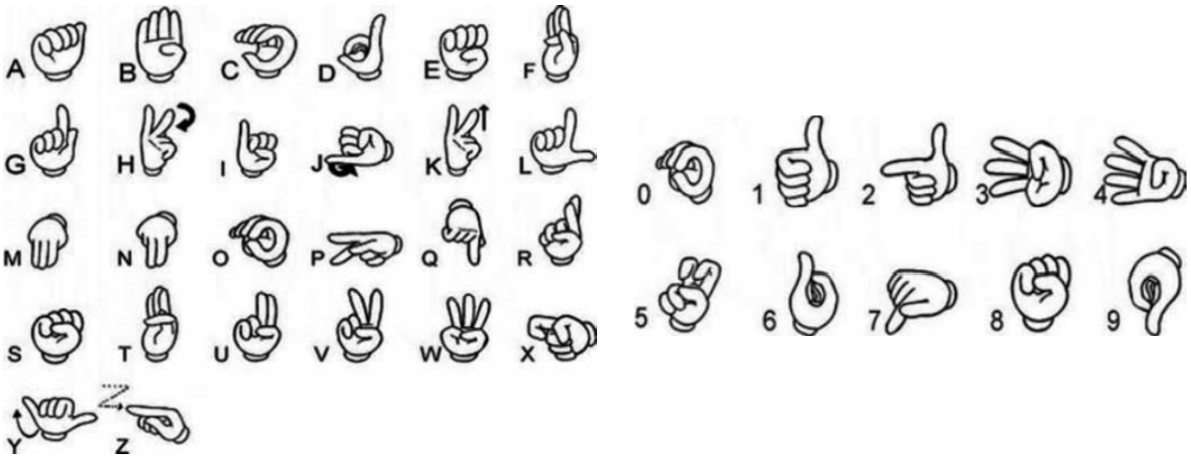
Reiterando a ideia exposta no início, buscou-se reunir um apanhado sobre a língua brasileira de sinais neste trecho do projeto. Apesar de todas as informações descritas, o conceito primordial que precisa ser frisado é que a essência dos sinais é representar a informação, pois os surdos lidam com memória visual (ALMEIDA, 2013). Portanto, os tópicos relevantes à aplicação móvel - como a estrutura das frases em Libras; como funcionam sentenças afirmativas, negativas e interrogativas; e as condições relevantes para a execução de um sinal - puderam ser abordados e, com isso, solidificou-se uma importante base de conhecimento para que, posteriormente, fique mais fácil entender as decisões arquiteturais que foram tomadas durante o desenvolvimento (Capítulo 4).

2.1.3.1 Alfabeto manual e números

Trazendo um acréscimo à Seção 2.1.3, é importante fazer um curto detalhamento sobre como são sinalizados dois dos elementos mais básicos da língua portuguesa: letras e números. A datilologia (ou alfabeto manual) é um sistema de representação, quer simbólica, quer icônica, das letras dos alfabetos das línguas orais na sua forma escrita, porém, por meio das mãos (ALMEIDA, 2013). Normalmente, o abecedário é utilizado para soletrar os nomes de pessoas, de lugares, de rótulos, entre outros, e para os vocábulos não existentes na língua de sinais (STROBEL; FERNANDES, 1998).

Dentro do contexto deste aplicativo, os gestos individuais para representar letras foram bastante determinantes. Isso porque, nas listagens de palavras - especificadas no Capítulo 4 - criadas na aplicação móvel, disponibilizou-se os caracteres, individualmente, como filtros. Dessa forma, o usuário pode filtras as palavras através de sua letra inicial. Por fim, aproveitou-se o espaço da Figura 5 para ilustrar, além do abecedário, como são marcados os números de 0 a 9.

Figura 5 – Datilologia: Alfabeto e números de 0-9



Fonte: Almeida (2013)

2.2 Artefatos tecnológicos

Quando o assunto é evolução, possivelmente nenhuma área seja tão lembrada quanto a de tecnologia. O mais curioso é que isso soa de maneira natural, pois a sociedade progrediu de máquinas avulsas e totalmente desconectadas para aparelhos que podem comunicar-se e trocar informações, tanto local quanto mundial, graças ao avanço da rede de computadores, internet e dos mecanismos *wireless*, 3G, 4G e, agora, o 5G (ALVES; TONIOLO, 2016). Contudo, mesmo que esta prosperidade tenha impactado os equipamentos de todo setor, existe um grupo de dispositivos que destacou-se dentre os demais: os móveis.

Devido ao atual modelo de vida das pessoas, no qual o tempo passou a ser o bem mais valioso existente e qualquer ferramenta que propicie um aumento de produtividade é muito bem recebida, era necessário eliminar, por exemplo, a necessidade de estar presente fisicamente no escritório para responder um *e-mail*. Isto é, o próprio cotidiano evidenciava a necessidade de utilização de um equipamento mais móvel. Mobilidade pode ser definida como a capacidade de poder deslocar-se ou ser deslocado facilmente (LEE; SCHNEIDER; SCHELL, 2005). Dessa forma, tal definição expõe claramente um dos principais motivos que, em conjunto com outras características detalhadas adiante neste Subcapítulo, resultaram em uma inserção rápida e progressiva destes equipamentos nas mais variadas atividades da população.

Esse crescimento acelerado atingiu patamares que eram inimagináveis alguns anos atrás e o uso de celulares, *smartphones* e *tablets* já é uma constante na vida dos indivíduos. Prova disso são os dados do relatório Digital 2019 (da We Are Social e da Hootsuite), compilados por Lopez (2019), que apontam que os brasileiros passam 4h 45 minutos por dia na internet em seus dispositivos móveis (pouco mais de 50% de todo tempo gasto diariamente), o que coloca o Brasil em terceiro lugar no *ranking* mundial no quesito “uso de internet móvel”. Em termos globais, 5,1 bilhões de pessoas usam algum tipo de aparelho de telefone celular, conforme o relatório Economia Móvel 2019, da GSMA, empresa que representa o interesse de operadoras

e avalia o ecossistema móvel. O número equivale a 67% da população mundial (VEJA, 2019).

Essa forte tendência dos indivíduos ao uso desses aparelhos provém - além da mobilidade, citada anteriormente - da diversidade de aplicativos disponibilizados nas lojas *online* de cada sistema operacional móvel (*Google Play* para Android e *App Store* para iOS). Essa abundância de opções possibilita que, além das funções básicas (como efetuar ligações, por exemplo), os celulares possuam milhares de aplicativos capazes de executar as mais diversas tarefas, listadas em categorias como entretenimento, medicina e finanças (OLIVEIRA et al., 2012). Devido à facilidade de acesso dos usuários às plataformas citadas, alguns dos milhares de *softwares* ofertados tornaram-se *cases* de sucesso e revolucionaram seus segmentos, como é o caso do *WhatsApp* para comunicação e do *Uber* para transporte.

Apesar do cenário positivo, a grande maioria dos aplicativos criados não é completamente acessível, fato que impede usuários com alguma incapacidade de usufruir do programa. Esse cenário ocorre pois, ao contrário dos computadores pessoais (PCs), a acessibilidade para dispositivos móveis traz uma complexidade extra, já que além das características específicas como tamanho de tela, portabilidade e contextos de uso variado, existem metodologias - detalhadas no Subcapítulo 2.3 - bastante distintas de desenvolvimento *mobile* (SILVA; FERREIRA; SACRAMENTO, 2018). Isso acaba evidenciando que, apesar da sua popularidade, os aplicativos móveis ainda precisam aprimorar certos detalhes para atender usuários que enfrentam barreiras de acesso, seja em virtude de alguma limitação ou devido ao dispositivo utilizado.

2.2.1 Áudios

Apesar de, após o advento dos dispositivos móveis, ser possível carregar dezenas de horas de música e milhares de áudios no bolso, muitas invenções tiveram de ser validadas e atitudes tomadas ao longo da história para que se pudesse usufruir dessa tecnologia hoje em dia. De acordo com Cinema (2019), a primeira gravação da voz humana foi obtida por Thomas Edison que, em 1877, gravou a frase “Mary tinha um carneirinho” em um fonógrafo de cilindros. De lá para cá, diversos estudiosos propuseram melhorias e substituições nos materiais utilizados por Edison. Após tais alterações serem realizadas, e somando o trabalho conjunto de diversas pessoas, chegou-se ao atual formato de áudio: o áudio digital.

Nos dias de hoje, enviar áudios é uma forma reconhecidamente útil para que as pessoas consigam comunicar-se umas com as outras no meio eletrônico. Além de sua utilidade natural - que fica comprovada ao observar que os aplicativos de comunicação mais famosos atualmente implementaram tal funcionalidade -, o surgimento de novas tecnologias (como o processamento de linguagem natural, abordado na Seção 2.4.1) fizeram com que os áudios se tornassem um importante recurso para a evolução computacional. Por isso, o recurso de voz possui grande potencial para a criação de aplicativos que possibilitem uma eficaz interação humano-computador, especialmente no atual contexto de acelerada miniaturização dos sistemas embarcados (NETO; SILVA; SOUSA, 2005).

Neto, Silva e Sousa (2005) ainda complementam sua ideia definindo duas técnicas mo-

dernas que apontam como relevantes dentro da área: a *síntese de voz* (ou TTS), que consiste no processo de, dado um texto, gerar amostras de um sinal digital que deveria soar como se um humano tivesse pronunciado o texto informado; e o *reconhecimento de voz* (ou ASR), que pode ser definido como o processo inverso, onde a aplicação converte voz digitalizada em texto. Felizmente, projetando tirar proveito dessa flexibilidade da conversação oral (apesar de sintética) e compreendendo que a emissão/recepção de voz são um dos pilares de qualquer comunicação (inclusive com a comunidade surda), estas afirmações apenas reforçam o intuito inicial do presente projeto em utilizar o tratamento de áudios - especialmente das duas técnicas citadas - para viabilizar a concepção de uma aplicação móvel que favoreça o diálogo direto com sujeitos surdos. O Capítulo 4 conterá mais detalhes sobre como este artefato foi utilizado no aplicativo.

2.2.2 GIFs

GIF é um formato de imagem que pode compactar várias cenas e com isso exibir movimentos. A sigla GIF significa *Graphics Interchange Format* que, na tradução literal para português, seria *formato para intercâmbio de gráficos* (BRASIL, 2018). Em 1987, quando foi criado por Steve Wilhite, na época engenheiro da CompuServe, o foco principal da tecnologia era ser um formato de imagem de tamanho reduzido e que funcionasse em diversos computadores que existiam na época, tais como o Macintosh e o Commodore. Porém, foi graças ao desenvolvedor Marc Andreessen (responsável pelo Netscape) que as animações em GIF passaram a ter a possibilidade de executar "em *loop*", ou seja, sem parar (CAPELAS, 2017).

Costa (2020) relata que a criação desse novo tipo de imagem permitiu substituir o defasado formato RLE (*Run-length encoding*), que apresentava apenas figuras em preto e branco. O autor também atribui a popularização do GIF ao fato de que, desde que foi concebido, o formato pôde ser compartilhado por meio da rede sem consumir muita banda de internet, já que possuía um eficiente algoritmo de compressão. Tal aspecto fez-se tão relevante porque permitiu que as imagens fossem utilizadas por qualquer usuário na década de 80, quando a maioria das conexões era feita através de *modems* seriamente limitados e a velocidade de *download* era extremamente baixa.

Apesar de terem alcançado seu ápice entre os anos 80 e 90, com a disseminação da banda larga e o aumento considerável na utilização da linguagem *Flash* (que permitia ao desenvolvedor criar páginas *web* animadas e cheias de interação), os GIFs perderam seu espaço. O fato intrigante é que após passar vários anos esquecido, devido à sua recente utilização em diversos *blogs* e *posts* nas redes sociais - principalmente no sentido humorístico - os GIFs passaram a ser encontrados mais frequentemente.

Além disso, uma variação desta categoria de imagem ganhou notoriedade nos últimos tempos: o *cinemagraphs*. Basicamente, este estilo de GIF é composto por fotografias extremamente nítidas, nas quais apenas uma pequena parte da figura apresenta movimento (HAMANN, 2012). Portanto, imagens animadas mostraram-se como uma opção bastante interessante para exibir pequenos movimentos. Sendo assim, principalmente devido à baixa demanda de armaze-

namento de que necessita, o formato GIF será utilizado neste projeto como meio de exibir para o usuário a representação - em Libras - dos áudios captados pelo aplicativo. O Subcapítulo 4.3 conterá mais detalhes sobre como deu-se esta utilização.

2.3 Metodologias de desenvolvimento móvel

Por ser cada vez mais importante no dia a dia das pessoas, a tecnologia móvel deixou de ser vista apenas como um recurso pelas empresas e passou a ser uma necessidade de mercado (DURÃES, 2020). O telefone celular já é utilizado como intermédio de conexão com à internet em 98,7% dos domicílios do Brasil que possuem esse acesso (IBGEEDUCA, 2017). Logo, existe um movimento natural de instituições e profissionais voltando seu foco - de produtos ou de carreira, respectivamente - em encontrar formas de criar *apps* eficientes, que atendam a demanda dos usuários e realcem o negócio.

Vale destacar, entretanto, que não é porque os dispositivos móveis estão em uma grande ascendência que os computadores pessoais tornaram-se descartáveis. Existem ainda diversas atividades que precisam ser realizadas em um computador que possua um teclado ou uma tela grande, por exemplo (ARAÚJO, 2019). Além disso, é normal que, como toda tecnologia recente, existam vantagens relevantes, mas também problemas e pontos para serem melhorados. Uma das maiores dificuldades dessa vertente da programação envolve o suporte à uma característica de qualidade - denominada *portabilidade* pela ISO/IEC 9126 - para atender uma diversidade de plataformas móveis disponíveis atualmente (ISO/IEC-9126, 2000 apud SOUZA et al., 2016).

Neste contexto de crescimento constante e pluralidade de alternativas, Araújo (2019) aponta que o desenvolvimento para dispositivos móveis, até a presente data, vem sendo focado nas plataformas iOS e Android — estas sendo as mais utilizadas. Acontece que, nos dias de hoje, além de precisar gerar um *software* compatível com os diferentes sistemas operacionais móveis, existem duas metodologias - focadas na criação de aplicativos - distintas para este propósito: a nativa e a híbrida. Nesta Seção, será exposto as diferenças entre elas e a motivação que definiu qual foi selecionada para a construção deste aplicativo.

2.3.1 Desenvolvimento nativo

Um aplicativo é nativo quando ele é programado em uma linguagem exclusiva do sistema usando o seu *Software Development Kit* (SDK). Por exemplo, para desenvolver um *app* nativo para Android, utiliza-se a linguagem de programação Java ou Kotlin. Já para o iOS, as opções seriam o Objective C ou o Swift (GUEDES, 2017). Sua maior vantagem é que, por "falar a mesma língua" que o sistema operacional do aparelho, ele está muito mais próximo do *hardware* e, por isso, pode explorar com mais profundidade funcionalidades como sensores, câmeras, geoposicionamento, entre outras (X-APPS, 2019). Esse formato focado em *software* e *hardware* específicos também utiliza elementos primários para criar a interface de usuário, o

que auxilia na preservação dos padrões específicos por plataforma - especialmente no caso da *Apple*.

Suas desvantagens ficam por conta das restrições. A primeira refere-se à compatibilidade, uma vez que a aplicação funciona apenas na plataforma para a qual foi desenvolvida, não sendo compatível com as demais (DURÃES, 2019). A segunda diz respeito à própria experiência de desenvolvimento, já que o desenvolvedor acaba sendo forçado a utilizar um *Integrated Development Environment* (IDE) específico daquele sistema. E, por fim, a base de conhecimento do projeto, que torna-se dividida e não é absorvida por todos integrantes da equipe de tecnologia, visto que cada profissional especializa-se no seu ecossistema de ferramentas. Além disso, essa distribuição dos programadores eleva os custos de produção do *app*, levando em consideração a curva de aprendizado e a atuação independente de cada grupo (GUEDES, 2017).

2.3.2 Desenvolvimento híbrido

O aplicativo híbrido, como o próprio nome já sugere, é uma mistura de um aplicativo nativo e recursos *web* (MARKETEAM, 2020). A maior vantagem deste formato de desenvolvimento é o menor custo. Um único código serve para todas as plataformas e não é mais necessário ter equipes específicas programando Java no Android e Objective C no iOS, por exemplo. Outro aspecto favorável tange àquelas equipes que já possuem um certo domínio em HTML, CSS e JS - tecnologias mais comuns desta metodologia - e terão uma curva de aprendizado bem pequena, além de poder aproveitar muita coisa que já foi desenvolvida para *web* (LOPES, 2016). Importante salientar que as linguagens citadas - ao lado do Dart, que será abordado na Subseção 2.3.2.2 - podem ser manipuladas em editores de texto mais leves e flexíveis que as IDEs, ampliando as opções de ferramentas à disposição dos programadores.

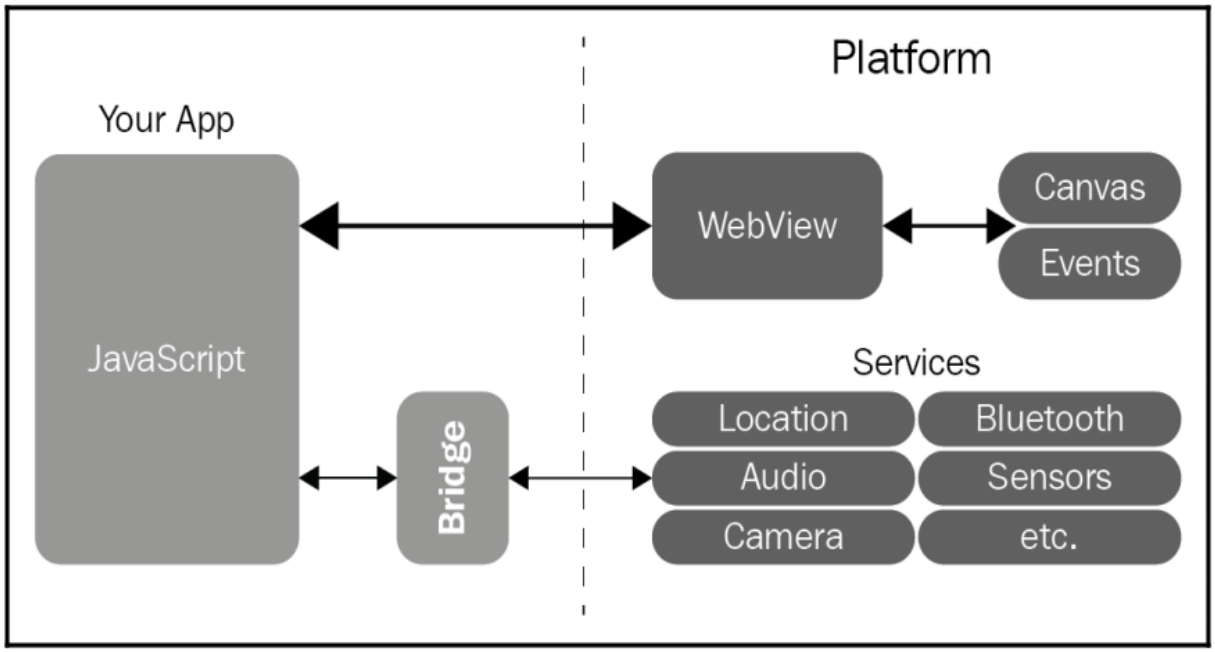
Contudo, entre as desvantagens dessa abordagem, pode-se destacar as características não funcionais da aplicação, como compatibilidade e usabilidade, que são melhores desenvolvidas e mais suscetíveis à mudança utilizando-se o procedimento nativo (BERNARDES; MIYAKE, 2016). Outra desvantagem refere-se ao padrão e qualidade do código, uma vez que pode ser necessário realizar validações em *runtime* (tempo de execução) para tratar comportamentos diferentes da aplicação, de acordo com a plataforma.

Em suma, escolher entre construir um aplicativo nativo ou híbrido envolve argumentos estratégicos e técnicos. Quesitos como os objetivos do projeto, qual é o público-alvo daquele *software* e o conhecimento prévio dos profissionais que irão criá-lo devem ser ponderados. Do ponto de vista do usuário, não há muita diferença. Um aplicativo híbrido bem construído integra-se à plataforma da mesma forma que um nativo. Existem diferenças de performance apenas em casos muito específicos que exijam realmente bastante processamento no dispositivo e, nesses casos, opta-se pelo nativo (LOPES, 2016). Como o presente projeto não encaixa-se nesta descrição, o Autor possui familiaridade com as ferramentas da *web* e existem bibliotecas compatíveis para suprir as necessidades conhecidas, decidiu-se criar este aplicativo usando desenvolvimento híbrido. Na sequência, serão expostas algumas opções que seguem este formato.

2.3.2.1 Ionic *Framework*

Ionic *Framework* é um kit de ferramentas de UI de código aberto para a construção de aplicativos móveis e de *desktop* usando tecnologias da *web* - HTML, CSS e JavaScript - com integrações para *frameworks* populares como Angular, React e Vue (FRAMEWORK, 2020, tradução do autor). Essa ferramenta foi introduzida em 2013 como um SDK de código aberto, sendo o mais antigo dos *frameworks* citados. Mesmo assim, ainda não possui nenhum *app* de destaque mundial como exemplo de uso - apesar de 86.400 times de desenvolvimento já terem utilizado-a (FRAMEWORK, 2020 apud MARTIN, 2020). Usufrui de várias tecnologias para desenvolver aplicações e acessa o *container* Cordova para usar controladores nativos da plataforma. É grátis, mas para contar com um ambiente de desenvolvimento profissional, é necessário migrar para a versão paga (MARTIN, 2020, tradução do autor). A Figura 6 demonstra sua disposição arquitetural.

Figura 6 – Arquitetura: Ionic *Framework*



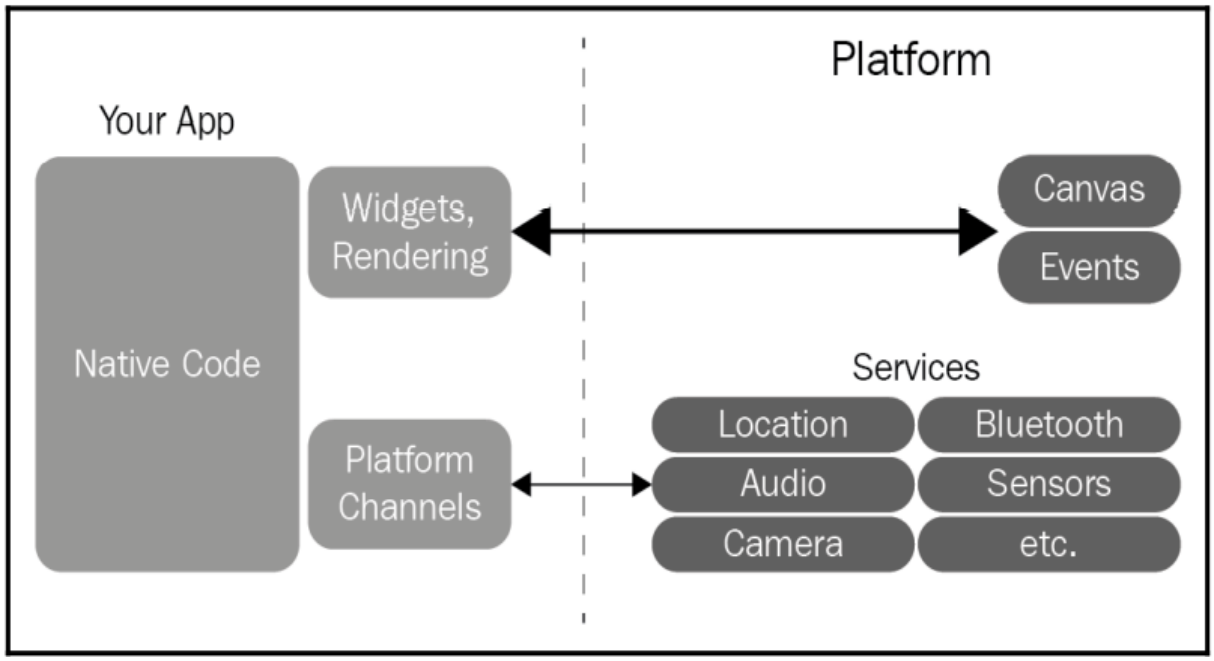
Fonte: Mainkar e Giordano (2019)

Nota-se que o Ionic, devido às possibilidades de escolha, é extremamente versátil. Além disso, em suas versões mais atuais, é possível dizer que ele conseguiu entregar os mesmos recursos de seus concorrentes, permitindo acesso à funções nativas - como a câmera - e mantendo sua compatibilidade com uma ampla gama de ferramentas *web*. Contudo, é importante ressaltar que toda essa adaptação em alto nível, impõe certa distância entre a ferramenta e o *hardware* do dispositivo, exigindo o uso de uma *Webview* e de uma "ponte" - que é o Cordova (LIMA, 2020). Dessa forma, a performance acaba não constando em sua lista de qualidades.

2.3.2.2 Flutter

Flutter é o kit de ferramentas de UI do Google para criar aplicativos compilados de forma nativa para *mobile*, *web* e *desktop* a partir de um único código-base (FLUTTER, 2020, tradução do autor). Além de ser mantido por uma empresa desse porte, essa coleção - lançada em 2018 - destaca-se especialmente quando trata-se de performance, já que possui seu próprio mecanismo de renderização. Utiliza a linguagem criada internamente pela instituição - o Dart - que, apesar de ser relativamente novo, possui um grande potencial para dominar o setor nos próximos anos. Também, é reconhecido por propiciar a melhor interface de usuário, visto que ele ignora "pontes" interativas com componentes nativos, mantendo esta ligação direta com o aparelho (MARTIN, 2020, tradução do autor). A Figura 7 expõe esses detalhes.

Figura 7 – Arquitetura: Flutter



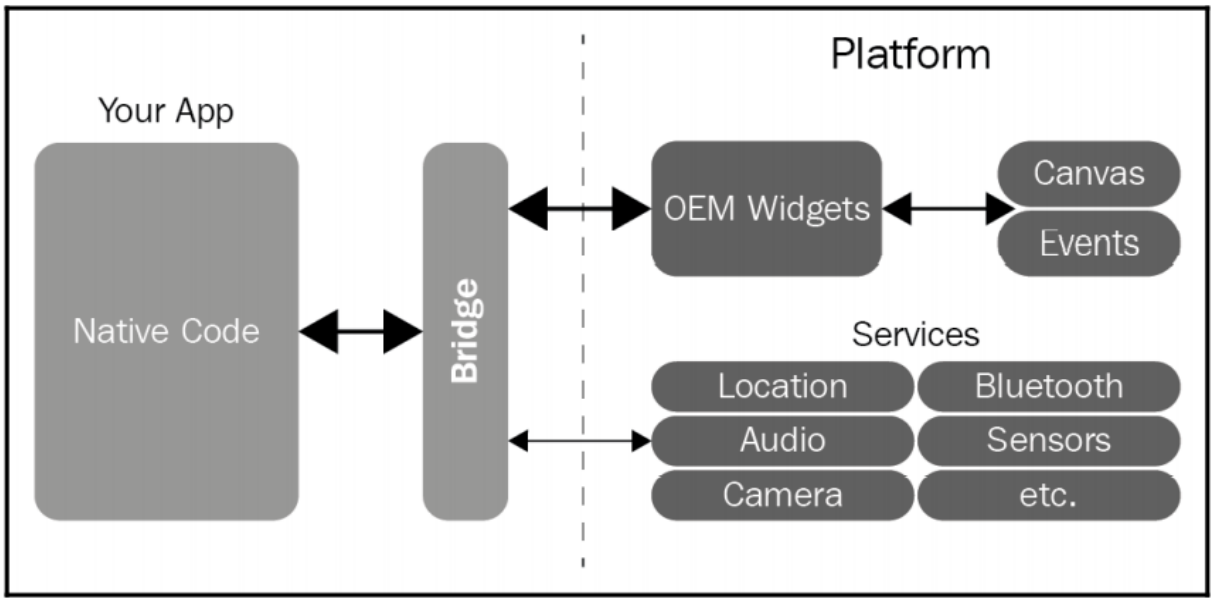
Fonte: Mainkar e Giordano (2019)

Outro detalhe interessante do Flutter é que, além das API's nativas, ele utiliza uma interface gráfica chamada Skia - também do Google - para comunicar-se com o dispositivo. Dessa forma, o aplicativo roda como um jogo no aparelho, entregando experiências bem mais interativas e visuais que as outras opções (LIMA, 2020). Todos esses detalhes fazem com que, mesmo sendo o mais novo dos 3 *frameworks*, ele seja utilizado por aplicativos de expressão, como o Alibaba e o Google Ads. Contudo, sua desvantagem também provém dessa "juventude" pois, além da curva de aprendizagem dessa linguagem de programação recente, a comunidade de desenvolvedores desta plataforma ainda é notavelmente menor que a do React Native e - até em virtude disso - não existem tantos *plugins* e bibliotecas disponíveis para projetos diversos.

2.3.2.3 React Native

React Native é um *framework* JavaScript baseado no React - uma biblioteca do Facebook para a construção de interfaces de usuário - para escrever aplicativos móveis nativos para dispositivos com sistemas operacionais iOS e Android (EISENMAN, 2018, tradução do autor). Essa coleção de código aberto, concebida em 2015, é escrita em várias linguagens, tais como JavaScript, Swift, C++ e Python. Em suma, ela providencia todas as ferramentas e serviços necessários para desenvolver um aplicação multiplataforma, com a habilidade de oferecer uma experiência "tipo-nativo" (MARTIN, 2020, tradução do autor). Da mesma forma que as anteriores, a Figura 8 reproduz a arquitetura desta plataforma.

Figura 8 – Arquitetura: React Native



Fonte: Mainkar e Giordano (2019)

Possivelmente, o aumento na popularidade deste *framework* provenha da crescente expansão do ReactJS que, assim como o React Native, tem sua estrutura baseada na biblioteca React - porém, com foco na *web*. A similaridade de sintaxe entre as duas adaptações do mesmo recurso e a baixa curva de aprendizado para desenvolvedores que já dominam uma das opções facilitam bastante esta transição. Seu grande diferencial para o Ionic é que a "ponte" com o dispositivo é muito mais performática e, com o JavaScript, é possível acessar as API's nativas do aparelho utilizado, fornecendo mais rapidez que a *Webview* de outras tecnologias híbridas (LIMA, 2020). A comparação realizada no estudo de Soral (2020) confirma esta afirmação e apresenta dados que mostram um desempenho melhor do React Native na maioria dos testes realizados, levando em considerações os quesitos: uso de memória e CPU e consumo de energia.

Quando comparado ao Flutter, a principal vantagem do React Native é a linguagem de programação, já que, segundo a pesquisa anual do StackOverflow (2020), o JavaScript é mais comumente utilizado pelos desenvolvedores (67,7%) - enquanto o Dart possui um percentual de

uso de 4,0%. Também, a própria biblioteca apresenta números melhores nessa análise quando o tópico é a de taxa de adoção, uma vez que conta com 11,5% das respostas, em detrimento dos 7,2% da outra opção (STACKOVERFLOW, 2020). Naturalmente, a maior quantidade de pessoas trabalhando com a tecnologia aumenta a quantidade de material disponível sobre o assunto. Quando soma-se a isto a maior familiaridade do Autor com este paradigma, torna-se simples explicar porque este foi o kit escolhido para sustentar a criação do aplicativo **DeafEnd**.

2.4 Inteligência artificial

Assim como outras questões complexas da existência humana, tais como "o que é amor?" ou "qual o sentido da vida?", definir o que é *inteligência* é uma tarefa bastante subjetiva e filosófica e, sem dúvida, nada trivial. Apesar disso, Medeiros (2018) faz um apanhado acerca da expressão em seu livro "Inteligência artificial aplicada: uma abordagem introdutória", no qual reúne explicações defendidas por grandes estudiosos, cada qual aceita em sua linha de pesquisa e, ao final dessa combinação, consegue observar que mesmo raciocínios tão diversos, apresentam alguns aspectos que repetem-se como características da inteligência:

- Capacidade de resolução de problemas;
- Aprendizado com o ambiente;
- Desenvolvimento de estruturas cognitivas;
- Orientação a metas.

Levando em consideração que *inteligência artificial* (IA) é uma categoria de inteligência, é possível deduzir que seja igualmente desafiador determinar este termo - mesmo que ele seja menos abrangente e também apresente os atributos listados anteriormente. Graças a isso, não existe uma definição aceita com unanimidade pelos especialistas. Luger (2013) reforça essa relação entre os termos e explica porque defini-los é tão difícil:

> O problema de *definir* o campo inteiro da inteligência artificial é semelhante ao de definir a própria inteligência: ela é uma única faculdade ou é apenas um nome para uma coleção de capacidades distintas e não relacionadas? Até que ponto a inteligência é aprendida e não existe desde o nascimento? O que acontece exatamente quando ocorre o aprendizado? O que é criatividade? O que é intuição? A inteligência pode ser deduzida do comportamento observável ou ela requer evidências de um mecanismo interno em particular? Como o conhecimento é representado no tecido nervoso de um ser humano e que lições isso nos traz para o projeto de máquinas inteligentes? O que é autopercepção? [...] Essas são perguntas não respondidas que têm ajudado a modelar os problemas e as metodologias de solução que constituem o núcleo da IA moderna. (LUGER, 2013)

Sendo assim, o presente trabalho dá preferência à descrição legítima mais simples, que expõe: "Inteligência Artificial é o estudo dos sistemas que agem de um modo que a um observador qualquer pareceria ser inteligente" (COPPIN, 2013). Contudo, mais importante do que a definição teórica são os avanços práticos que este gigantesco campo da computação vem apresentando nos últimos tempos, posto que, até alguns anos atrás, o mesmo era visto como

essencialmente acadêmico e com pouco valor para os problemas reais. Além da ampla quantidade de soluções que tiram proveitos das características dos processos de IA, outras condições que têm contribuído para essa ascensão são o desenvolvimento de algoritmos cada vez mais eficazes e eficientes e a elevada capacidade dos recursos computacionais atualmente disponíveis (FACELI et al., 2011).

Na atualidade, não é difícil encontrar aplicações que fazem uso contínuo de técnicas provenientes das pesquisas neste tema. Sistemas de inteligência artificial são utilizados em uma ampla gama de atividades, desde ajudar agentes de viagem a selecionar férias adequadas, até permitir fábricas escalonar máquinas, já que este tipo de tecnologia é especialmente útil nas situações em que métodos tradicionais seriam muito lentos (COPPIN, 2013). A utilização crescente deste modelo tecnológico em empresas mostra-se bastante sólida quando Korolov (2019) apresenta dados relevantes em seu artigo, destacando que, de acordo com uma pesquisa da Deloitte com executivos dos EUA divulgada no final do ano passado, a automação de TI é a aplicação mais popular para a IA, presente em 47% das empresas.

Esta porcentagem indica que quase metade das companhias nacionais de um país que possui uma das maiores economias globais já contam com aplicações que utilizam inteligência artificial. Com esse progresso, é natural que cada vez mais pessoas interessem-se pela área, descobertas sejam feitas e tópicos sejam expandidos. Pensando nisso, com o intuito de facilitar futuras pesquisas e auxiliar na compreensão do grau de desenvolvimento do assunto, Silva et al. (2019) explicam que qualquer produto dessa ciência pode ser classificado em três categorias:

- *Inteligência artificial fraca:* é uma corrente de pesquisa e desenvolvimento que defende a ideia de que nunca será possível construir máquinas realmente inteligentes, no real sentido da palavra, pois essa corrente alega que a inteligência demanda consciência e autopercepção, habilidades impossíveis de serem recriadas.
- *Inteligência artificial forte:* este grupo, por sua vez, acredita que um dia será possível recriar máquinas capazes de pensar, criar e exibir comportamento inteligente assim como os seres humanos, com base na concepção de algoritmos cognitivos que possam executar em computadores.
- *Superinteligência:* diferente das duas vertentes anteriores, que apresentam um caráter filosófico, esse termo foi definido pelo filósofo sueco Nick Bostrom e abrange possibilidades que variam desde o computador um pouco mais inteligente do que um ser humano até aquele milhões de vezes mais inteligente que uma pessoa normal em todas as suas capacidades intelectuais. É neste tipo hipotético de IA que estão centralizadas as principais discussões, pois também é nele que estão incluídas as promessas mais impactantes para a humanidade, como a imortalidade ou a extinção dos seres humanos.

Deste modo, independente de qual categoria seja utilizada em determinado projeto, é possível notar que as possibilidades de criação quando fala-se de IA são quase inesgotáveis. Inevitavelmente, isto gera uma grande esperança quanto à disponibilidade iminente de mecanismos revolucionários e extremamente independentes para atuarem nos mais diversos setores da eco-

nomia. Contudo, assim como tudo aquilo que é desconhecido, esse aspecto também causa uma certa preocupação sobre os limites e escopo de sua atuação. Tendo em vista que é impossível traçar a dimensão deste assunto, este trabalho resguarda-se a utilizar apenas dois dos inúmeros frutos resultantes das pesquisas nesta área e utilizará as próximas seções para falar sobre contextos específicos do processamento de linguagem natural e do reconhecimento de fala.

2.4.1 Processamento de linguagem natural (PLN)

A popularidade cada vez maior da internet, aliada à impactante evolução tecnológica citada inúmeras vezes ao longo deste trabalho, ocasionou um aumento significativo nas informações disponíveis na *web*. Em paralelo, provocou uma sobrecarga de dados, fazendo com que a busca por conteúdo relevante geralmente demande uma análise extensa de diversos documentos, em sua maior parte escritos em linguagem natural (SANTOS et al., 2014). Com isso em mente, ao analisar o atual formato de interação humano-computador, apesar das recentes melhorias de *hardware* e interface, nota-se que nem todas as pessoas têm facilidade de operar uma máquina e sabe-se, por outro lado, que os equipamentos operam melhor sob dados organizados e devidamente estruturados.

Diante do cenário exposto, é possível observar que a relação dos indivíduos com os dispositivos eletrônicos - crucial nos dias atuais - sofre certos ruídos em virtude desse contato não ser completamente orgânico e sutil. Vale ressaltar também que a Ciência da Computação (CC) sozinha, assim como, em princípio, todas as outras ciências, não consegue tratar todas essas questões. É por isso que, cada vez mais, constata-se as justificadas associações com outras áreas do conhecimento (JUNIOR, 2013). E, conforme aponta Rodrigues (2017), tais associações vêm ampliando possibilidades, já que linguistas e cientistas da computação acabam unindo-se na busca pela solução de seus entraves através de fundamentos de várias outras disciplinas, tais como: Filosofia da Linguagem, Psicologia, Lógica, Inteligência Artificial, Matemática, Ciência da Computação e Linguística.

Na mesma proporção em que estas obstruções forem sendo removidas do processo, a área ainda jovem de TI irá incorporar usuários e, consequentemente, alavancar sua expansão. Isso porque a comoditização deste domínio ainda é pequena quando comparada à ciências mais maduras e estabelecidas no campo de pesquisa como, por exemplo, a engenharia elétrica, uma vez que para fazer funcionar uma televisão, basta ligá-la na tomada, mesmo sem conhecer como ocorre o processo de transmissão da eletricidade, tensão e polaridade (JUNIOR, 2013).

Assim sendo, é exatamente nesta conjuntura que esta subárea da IA, que consiste no desenvolvimento de modelos computacionais para a realização de tarefas que dependem de informações expressas em alguma língua natural (e.g. tradução e interpretação de textos), mostra-se relevante, servindo como intermédio entre a linguagem que os humanos falam de forma natural e a linguagem compreendida pelos PCs (PEREIRA, 2011b). Apesar de não ser tão familiar para pessoas leigas em informática, o *Natural Language Processing* (NLP) pode ser facilmente encontrado em diversas atividades do cotidiano moderno. Coelho (2019) exemplifica, destacando

que no momento que o indivíduo conversa com o celular através de comandos de voz, dá instruções faladas para a assistente virtual que tem em casa ou pergunta algo para o manual cognitivo do carro, ele está utilizando processamento de linguagem natural (PLN).

Em virtude de todas estas possibilidades e com a intenção de tornar seus produtos cada vez mais próximos dos consumidores finais, as empresas e organizações vêm inserindo PLN cada vez mais em suas mercadorias. Atualmente, as técnicas desse ramo podem ser encontradas em consultas a banco de dados, de forma que o usuário não precisa conhecer a estrutura, funcionamento e nem a linguagem a ser utilizada para realizar as *queries*; na criação de robôs virtuais, que podem comunicar-se com as pessoas para esclarecer dúvidas sobre alguns temas afins, entreter e ensinar (ex: Robô "Ed"); na confecção de *chatterbots*, tal como o *ChatterBot Doroty* e o *Sete Zoom*, que tem como função interagir com os internautas e simular uma conversação; entre outros (NETO; TONIN; PRIETCH, 2010).

Apesar da amplitude do conteúdo, todas as pesquisas desenvolvidas dentro deste assunto buscam um único propósito que, conforme Coppin (2013), seria ter um sistema com conhecimento de mundo o suficiente para ser capaz de envolver-se em uma discussão com humanos sobre qualquer assunto. Ao olhar para o contexto do presente projeto, observa-se que as três características da comunicação em que foca o PLN serão utilizadas, sendo elas: som (fonologia), estrutura (morfologia e sintaxe) e significado (semântica e pragmática). A parte do som será abordada em separado na Subseção 2.4.2. Estrutura e significado, por sua vez, serão fundamentais a partir do momento em que o aplicativo possuir a fala do usuário transcrita em texto. Isso porque, na sequência, será necessário tratá-la para viabilizar a busca pelos GIFs correspondentes às palavras-chave encontradas para, só então, poder exibi-los ao surdo.

2.4.2 Reconhecimento de fala

Devido à abrangência do campo descrito na Subseção 2.4.1, diversos estudiosos e pesquisadores classificam o reconhecimento de fala como apenas mais um tópico do mesmo. Como o objetivo deste trabalho não é entrar no mérito desta questão e, tendo em vista os casos de uso planejados para esta aplicação, os dois assuntos mostram-se equivalentes em nível de importância e indispensáveis para atingir o resultado final almejado. Desse modo, nada mais justo que destinar uma subseção deste documento para cada e abordar, individualmente, o contexto histórico, vantagens e incumbência dentro do projeto.

Engana-se quem pensa que o reconhecimento de fala é uma revolução tecnológica ou uma funcionalidade desenvolvida a partir dos anos 2000. Conforme Vieira (2016), os estudos pioneiros, que estabeleceram a relação entre as características do som e sua representação espectral, datam da década de 1930. Este episódio demonstra o longevo interesse dos cientistas em reproduzir, nas máquinas, os comportamentos humanos. Dentro destas condutas, a mais genuína é a necessidade de expressar-se, dado que a comunicação é a base para a interação social dos indivíduos e, para que esta aconteça, é necessária uma boa compreensão de fala (ZABONI; IORIO, 2009). À vista disso, percebe-se a importância do som na evolução dos membros da

sociedade - em todos os sentidos - e a utilização da voz como instrumento de sua realização.

No entanto, as pesquisas nesta área, historicamente, precisaram superar diversos obstáculos por conta das características únicas que rodeiam este tópico, tais como sotaques, regionalismos, gírias e homônimos (PEREIRA, 2009). Estas particularidades costumam estabelecer o grau de dificuldade do estudo e a complexidade dessas aplicações caracteriza-se ao longo de duas dimensões: o tamanho do vocabulário e a forma da elocução. Quanto maior o vocabulário, maior será a dificuldade de reconhecimento (ANDRADE et al., 2016). Posto isto, constata-se que o português, língua notoriamente sinuosa, apresenta um nível considerável de contratempos que devem ser suplantados, visando o êxito no desenvolvimento.

Apesar destas conhecidas dificuldades, a progressiva sofisticação dos processos de computação, acrescida do crescimento constante da capacidade dos dispositivos computacionais, propiciou que este setor da IA prosperasse grandemente, satisfazendo tarefas como o reconhecimento de fala ininterrupto e a capacidade de discernir uma voz em ambientes ruidosos, independente de quem seja o interlocutor (VIEIRA, 2016). Este avanço também tornou estas técnicas mais acessíveis pois, segundo Manfio (2017), o comando por voz evoluiu tanto nas últimas duas décadas que não são mais necessários equipamentos caríssimos e *softwares* dedicados para que ele seja possível. Logo, ao associar esta notável evolução técnica com os potenciais frutos positivos que a facilidade do uso da voz como interface pode trazer, vislumbra-se um cenário promissor, no qual sistemas dessa categoria estejam razoavelmente difundidos em poucos anos (TEVAH, 2006).

Sendo assim, incluir este mecanismo moderno e em pleno aperfeiçoamento na estrutura deste aplicativo, aumenta suas capacidades e torna-o suficientemente robusto para executar as tarefas pretendidas. Ademais, diante dos fatos expostos, evidencia-se que a comunidade surda, em razão de sua condição, não dispõe de uma das mais poderosas ferramentas de expressão individual. Semelhantemente, apesar de ser importante que todos ouvintes possuam um conhecimento básico da língua de sinais do seu país, é indevido privá-los de utilizar sua forma natural de conversação para comunicar-se com os sujeitos surdos. Baseado nesta concepção, este trabalho objetiva usufruir do ecossistema familiar para cada uma das partes e, dessa forma, permitir que a fala reconhecida seja transformada em Libras por meio dos GIFs.

2.5 Bibliotecas externas

Não há dúvidas de que cada sistema possui o seu propósito e entrega valor para aquele nicho de usuários em que atua. É por isso que a lógica de negócio é o ponto central de qualquer ferramenta, pois é através desta implementação customizada que a equipe de desenvolvimento consegue aplicar as premissas daquela atividade. Contudo, grande parte de uma aplicação diz respeito à adaptações para a plataforma em que ela será executada, escrita e leitura de arquivos, questões de segurança ou mesmo de acesso ao banco de dados. Ou seja, pontos técnicos que competem apenas ao desenvolvimento e que podem repetir-se dentro de um mesmo paradigma

de programação. Visando evitar retrabalho, estabelecer padrões e aumentar a produtividade, diversas organizações utilizam bibliotecas que auxiliam no processo de construção de *software*.

Uma biblioteca (ou *lib*) é um arquivo que contém um conjunto de funções. Existem diversos tipos de *libs* e com diversas finalidades mas, no geral, servem para facilitar a vida do desenvolvedor, permitindo que o desenvolvimento seja focado na regra de negócio e não em algumas funcionalidades (LOCAWEB, 2016). O mesmo conceito aplica-se para a utilização de *frameworks*. Com esse pensamento, no contexto do presente projeto, o importante é focar em reconhecer a fala, e não em como fazer esse reconhecimento; em adaptar os textos reconhecidos para sinais em Libras, e não em como adaptar; estes exemplos reforçam a ideia de que se um aplicativo - seja da área de saúde ou de finanças - precisa emitir áudio, o meio com que será realizada essa emissão pode ser o mesmo para ambos.

Consequentemente, se existir um pacote externo focado em uma funcionalidade específica e que execute-a com perfeição, é interessante utilizá-lo em detrimento de recriar todo seu comportamento. E foi baseado nesta concepção que, para criar este aplicativo, utilizou-se diversas bibliotecas de terceiros. Ademais, esta utilização coincide com duas características deste trabalho: primeiramente, por ser uma POC, o objetivo, de fato, é *provar* que a solução é viável; em segundo lugar, o próprio *framework* escolhido - React Native - foi concebido justamente para adaptar-se muito bem à inserção de *plugins*. À visto de tudo isso, esta Seção será utilizada para listar e conceituar, rapidamente, as ferramentas que viabilizaram a elaboração desta aplicação móvel.

2.5.1 UI Kitten

UI Kitten é uma implementação, em React Native, do Eva Design System. Ele contém um conjunto de componentes de UI de uso geral estilizados de maneira semelhante (padronizados). O desenvolvedor, com isso, pode focar na lógica de negócios e o Kitten cuida da aparência visual. E a coisa mais incrível: os temas podem ser alterados em tempo de execução, sem a necessidade de recarregar o aplicativo (AKVEO, 2019, tradução do autor). Com quase 20 componentes de propósito geral prontos para uso e uma documentação clara de suas configurações, esta biblioteca foi fundamental para a construção de um aplicativo com visual moderno e agradável. Ademais, os arquivos originais cedidos para prototipação e o aplicativo *Kitten Tricks* - disponível na *Google Play* e na *App Store* - permitiram testes e validações antes mesmo de codificar, fato que, sem dúvida, minimizou as chances de erro durante o desenvolvimento.

`Documentação:` https://akveo.github.io/react-native-ui-kitten/

2.5.2 React Native TTS

React Native TTS é uma biblioteca de texto para voz para React Native, compatível com sistemas operacionais iOS e Android (TTS, 2020, tradução do autor). Esse pacote permite que um texto - independente da sua origem - seja emitido pelos alto-falantes do dispositivo

em uso. Para viabilizar essa funcionalidade, ele reconhece os mecanismos nativos (as vozes) disponíveis no aparelho (tal como a voz do Google no Android, por exemplo) e permite sua utilização mesmo sem conexão com a internet (IDASZAK, 2020, tradução do autor). Além disso, a biblioteca manipula estes mecanismos de modo a expor suas configurações, permitindo que o desenvolvedor defina seu funcionamento através de parâmetros, tal como a velocidade e o volume com que o texto será pronunciado.

`Documentação:` https://www.npmjs.com/package/react-native-tts

2.5.3 React Native Fast Image

O componente `Image` do React Native gerencia o cache de imagens como a maioria dos *browsers*. Então, se o servidor estiver retornando os cabeçalhos de controle de cache apropriado para imagens, será possível obter o mesmo comportamento de cache embutido dos navegadores. Contudo, frequentemente é possível notar: perdas de cache, carregamento de baixo desempenho do cache e baixo desempenho em geral. O componente `FastImage` é uma substituição de imagem que resolve esses problemas. Funciona como um *wrapper* para o `SDWebIMage` (iOS) e para o `Glide` (Android) (IMAGE, 2020, tradução do autor). Em resumo, utilizar a abstração criada por esta biblioteca potencializa (muito) a performance do aplicativo, uma vez que ela é focada em gerenciar o cache das imagens e evitar consultas desnecessárias, além de viabilizar a utilização de GIFs (que não são suportados nativamente pelo React Native).

`Documentação:` https://www.npmjs.com/package/react-native-fast-image

2.5.4 React Native Snackbar

Snackbars são usados para exibir uma breve mensagem para o usuário, junto com um botão de ação opcional. Geralmente, eles são animados de baixo para cima na parte inferior da tela e desaparecem logo em seguida (SNACKBAR, 2020, tradução do autor). Essa biblioteca fornece uma reprodução do componente `Snackbar` do *Material Design* (*design system* criado pelo Google). Este tipo de componente costuma ser muito utilizado para fornecer *feedback* de sucesso ou falha para o usuário em formulários ou ações definitivas.

`Documentação:` https://www.npmjs.com/package/react-native-snackbar

2.5.5 React Native Community - Async Storage

Async Storage é um sistema de armazenamento de chave-valor assíncrono, não criptografado e persistente para React Native, compatível com Android, iOS, Web, MacOS e Windows (STORAGE, 2020, tradução do autor). Essa biblioteca, responsável pela armazenagem de dados local, é frequentemente usada para salvar informações pessoais ou descartáveis do usuário, tal como o tema da aplicação. Dessa forma, esses dados permanecem apenas no dispositivo. No que refere-se ao desenvolvimento, possibilita operações básicas como salvar, ler e remover através de uma API simples - e, no caso do React Native, através do *hook* `useAsyncStorage`.

`Documentação:` https://react-native-async-storage.github.io/async-storage/

2.5.6 React Native Community - Voice

Voice é uma biblioteca para conversão de voz em texto para React Native (VOICE, 2020, tradução do autor). O componente fornecido por este pacote viabiliza o reconhecimento de voz porque possui uma série de eventos que podem ser usados para iniciar ou parar essa recognição, além de fornecer o status da mesma (JABBAR, 2020, tradução do autor). Na primeira vez que a funcionalidade for ativada, a biblioteca, automaticamente, solicita a permissão do usuário para utilizar o microfone do aparelho. Também é nesse momento que o desenvolver pode determinar qual a linguagem que será reconhecida (como *pt-BR*, por exemplo).

`Documentação:` https://www.npmjs.com/package/@react-native-community/voice

2.5.7 React Navigation

Nos navegadores *web*, o programador pode apontar páginas diferentes usando uma *tag* âncora (`<a>`). Quando o usuário clica em uma dessas âncoras, o endereço (URL[2]) é adicionado na pilha de histórico do *browser*. Caso o usuário pressione o botão para voltar, o navegador remove o item do topo desta pilha, de forma que a página ativa agora é a página visitada anteriormente. O React Native não possui esse tipo de funcionalidade de histórico global embutido nativamente no *framework*, por isso o React Navigation mostra-se relevante (NAVIGATION, 2020, tradução do autor). Esta biblioteca providencia diversas opções de navegação - *stack*, *tab*, *drawer* - que são responsáveis por gerenciar as telas do aplicativo. Outro ponto importante é que o UI Kitten - descrito na Seção 2.5.1 - possui componentes específicos para facilitar a integração com este pacote de gerenciamento de rotas.

`Documentação:` https://reactnavigation.org/

2.5.8 AWS SDK para JavaScript

O AWS SDK para JavaScript, como o próprio nome sugere, providencia uma API JavaScript para os serviços da Amazon Web Services (AWS). Dessa forma, é possível utilizar esta API para construir bibliotecas ou aplicações para dispositivos móveis, *browsers* ou para o Node.js (AWS, 2020d, tradução do autor). Tal SDK ajuda a eliminar as complexidades de codificação ao fornecer métodos do JavaScript para vários dos serviços da plataforma, incluindo Amazon S3, Amazon EC2, DynamoDB e Amazon SWF (AWS, 2020c). Esse kit de desenvolvimento assume um papel de facilitador para acesso aos serviços em *cloud* da Amazon e, após a configuração das credenciais de acesso da conta, utilizar qualquer recurso dessa gama de opções passa a ser tão simples quanto conectar em um banco de dados local.

`Documentação:` https://aws.amazon.com/pt/sdk-for-node-js/

[2]URL é o endereço de um recurso na rede como a internet. Em inglês, é a sigla de *Uniform Resource Locator*, e em português significa Localizador Padrão de Recursos (GONÇALVES, 2020).

2.5.9 Boto3

Boto é o SDK da Amazon Web Services para Python. Ele permite que os desenvolvedores dessa linguagem criem, configurem e gerenciem serviços AWS, como o EC2 e o S3. Boto fornece uma API orientada a objetos fácil de usar, bem como acesso detalhado aos recursos da plataforma (BOTO3, 2020, tradução do autor). Sem mais, esta biblioteca desempenha o mesmo papel que a anterior: facilitar a utilização das ferramentas AWS por parte do desenvolvedor. A única diferença entre as duas é que, enquanto a antecedente limita-se a suportar a linguagem JavaScript, esta concentra-se nas implementações em Python.

`Documentação:` https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

2.5.10 Amazon DynamoDB

O Amazon DynamoDB é um banco de dados de chave-valor e documento NoSQL (não relacional) que oferece um desempenho de milissegundos com um dígito em qualquer escala (AWS, 2020a). Um dos atributos mais interessantes desta plataforma - por apresentar uma estrutura não relacional - é a flexibilidade, uma vez que não é preciso definir quais são as colunas e o tipo de cada uma no momento da concepção da tabela. Isso porque a própria ferramenta reconhece o tipo de dado que será inserido, estabelecendo automaticamente sua categoria. Além dos demais benefícios já conhecidos do *cloud* - não precisar instalar nada localmente, definições de segurança já definidas, entre outras - este tipo de característica é muito relevante para projetos iniciantes que, assim como este trabalho, estão suscetíveis a mudar alguma regra de negócio e não desejam recriar ou atualizar todos registros da sua base de dados por causa disso.

`Documentação:` https://docs.aws.amazon.com/pt_br/dynamodb

2.5.11 Amazon S3

O Amazon Simple Storage Service (Amazon S3) é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados e segurança. Ele armazena dados como objetos dentro de *buckets*, no qual um objeto é um arquivo e todos os metadados opcionais que o descrevem (AWS, 2020b). Já que esta plataforma concentra-se no tratamento de objetos, ela é capaz de disponibilizar otimizações para a leitura e escrita dos mesmos, além de recursos de controle para gerenciar suas permissões de acesso. Dentre essas permissões, uma das possibilidades é definir a visibilidade de um objeto como *pública*, permitindo que qualquer usuário da internet possa acessá-lo através do *link* correto. Essa funcionalidade é proveitosa, por exemplo, para *sites* ou *apps* que precisam exibir uma imagem através de uma URL válida na rede, bastando realizar o *upload* do arquivo e seguir esta estratégia. Também, tendo em vista que a ferramenta replica o comportamento de um *filesystem*, é possível separar os documentos em diretórios virtuais, que favorecem a organização.

`Documentação:` https://docs.aws.amazon.com/pt_br/s3

2.5.12 spaCy

spaCy é uma biblioteca grátis e de código aberto para processamento de linguagem natural em Python. Esta ferramenta foi projetada especificamente para uso em produção e ajuda a criar aplicativos que processam e "entendem" grandes volumes de texto. Ela pode ser usada para construir sistemas de extração de informações ou de compreensão de uma língua natural, além de pré-processar textos para *deep learning* (SPACY, 2020, tradução do autor). Sobre a comparação com outras excelentes ferramentas do mercado para esse propósito, a própria documentação deixa claro que o spaCy apresenta decisões de *design* bastante diferentes do NLTK ou CoreNLP, por exemplo, que foram criados como plataformas de ensino e pesquisa. Isso porque esta ferramenta é integrada e opinativa e tenta evitar que o usuário precise escolher entre vários algoritmos que oferecem funcionalidades equivalentes entre si. Entre as funcionalidades oferecidas pela *lib*, estão: tokenização, lematização, classificação de texto, entre outras.

`Documentação:` https://spacy.io/

2.5.13 FastAPI

FastAPI é um *framework web* moderno e rápido (de alto desempenho) para a construção de APIs RESTful com Python 3.6+ (FASTAPI, 2020, tradução do autor). Esta biblioteca, leve e muito bem documentada, é uma ótima opção para criar rotas que irão funcionar no modelo *request* e *response*. Possui um utilitário de linha de comanda eficiente, no qual, através de alguns parâmetros, é possível criar a aplicação e fazer as configurações necessárias. Também possui, nativamente, documentações interativas que são criadas automaticamente com base nas rotas disponibilizadas. Entretanto, o detalhe mais importante é que esta *lib* utiliza Python como linguagem, sendo, desta forma, compatível com o spaCy - descrito na Seção anterior.

`Documentação:` https://fastapi.tiangolo.com/

2.5.14 FFMPY

FFMPY é um *wrapper* simplista da linha de comando FFmpeg. Ele implementa uma interface Pythonic para a compilação da linha de comando e usa o módulo `subprocess` do Python para executá-la após compilada (FFMPY, 2020, tradução do autor). O FFmpeg, por sua vez, é uma solução completa para gravar, converter e transmitir áudio e vídeo (FFMPEG, 2019, tradução do autor). Dessa forma, esta ferramenta é muito útil quando é necessário transformar um recurso que está em determinado formato para outro - no cenário do presente projeto, um vídeo MP4 em GIF (detalhado na Seção 4.3).

`Documentação:` https://pypi.org/project/ffmpy/

2.5.15 Docusaurus

Docusaurus é um gerador de *site* estático de alto desempenho e pode ser usado para criar *sites* baseados em conteúdo comum (por exemplo: documentação, blogs, páginas de des-

tino e *marketing* de produtos, entre outros) com extrema rapidez (DOCUSAURUS, 2020, tradução do autor). Esta ferramenta é utilizada por várias bibliotecas para produzir a documentação pública de suas implementações - o Async Storage, descrito na Subseção 2.5.5, é um exemplo. Apesar da maioria dos casos de uso apresentar uma documentação mais técnica, o teor da documentação depende de quem escreve-a (a plataforma não interfere nisso). Portanto, por suportar Markdown, ser construída utilizando React e possuir versionamento da documentação, também é considerada uma excelente alternativa para criar documentações focadas no usuário final.

`Documentação:` https://v2.docusaurus.io/

2.5.16 Síntese

Como citado no início deste Subcapítulo, todas as ferramentas descritas anteriormente concederam o suporte necessário para a construção desta aplicação móvel. Notoriamente, cada uma, em sua individualidade, desempenha papel crucial no funcionamento completo da solução, desde o processamento das informações e conexões com os serviços da AWS - realizados pela API - até a interação final com o usuário - responsabilidade do aplicativo. Inclusive, graças às especialidades fornecidas por tais bibliotecas, foi possível obter um bom desempenho em todas as funcionalidades suportadas.

Considerando o panorama da API, optar por desenvolvê-la na linguagem de programação Python, apoiada no *framework* FastAPI, destravou uma série de alternativas para a utilização de inteligência artificial. Além disso, esta escolha tornou-a compatível - com certa facilidade, inclusive - com várias soluções robustas de processamento de linguagem natural, como é o caso da spaCy, que foi selecionada para realizar esta função neste trabalho. Já no ponto de vista do aplicativo, utilizar o *framework* React Native - uma variação do React que, por sua vez, é direcionado para a linguagem de programação JavaScript - permitiu usufruir de diversas contribuições da comunidade desta tecnologia que, ao notar uma necessidade em comum aos desenvolvedores, frequentemente cria pacotes genéricos que servirão para muitos projetos.

Os melhores exemplos desta contribuição, no contexto desta aplicação móvel, são o Voice e o React Native TTS, que providenciaram formas de incluir, respectivamente, os dois recursos críticos para alcançar o objetivo tecnológico inicial desta atividade: transformar fala em texto e texto em voz. Por fim, é interessante salientar a relevância do Docusaurus. Este gerador de *site* permitiu que todos os conceitos essencialmente técnicos descritos neste documento, fossem abstraídos em imagens animadas, ícones e figuras e disponibilizados, de forma *online*, em um manual com endereço válido na internet. Sendo assim, qualquer usuário do aplicativo poderá navegar por este guia e, rapidamente, ter uma visão completa do resultado desta combinação composta por, pelo menos, quinze bibliotecas ou ferramentas que, dentro de suas finalidades, contribuíram para o propósito desta aplicação.

3 MATERIAIS E MÉTODOS

Ao analisar o momento atual da humanidade, é possível observar a ocorrência de um processo irreversível pelo qual a mesma tem passado, no sentido de tornar-se cada vez mais globalizada. Em virtude disso, os termos *integração* e *inclusão* têm sido muito utilizados - mesmo que, por vezes, de maneira confusa (SONZA et al., 2013). Harmonizando-se com essa propensão inclusiva, o ramo de desenvolvimento de *software* também tem buscado criar soluções notáveis no âmbito da acessibilidade.

Sendo assim, é possível identificar a existência ou o surgimento de aplicativos focados em tratar do tópico *integração da comunidade surda*. Como forma de nortear possíveis evoluções no presente projeto e, também, de reconhecer funcionalidades cruciais que já estão disponíveis na aplicação móvel **DeafEnd**, este Capítulo irá detalhar três soluções que vêm destacando-se neste nicho de atuação. Além desta relevância, as ferramentas que serão abordadas nos subcapítulos seguintes possuem características similares às embutidas no aplicativo criado neste trabalho e, por isso, servem como referência em um processo de *benchmark*[1].

3.1 Hand Talk Tradutor para Libras

O aplicativo Hand Talk, com mais de 1.000.000 de instalações, é o mais conhecido dentre os abordados neste Capítulo. Utilizando-se de intérpretes 3D - denominados como Hugo e Maya, em suas versões masculina e feminina, respectivamente - ele traduz automaticamente textos e áudios para a Língua Brasileira de Sinais (Libras) e para a Língua Americana de Sinais (ASL) por meio de inteligência artificial (TALK, 2020). Além de poder optar pela língua desejada, o usuário também pode definir a velocidade em que o intérprete deve executar a sinalização, alternando entre as opções *lenta*, *normal* e *máxima*.

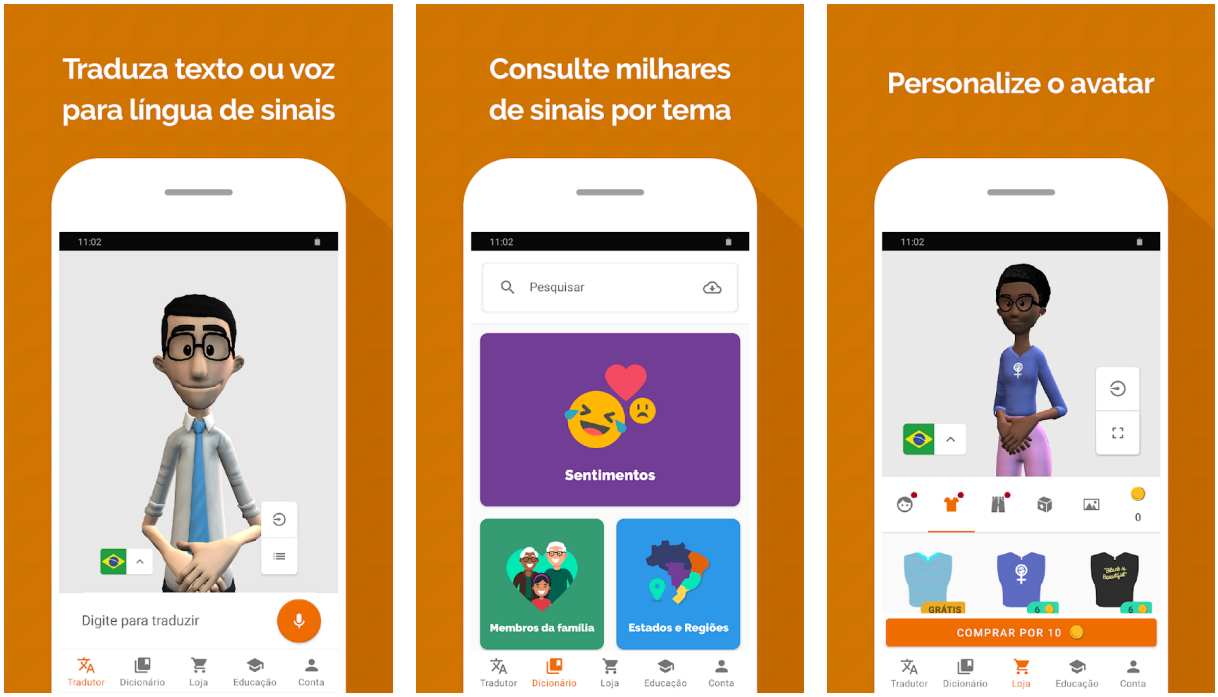
Um dos pontos interessantes da solução é o menu *Dicionário*, que agrupa os sinais em assuntos e permite que o utilizador pesquise - em Português - por uma palavra específica. Essa opção é muito útil, pois permite visualizar, em um único espaço, todos sinais disponíveis. Ao selecionar qualquer termo, o intérprete sinaliza-o automaticamente. O aplicativo é gratuito, mas algumas customizações passíveis de serem aplicadas no tradutor ou no cenário, precisam ser adquiridas com uma moeda própria da plataforma - que pode ser adquirida em lotes que variam entre R$ 9,90 e R$ 139,90. Também, algumas funções passam a estar acessíveis somente caso o usuário crie ou possua uma conta.

A ferramenta, que está disponível nos dispositivos com sistemas operacionais Android e iOS, possui uma interface simples e bastante amigável - apesar da poluição visual causada pelas propagandas e anúncios. Por último, a funcionalidade de exportar qualquer sinal escolhido

[1] *Benchmark*, palavra derivada do inglês, significa comparação e, neste contexto, indica uma verificação feita entre produtos, serviços ou processos (LINS, 2019).

para uma imagem animada (*GIF*) é um ponto adicional bastante interessante.

Figura 9 – Interface do aplicativo: Hand Talk Tradutor para Libras



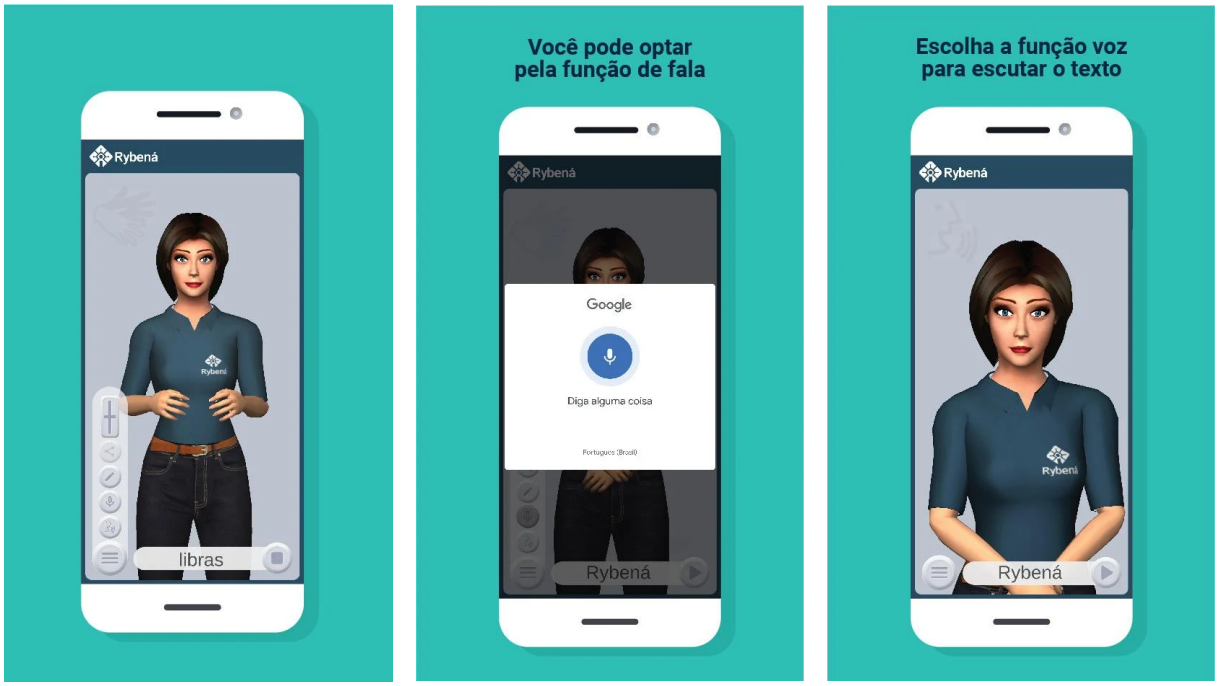
Fonte: Talk (2020)

3.2 Rybená Tradutor Libras Voz

O Rybená, por sua vez, apresenta uma interface mais simples quando comparado ao Hand Talk. Apesar de seu visual mais antigo - e um pouco confuso - diversas funcionalidades também estão presentes na solução, tal como reconhecimento de fala, tradução de Português para Libras e controle de velocidade da execução da sinalização. Falta, no entanto, um local centralizado para apresentar todos os sinais disponíveis na plataforma, permitindo uma visualização mais geral e uma pesquisa mais específica - para encontrar um determinado termo, por exemplo.

Semelhante à opção de exportação citada no fim do Subcapítulo 3.1, esta ferramenta também oferece um meio de compartilhar um sinal escolhido, com a diferença de gerar um vídeo e não um *GIF*. Contudo, seu principal diferencial para os demais fica por conta da interação com voz. Isto é, ao trocar o *modo* do *app*, o mesmo permite que a frase de entrada - na língua portuguesa - seja emitida nos alto-falantes do dispositivo ao invés de ser traduzida pela intérprete 3D. Assim, surdos e pessoas com deficiências intelectuais, disléxicos e outros com dificuldades de leitura podem interagir com outros cidadãos. É pertinente ressaltar que esta tecnologia é 100% brasileira e desenvolvida por surdos (ICTS, 2020a). O *software* também possui versões para os aparelhos com sistemas operacionais Android e iOS.

Figura 10 – Interface do aplicativo: Rybená Tradutor Libras Voz



Fonte: ICTS (2020b)

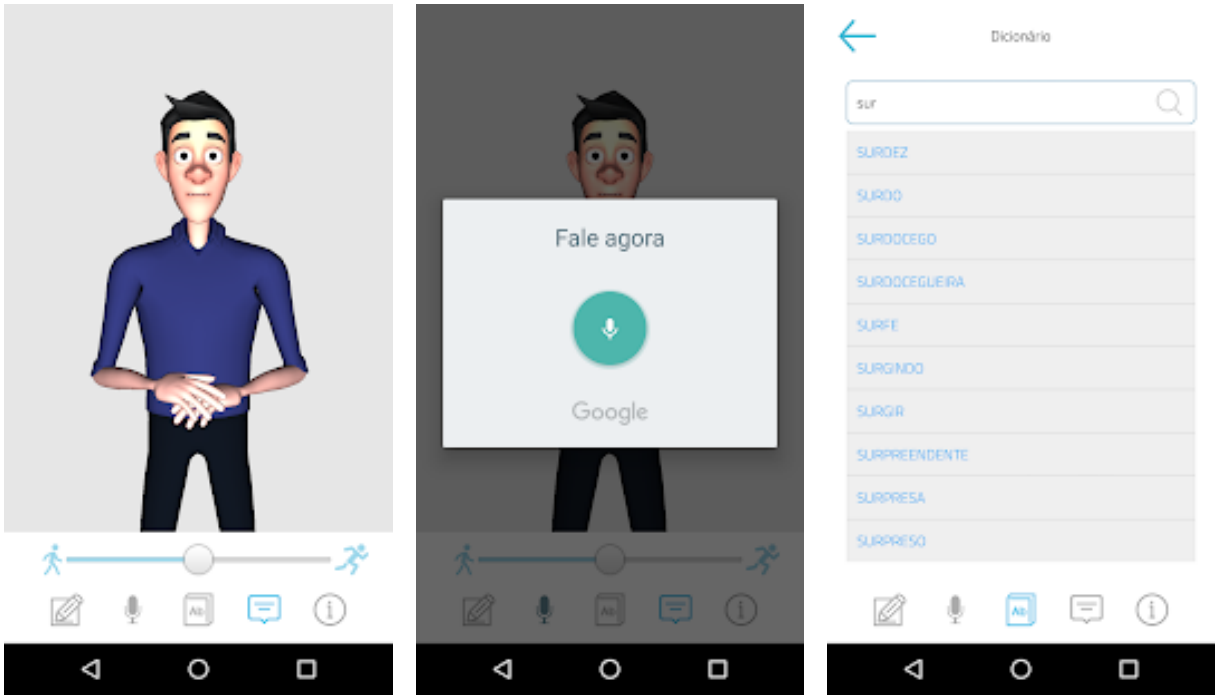
3.3 VLibras

Mesmo distinguindo-se como o mais simples dos três, o aplicativo VLibras contém a funcionalidade essencial de converter Português para Libras. Também, ao contrário do Rybená, possui um menu *Dicionário* que lista todas as palavras que possuem sinal cadastrado na ferramenta, acrescido de um campo de texto que filtra termos específicos. Ainda, permite o controle da velocidade da execução da sinalização e possui a opção de exportar um sinal - em teoria, pois apresentou problema quando esta função foi testada durante a criação deste documento.

Seu ponto negativo evidencia-se no âmbito da experiência do usuário. Isso porque, ao utilizar a funcionalidade de reconhecimento de fala, o aplicativo não traduz o que foi discernido automaticamente. Diferentemente dos demais, o *app* simplesmente define a frase identificada como o valor do campo que é utilizado para traduzir sentenças digitadas manualmente, e acaba impondo a realização de dois cliques extras para que a tradução, finalmente, seja executada.

Todavia, um detalhe expressivo que realça esta plataforma é que o VLibras não trata-se apenas de uma, mas sim de um conjunto de ferramentas computacionais de código aberto (VLIBRAS, 2016). Dessa forma, qualquer entusiasta da área tem a possibilidade de, partindo de uma base robusta, realizar os incrementos ou alterações que julgar necessárias. Além de tudo, a solução também conta com dois intérpretes 3D - versão masculina e feminina - e está disponível nos dispositivos com sistemas operacionais Android e iOS.

Figura 11 – Interface do aplicativo: VLibras



Fonte: VLibras (2020)

3.4 Quadro comparativo

Objetivando simplificar o entendimento de todas as características descritas separadamente nos subcapítulos anteriores, criou-se o Quadro 3 para compilar as funcionalidades que estão presentes por aplicativo. Os apontamentos realizados foram feitos com base na leitura da descrição dos *apps*, buscas na internet e utilização própria (executada durante a criação deste documento).

Quadro 3 – Comparativo entre aplicativos.

	Hand Talk	**Rybená**	**VLibras**	**DeafEnd**
Última atualização	26.10.2020	27.07.2020	13.07.2020	10.11.2020
Ambiente *web*	✔	✔	✔	✔
Documentação de usuário	✖	✖	✔	✔
Reconhecimento de fala automático	✔	✔	✖	✔
Português para Libras	✔	✔	✔	✔
Libras para Português	✖	✖	✖	✔
Inglês para ASL	✔	✖	✖	✖
Customizável	✔	✖	✔	✔

Fonte: Autor.

Na sequência, cada uma das linhas do Quadro 3 será brevemente clarificada, com o intuito de expor alguns detalhes adicionais que fazem-se relevantes:

- **Última atualização**: dado extraído da *Google Play* no dia 29 de outubro de 2020 e retrata quão atuais são as ferramentas citadas.
- **Ambiente *web***: considerou-se como "ambiente" qualquer *site* ou portfólio que sirva como boas-vindas a um indivíduo que queira saber mais sobre a solução. Segue, portanto, a lista de endereços *online* considerados na comparação:
  - Hand Talk: https://handtalk.me/br
  - Rybená: https://portal.rybena.com.br/site-rybena/
  - VLibras: https://www.vlibras.gov.br/
  - DeafEnd: https://deafend-documentation.vercel.app/
- **Documentação do usuário**: buscava-se exemplos sequenciais que demonstrassem ao usuário como utilizar a ferramenta. Apenas o aplicativo **DeafEnd** possui algo nesse sentido - que, inclusive, está incorporado em seu ambiente *web*. Porém, o *software* VLibras disponibiliza manuais[2] que orientam o uso da solução nos sistemas operacionais Windows e Linux e nos navegadores. Logo, apesar de não ser específico para o *app*, este conteúdo mostra-se uma boa fonte pesquisa e também cumpre o propósito desta característica.
  - É importante ressaltar que os aplicativos Rybená e Hand Talk possuem, respectivamente, um vídeo[3] sobre as formas de interagir com a solução e uma completa documentação[4] para desenvolvedores embutirem o *plugin* em seu *site*. No entanto, estes materiais não correspondem às condições definidas e, em virtude disso, esta característica não foi identificada no Quadro 3 para nenhum dos dois *softwares*.
- **Reconhecimento de fala automático**: função que viabiliza ao usuário acessar um determinado local do aplicativo, proferir uma frase da língua portuguesa e, sem nenhuma outra interação, visualizar os sinais correspondentes ao texto pronunciado.
- **Português para Libras**: representa o conceito de que, através do teclado ou da fala, o aplicativo reconhece os dados de entrada na língua portuguesa e apresenta os sinais relacionados na língua brasileira de sinais.
- **Libras para Português**: ao contrário do item anterior, significa que o *software* aceita ou oportuniza que os dados de entrada sejam fornecidos em Libras e, na sequência, sejam expostos através dos alto-falantes do dispositivo em Português.
- **Inglês para ASL**: mesma definição da conversão Português para Libras, apenas substituindo a língua portuguesa pela inglesa e os sinais da língua brasileira pelos da língua americana de sinais.
- **Customizável**: providenciar formas para que o usuário sinta-se, de fato, como dono do *app*. Foram consideradas funcionalidades como alteração de tema (claro e escuro), criação de listas ou botões de acordo com o desejo do utilizador e opções de intérprete masculino e feminino.

[2] *Link* do manual - Windows: https://vlibras.gov.br/doc/windows/introduction/presentation.html
[3] *Link* do vídeo: https://portal.rybena.com.br/site-rybena/sobre.html
[4] *Link* da documentação: https://docs.handtalk.me/br/4/introducao

3.4.1 Constatações coletadas

Examinar aplicações móveis específicas da área de atuação do presente projeto, com boas avaliações nas lojas de aplicativos e com um considerável tempo de existência, mostrou-se uma tarefa fundamental para ser realizada durante o período de planejamento deste trabalho. Ao utilizá-los, foi possível extrair impressões sobre as funcionalidades apresentadas e possíveis carências percebidas. Uma característica comum nos três *softwares* utilizados na comparação foi que todos utilizam intérpretes 3D criados a partir de computação gráfica. Esta abordagem permite que os desenvolvedores controlem, por exemplo, a velocidade com que os sinais serão reproduzidos e apresentem mais detalhes faciais durante a sinalização. Contudo, deduziu-se que utilizar uma animação possui algumas desvantagens, tais como o distanciamento da realidade - um intérprete humano transparece mais naturalidade - e as adversidades de codificação[5].

Seguramente, todos os atributos positivos encontrados nesta análise comparativa influenciaram as decisões arquiteturais do aplicativo **DeafEnd**, enquanto pontos considerados como passíveis de melhorias serviram de incentivo à busca por caminhos melhores para entregar excelentes resultados. Seguindo esta metodologia, tornou-se viável incluir na aplicação móvel desenvolvida a grande maioria das funções listadas no Quadro 3 - com as customizações consideradas necessárias ou válidas - exceto o suporte à língua americana. Além disso, é importante salientar que o aplicativo **DeafEnd**, apesar de ser a mais recente das ferramentas citadas, é o único que contempla o *input* de informações em Libras e o *output* em Português. Esta possibilidade foi, desde o início desta tarefa, considerada como uma das prioridades, uma vez que o objetivo estipulado é facilitar a comunicação entre a comunidade surda e a ouvinte e, portanto, não seria possível alcançá-lo disponibilizando apenas uma via da conversação.

Dito isso, ao iniciar a codificação do aplicativo, cogitou-se a possibilidade de usufruir do código-fonte aberto disponibilizado pelo projeto VLibras. Porém, ao investigar o repositório encontrado e suas atividades recentes, constatou-se que a última atualização programática ocorreu 3 anos atrás e a última interação pública (uma *issue*) foi efetuada há 2 anos - ainda sem solução (BRASILEIRO, 2019). Em razão disso, considerou-se incerto e um tanto quanto limitante utilizar esta opção como base da aplicação móvel, optando-se, portanto, em criar uma completamente nova. O Capítulo 4 descreve todo o processo de construção que culminou na entrega de um aplicativo que, apesar de não ser tão robusto, apresenta as mesmas *features* - e, talvez, o mesmo potencial - de *softwares* vistos como referências do ramo.

[5]Não são todas as tecnologias que permitem a movimentação autêntica de um boneco virtual, o que significa que, além das opções limitadas para utilizar no desenvolvimento, este tipo de atividade costumar exigir conhecimento de técnicas bastante específicas.

4 DESENVOLVIMENTO

Com o suporte fornecido pela existência de tantas ferramentas - como as citadas no Subcapítulo 2.5 - capazes de suprir as necessidades de um *software* novo, é possível devanear sobre as melhores funcionalidades que poderiam ser incluídas na aplicação. Ocorre que iniciar a construção de um sistema apenas com a visão macro de suas necessidades, costuma ser um erro gravíssimo - e nem tão raro. Como consequência, é comum presenciar cenas em times de desenvolvimento onde, por exemplo, o desenvolvedor recebe uma especificação para trabalhar e, enquanto está codificando, percebe que algum detalhe passou despercebido pelo processo de análise de requisito. Isto gera uma dúvida que o impede de prosseguir seu trabalho com segurança e deflagra que, a partir deste momento, o desenvolvedor não tem certeza do que exatamente deve ser feito (WILDT et al., 2015).

Pensando em evitar este tipo de situação e levantar minuciosamente as informações necessárias para construir uma solução tecnológica de qualidade, o presente projeto foi desenvolvido em 3 etapas distintas, que são: levantamento de requisitos, prototipação e implementação do aplicativo. Nos subcapítulos seguintes que, respectivamente, dissertarão sobre cada uma destas fases, será possível encontrar desde o planejamento dos cenários contemplados na aplicação móvel, até o mapeamento das exigências técnicas e conceituais que, obrigatoriamente, precisariam ser atendidas para o êxito do trabalho.

4.1 Levantamento de requisitos

Requisitos são capacidades e condições às quais o sistema - e em termos mais amplos, o projeto - deve atender (JACOBSON; BOOCH; RUMBAUGH, 1999, tradução do autor). A investigação para compreender um problema aplicado ao desenvolvimento de *software* e conseguir identificar, da forma mais assertiva possível, tais requisitos é conhecida como levantamento ou elicitação de requisitos (BEZERRA, 2015). De forma resumida, o desafio básico deste processo é encontrar, comunicar e lembrar (o que geralmente significa registrar) o que é realmente necessário, expressando isso de forma clara para o cliente e os membros da equipe de desenvolvimento (LARMAN, 2007).

À medida que os requisitos são reunidos, inicia-se a materialização de uma visão mais completa das funções e características do sistema a ser desenvolvido. Entretanto, continua sendo difícil migrar para a execução das técnicas de engenharia de *software* até que entenda-se como tais funções serão utilizadas por diferentes classes de usuários (PRESSMAN; MAXIM, 2016). Para solucionar essa questão, ainda segundo Pressman e Maxim (2016), é possível que desenvolvedores e usuários criem um conjunto de cenários - comumente chamados de *casos de uso* - que identifiquem um roteiro de uso da ferramenta.

Desde que foi aprovada como padrão pelo Grupo de Gerenciamento de Objetos[1] (*Object Management Group*) em 1997, a UML tem tido grande aceitação pela comunidade de programadores (BEZERRA, 2015). Esta sigla, que significa Linguagem de Modelagem Unificada (*Unified Modeling Language*), representa uma família de notações gráficas que auxilia na descrição e no projeto de sistemas de *software*, principalmente daqueles construídos utilizando o estilo orientado a objetos (FOWLER, 2005).

Apesar deste não ser o paradigma de construção usado neste aplicativo, seguiu-se o padrão UML para criar os diagramas que auxiliaram nas definições estratégicas e representam a essência do presente projeto. Exceto os contidos na Subseção 4.1.5, que sofreram pequenas adaptações para que fosse possível simbolizar a composição das ferramentas e *frameworks* utilizados de acordo com a forma convencional, todos os desenhos que serão expostos nas seções seguintes foram produzidos com o suporte do programa *Astah UML*.

4.1.1 Definição dos atores

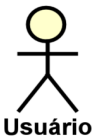
Conforme explica Bezerra (2015), na terminologia da UML, qualquer elemento externo ao sistema que interage com o mesmo é, por definição, denominado ator. Esse agente estimula, solicita ações, dispara eventos ou recebe reações do programa e, por isso, representa a forma pela qual um *software* percebe o seu ambiente. Um único ator pode realizar muitos casos de uso e, inversamente, um caso de uso pode ter vários atores executando-o (FOWLER, 2005).

Levando em consideração que um ator pode ser um usuário, um dispositivo ou, até mesmo, outro sistema, é importante responder algumas perguntas essenciais no momento de identificar possíveis atores da aplicação, tais como:

- Quem usa o sistema?
- Quem inicia o processo?
- Quem fornece os dados ou recebe as informações?

Após analisar as necessidades do projeto e responder às perguntas anteriores, foi possível identificar um único usuário nos cenários de interação com esta aplicação móvel. De forma prática, na maior parte das vezes este utilizador será o sujeito surdo, já que o aplicativo foi pensado para adaptar-se à sua realidade. Contudo, nada impede que um sujeito ouvinte também utilize-o como facilitador de conexão com a comunidade surda. Desse modo, a Figura 12 demonstra de que forma esta classe de usuários será representada nos demais diagramas.

Figura 12 – Ator do aplicativo



Fonte: Autor.

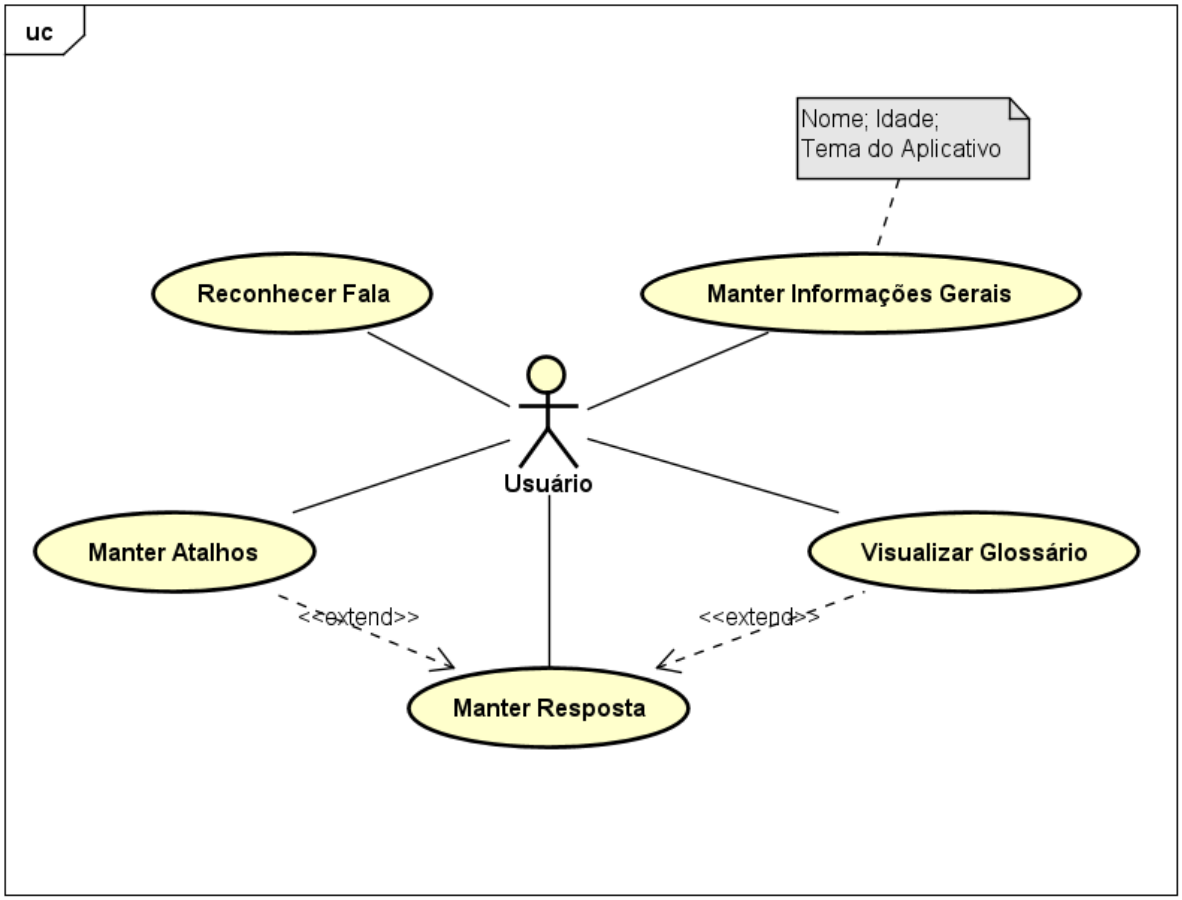
[1] O OMG é um consórcio internacional de empresa que define e ratifica padrões na área da orientação a objetos.

4.1.2 Diagrama de caso de uso

Um caso de uso deve descrever uma jornada estilizada sobre como um usuário (desempenhando um de uma série de papéis possíveis) interage com o sistema sob um conjunto de circunstâncias específicas. Dessa forma, um caso de uso representa o *software* ou o sistema do ponto de vista do usuário (PRESSMAN; MAXIM, 2016). Tendo em vista que o presente projeto conta com apenas uma classe de usuários identificada, decidiu-se criar um caso de uso geral. Logo, ao invés de montar diagramas específicos sobre as várias possibilidades de interação com o aplicativo, optou-se por produzir uma representação mais abrangente dos seus módulos.

Sendo assim, na Figura 13 estão dispostas as seguintes conjunturas: *Manter atalhos*; *Reconhecer fala*; *Manter informações gerais*; *Manter resposta* e *Visualizar glossário*. Todas estas funcionalidades estão fragmentadas ao longo das telas da aplicação e serão explanadas, com suas particularidades, no Subcapítulo 4.3. Além disso, cada um destes cenários contém um diagrama de atividade e de sequência exclusivo para complementar o entendimento do fluxo de execução. Por fim, é válido ressaltar que a descrição dos principais casos de uso foram inseridas no apêndice A.

Figura 13 – Diagrama de caso de uso: Geral



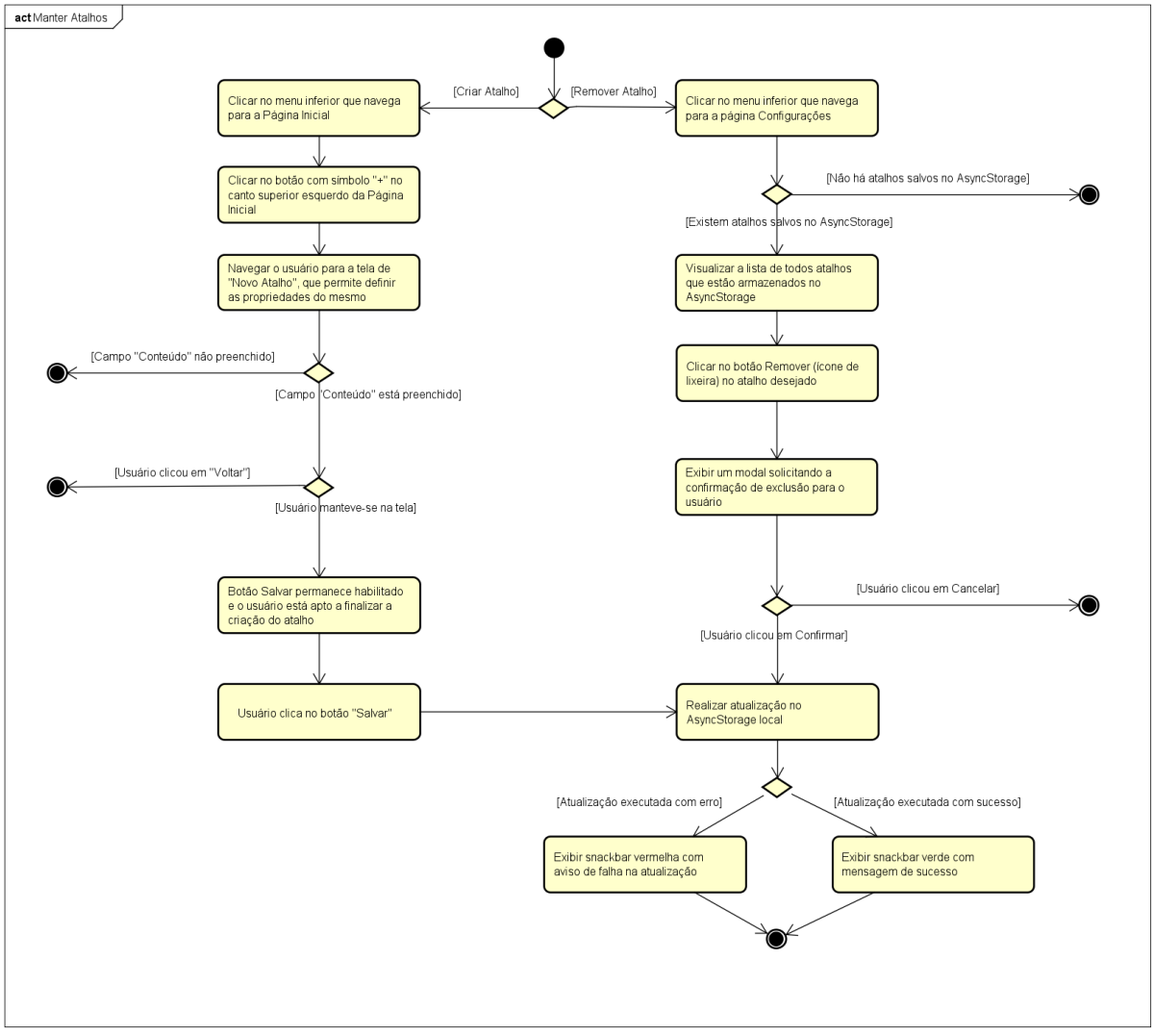
Fonte: Autor.

4.1.3 Diagrama de atividade

Wazlawick (2015) explica que o diagrama de atividade da UML é uma opção extremamente popular para estudar os detalhes de um caso de uso de negócio. Neste formato de notação, podem ser especificadas as diferentes atividades realizadas pelos atores e trabalhadores de negócio na direção da meta geral do caso de uso que está sendo informatizado. Em virtude disso, essa representação vem diretamente de encontro à ideia deste trabalho em utilizar este modelo como complemento aos genéricos casos de uso exibidos anteriormente.

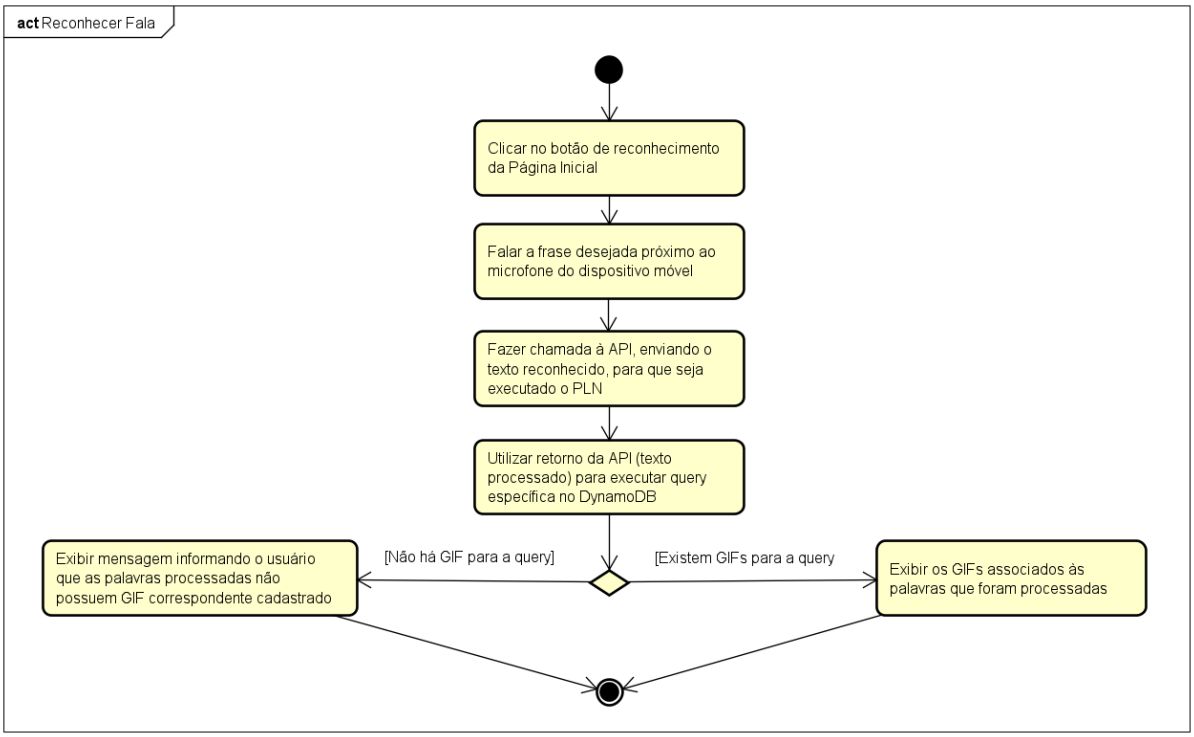
Por mais simples que seja determinada tarefa, o processo de investigar a sequência de passos necessários para executá-la pode reservar surpresas. Em razão disso, os diagramas de atividade enfocam o fluxo de controle entre ações que compõem um processo qualquer (PEREIRA, 2011a). Utilizando-se desta abstração, as Figuras 14 até 18 demonstram a concepção das operações que serão praticadas no aplicativo.

Figura 14 – Diagrama de atividade: Manter atalhos



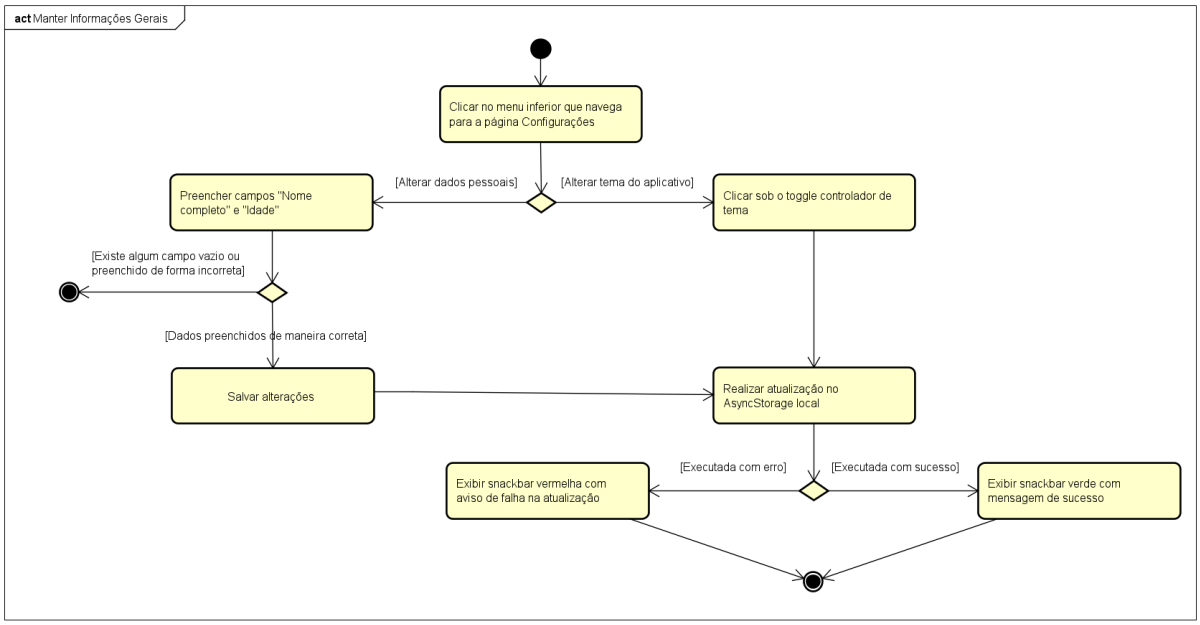
Fonte: Autor.

Figura 15 – Diagrama de atividade: Reconhecer fala



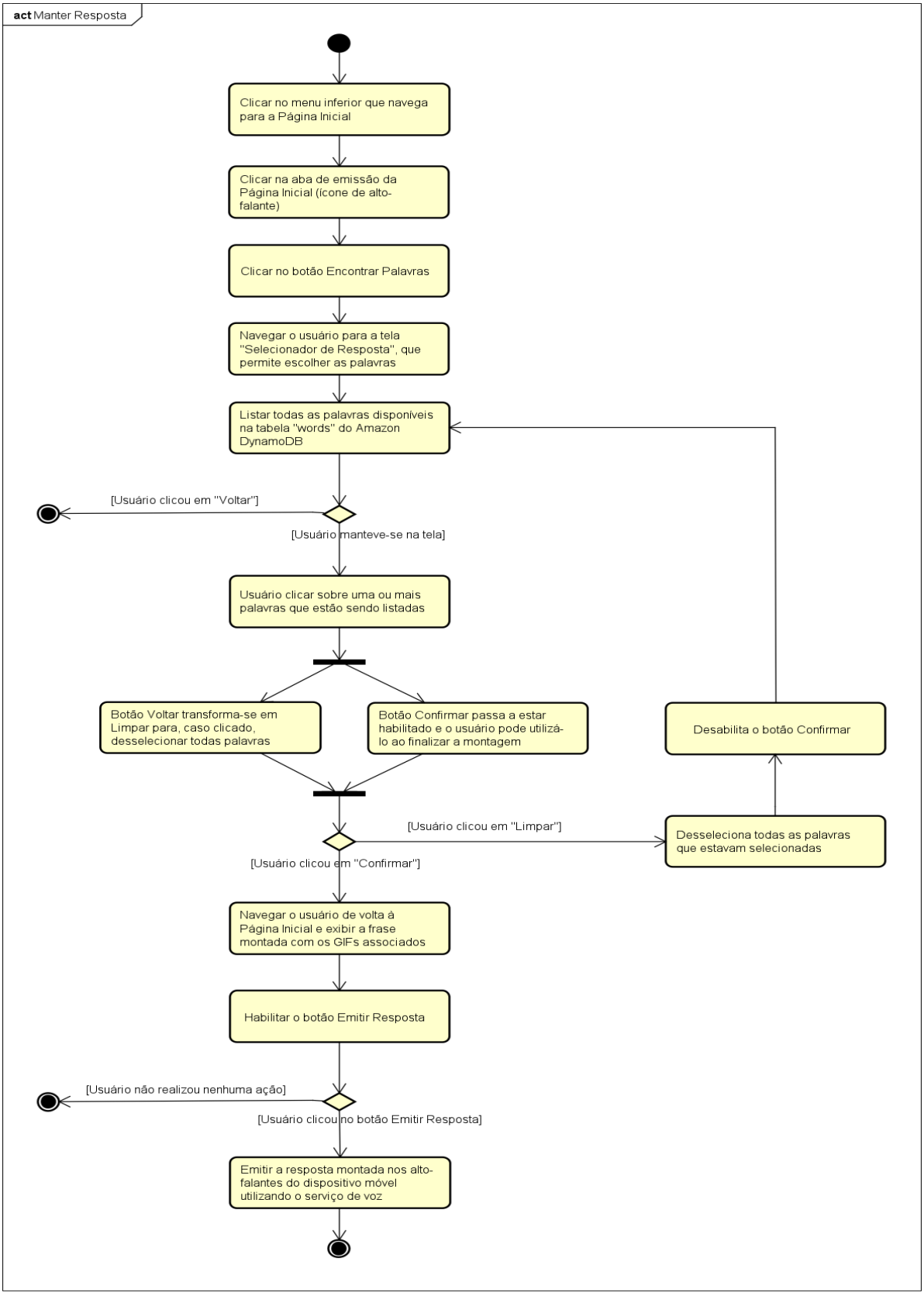
Fonte: Autor.

Figura 16 – Diagrama de atividade: Manter informações gerais



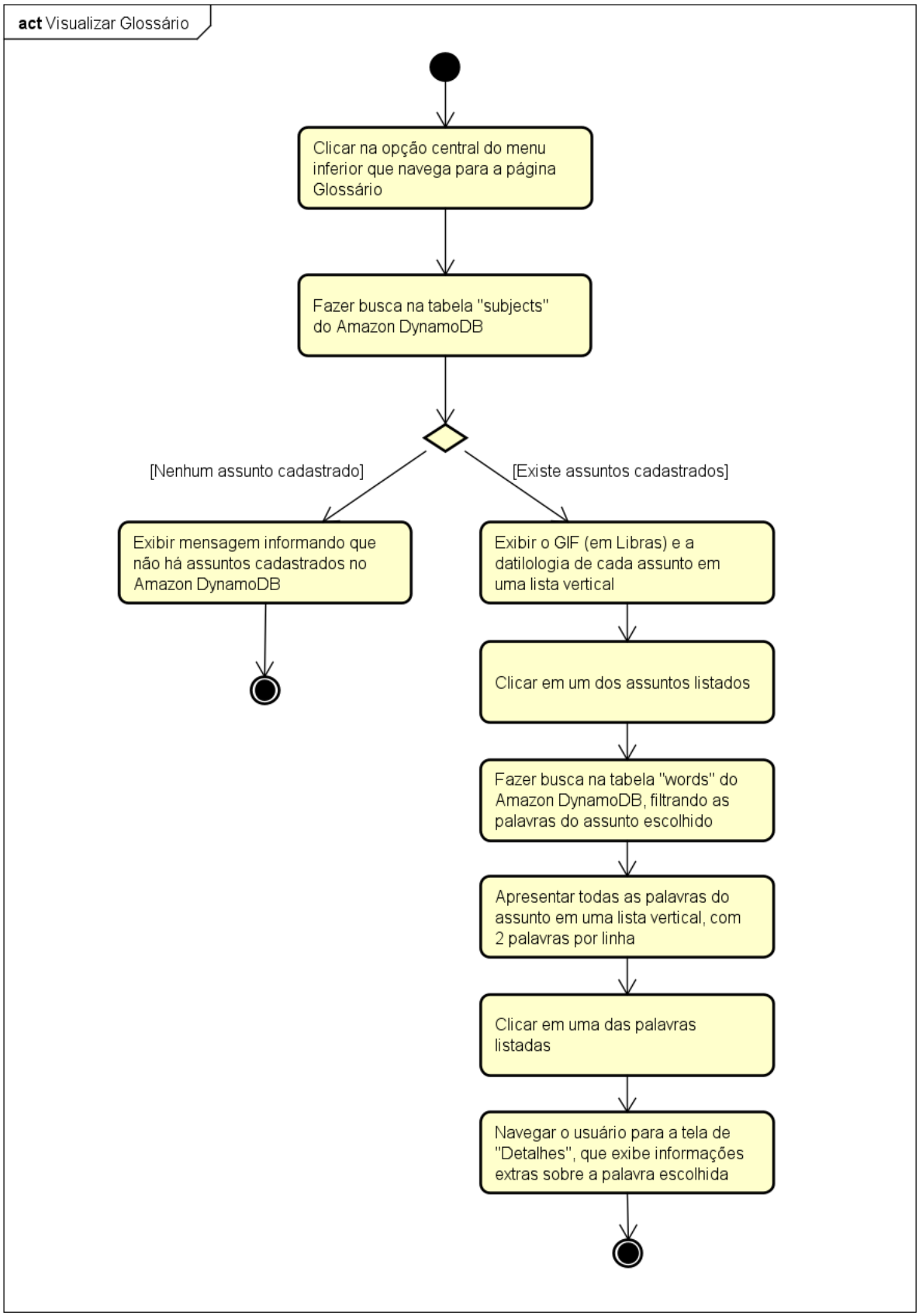
Fonte: Autor.

Figura 17 – Diagrama de atividade: Manter resposta



Fonte: Autor.

Figura 18 – Diagrama de atividade: Visualizar glossário



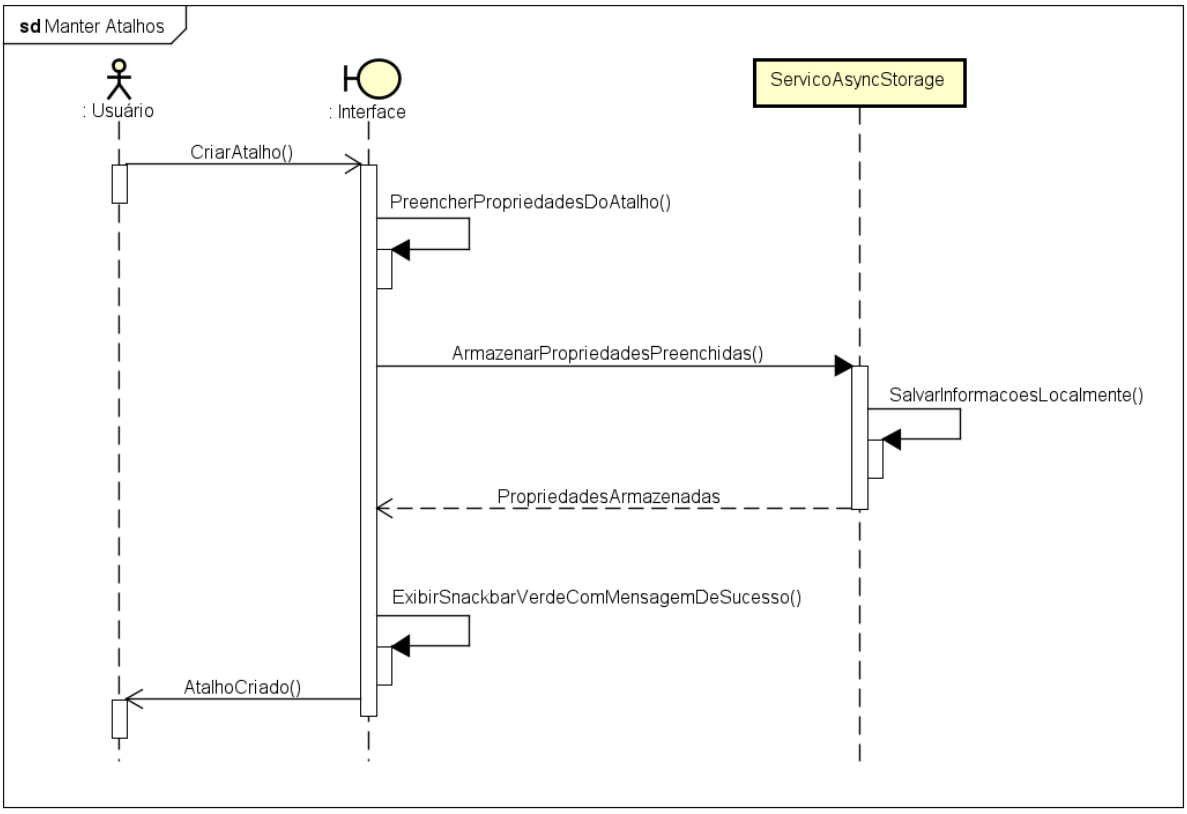
Fonte: Autor.

4.1.4 Diagrama de sequência

O objetivo do diagrama de sequência é apresentar as interações entre objetos na ordem temporal em que elas acontecem. Assim como os outros diagramas da UML, este possui um conjunto de elementos gráficos e baseia-se, dentro do possível, nos casos de uso da aplicação (BEZERRA, 2015). Alinhado a este propósito, Wazlawick (2015) explica que existem duas formas principais de utilizar essa notação. A primeira é construí-la para o fluxo principal de um caso de uso e completá-la com fluxos alternativos; opcionalmente, diagramas distintos poderiam ser construídos para cada fluxo. O autor deixa claro, contudo, que independente da metodologia escolhida, a coisa mais importante é saber qual informação é trocada entre os atores e o sistema.

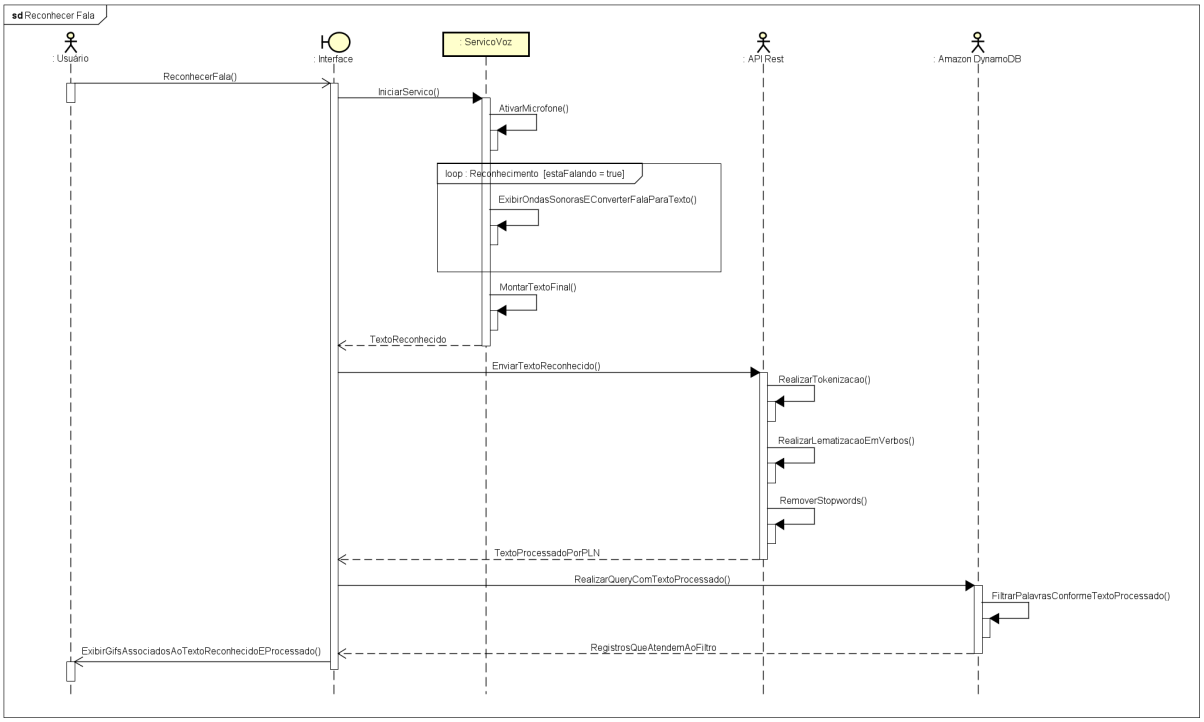
No contexto deste projeto, optou-se pela segunda opção. Em virtude dessa escolha, assim como na Seção 4.1.3, criou-se um diagrama para cada caso de uso mapeado - exibidos nas Figuras 19 até 23.

Figura 19 – Diagrama de sequência: Manter atalhos



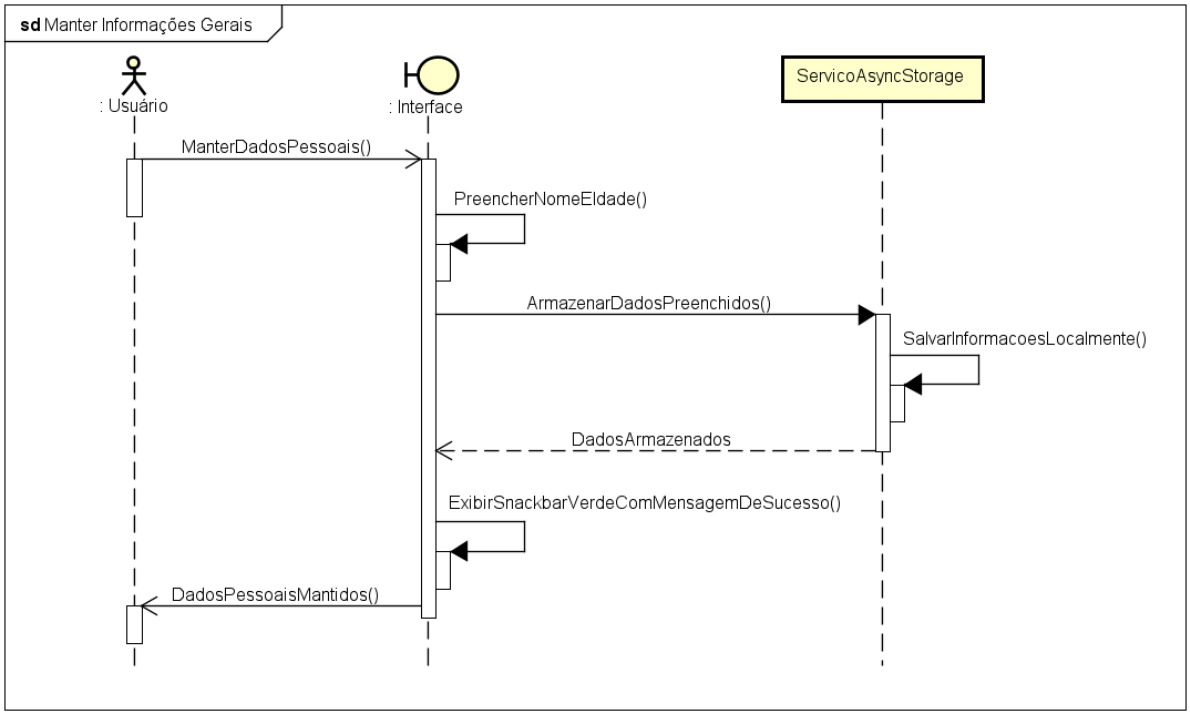
Fonte: Autor.

Figura 20 – Diagrama de sequência: Reconhecer fala



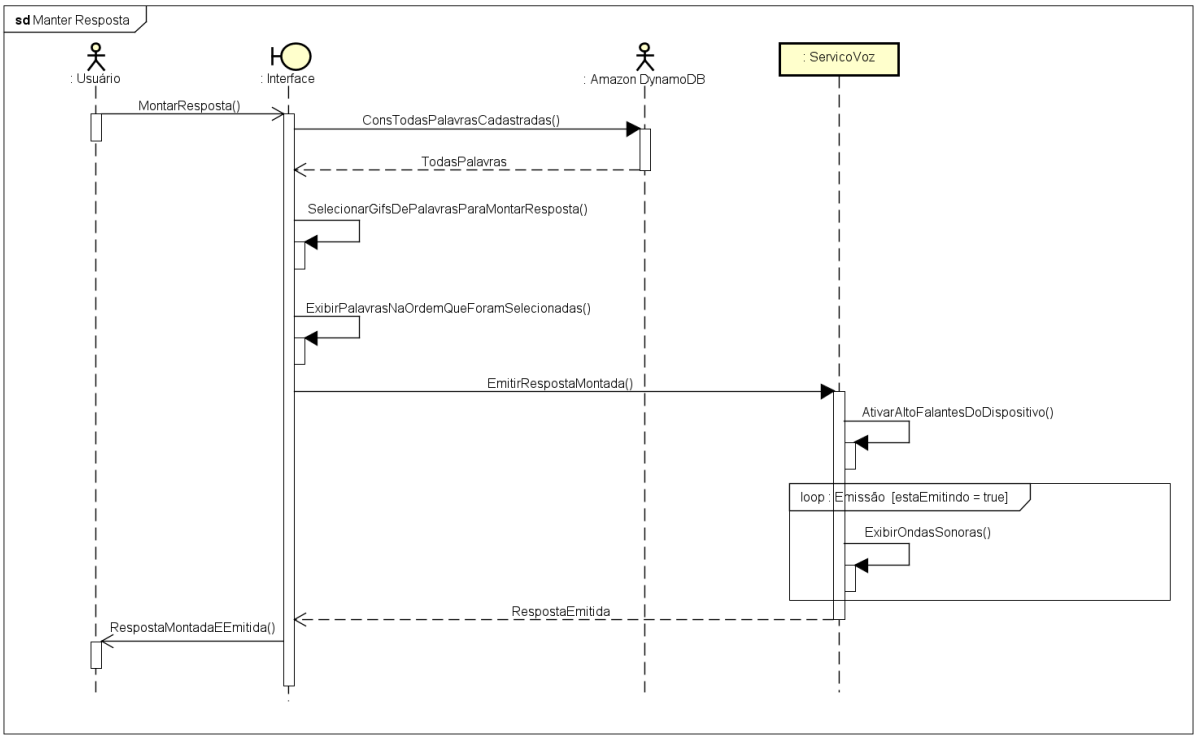
Fonte: Autor.

Figura 21 – Diagrama de sequência: Manter informações gerais



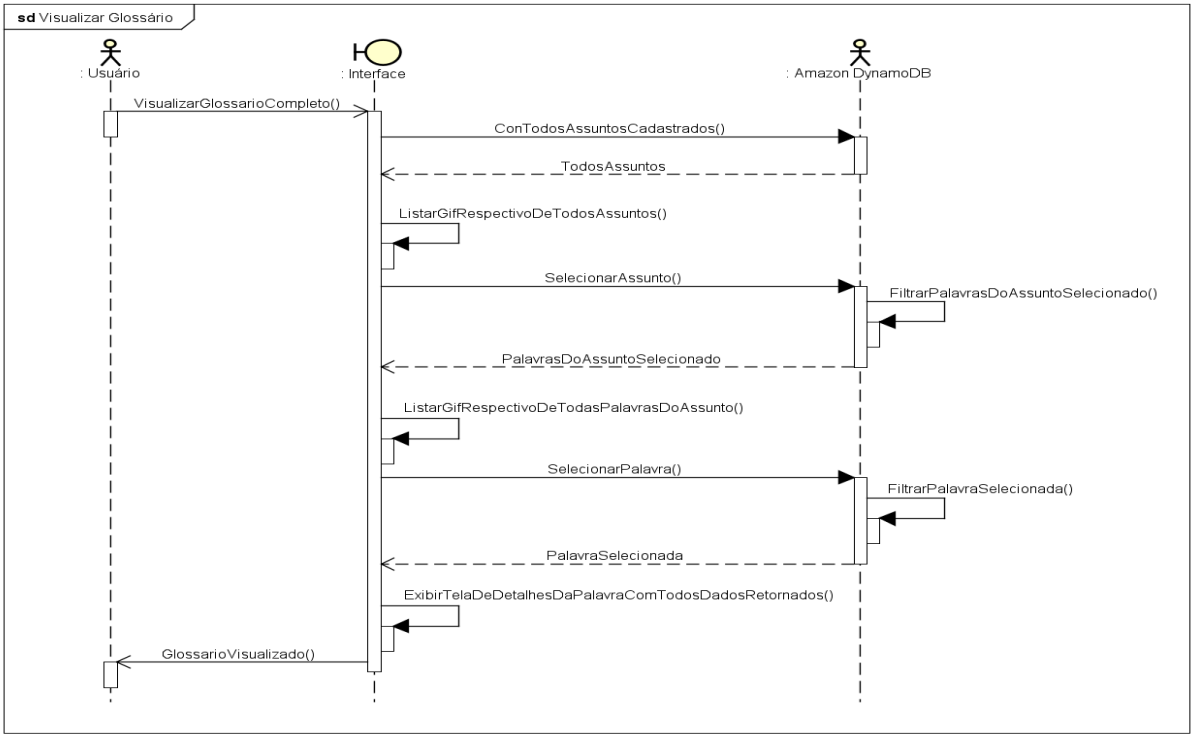
Fonte: Autor.

Figura 22 – Diagrama de sequência: Manter resposta



Fonte: Autor.

Figura 23 – Diagrama de sequência: Visualizar glossário



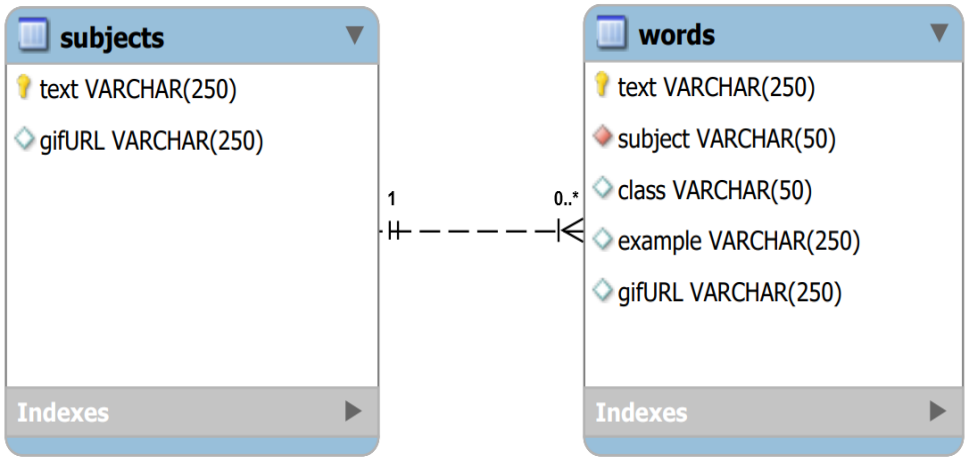
Fonte: Autor.

4.1.5 Modelo ER e Componentização

Em conformidade com a afirmação do início deste Capítulo, esta Seção será utilizada para expor aqueles diagramas que precisaram ser adaptados para esboçar, de fato, a estrutura existente. Retomando a questão do armazenamento de informações mencionada na introdução deste documento, é importante recordar que o presente projeto utilizou um banco de dados NoSQL (não relacional). Dessa forma, acaba sendo difícil demonstrar sua estrutura através de um Modelo ER (Entidade-Relacionamento ou MER), que é constituído por três elementos básicos: entidade, atributos e relacionamento (BARBOZA; FREITAS, 2018).

Entretanto, conforme Machado (2020), a técnica de MER proposta por Peter Chen está definida como uma notação orientada para o desenho de um modelo conceitual, pois permite a descrição desse esquema sem preocupação com problemas de implementação física ou de performance da base de dados. Unindo este foco teórico com a pequena quantidade de informações armazenadas nesta aplicação - a maioria dos dados é tratada em tempo real - decidiu-se fazer os ajustes necessários e representar as tabelas do Amazon DynamoDB em um diagrama que auxilie o leitor a entender sua organização. Essa modificação pode ser visualizada na Figura 24.

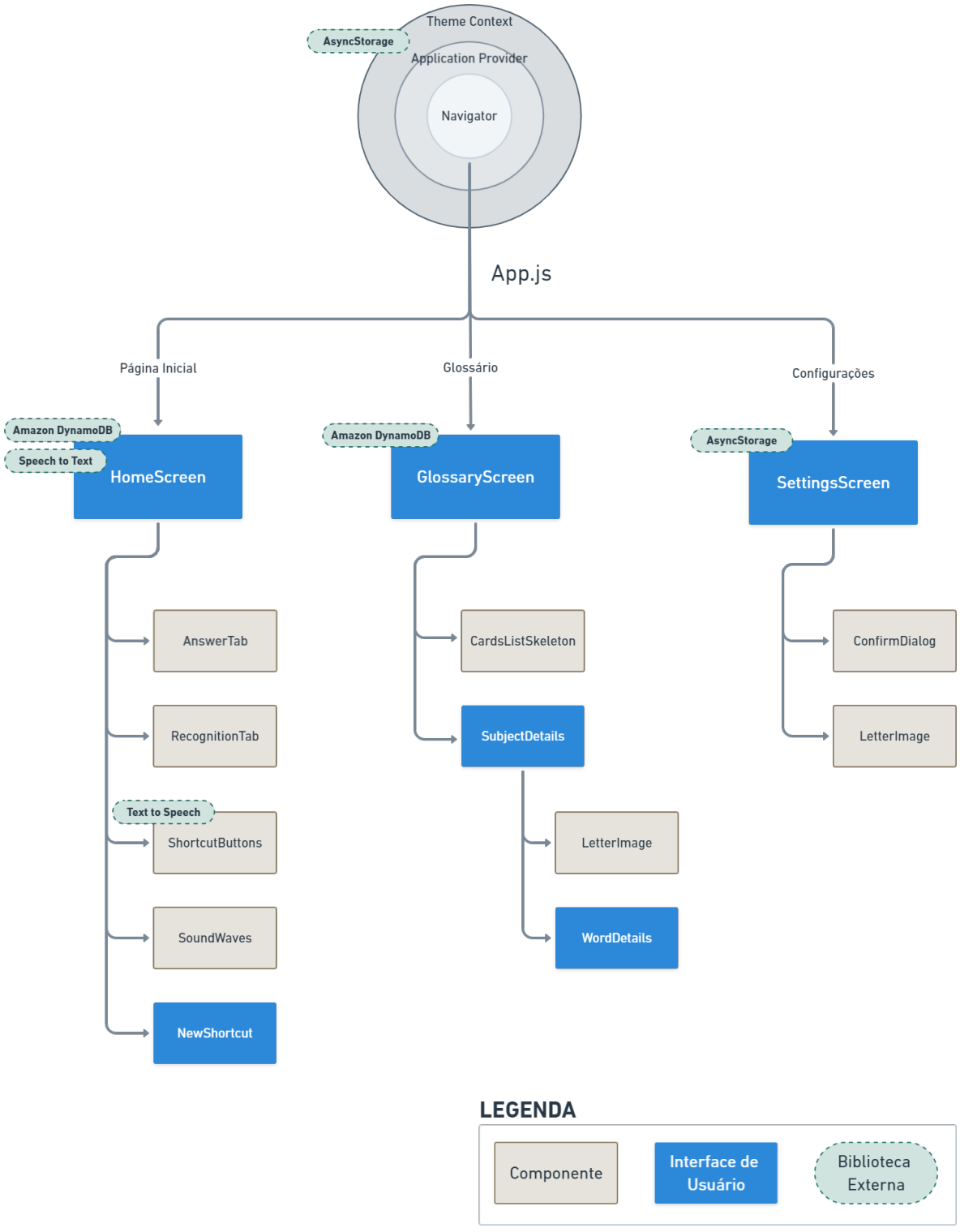
Figura 24 – Modelo ER adaptado: tabelas do Amazon DynamoDB



Fonte: Autor.

Quanto ao diagrama de classes, julgou-se que o mesmo ficaria inconsistente após as adaptações. Isso porque essa representação, que é utilizada na construção do modelo desde o nível de análise até o nível de especificação e é extremamente rica em termos de notação, é focada no paradigma de orientação a objetos (BEZERRA, 2015). O React Native - apesar de possuir essa opção - admite também a utilização exclusiva de componentes que, na prática, são funções, descartando o uso de classes e objetos (estilo adotado neste aplicativo). Sendo assim, considerou-se mais produtivo criar uma representação que expresse como esses componentes/métodos comunicam-se entre si. A Figura 25 aponta essa hierarquia e qual serviço é utilizado em cada uma dessas funções.

Figura 25 – Componentização: estrutura hierárquica das funções do aplicativo



Fonte: Autor.

4.2 Prototipação

Após ter averiguado as informações pertinentes à cada caso de interação com a aplicação e sintetizado todos detalhes nos diagramas construídos, é possível afirmar que obteve-se um bom domínio sobre o que precisa ser desenvolvido. Contudo, tendo em vista que os requisitos concentram-se nas partes conceituais e comportamentais de um sistema, ainda é possível identificar lacunas de definições que, caso não preenchidas, podem gerar retrabalho ou problemas na aplicação móvel. As principais carências que ainda não foram estipuladas remetem essencialmente à parte visual, tais como as cores, posicionamento, ícones e espaçamento dos componentes do aplicativo.

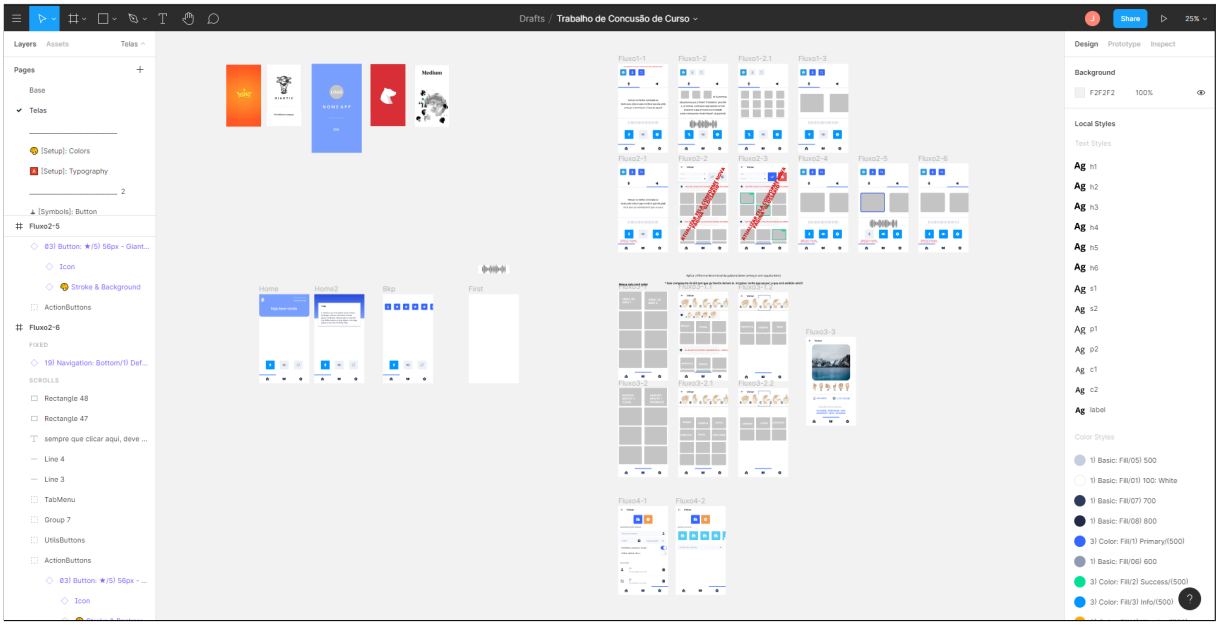
Todos estes pareceres impactam diretamente a forma como o utilizador vai enxergar e interagir com a aplicação móvel. Com a mudança das demandas dos consumidores e o aumento da concorrência, é preciso atentar-se para todos os detalhes que irão melhorar a experiência dos usuários e fazer com que eles consigam acessar todas as informações que precisam e finalizar um processo de forma simples, rápida e prática (SOUZA, 2020). Nos últimos tempos, os debates acerca desses atributos cresceram admiravelmente, chegando a conceber uma área de atuação totalmente focada nestes tópicos - conhecida como interface e experiência de usuário (do inglês *User Interface/User Experience* - UI/UX).

Em virtude dessa atenção especial à interface que seria criada e visando realizar uma migração sutil da teoria para a prática, não seria eficiente iniciar a codificação do aplicativo sem realizar ao menos um esboço de como ficariam as telas. Sendo assim, encontrou-se na prototipação um método atual e bastante útil para criar rascunhos daquilo que, quando finalizado, deveria ser um modelo concreto do *layout* do *app*. Esse processo apresenta como resultado os protótipos, que são a tangibilização de uma ideia, a passagem do abstrato para o físico de forma a representar a realidade - mesmo que simplificada - e propiciar validações (SILVA et al., 2012).

No presente projeto, realizar a prototipagem também foi fundamental para testar as possibilidades que o *Eva Design System* disponibilizava. *Eva* é um *design system* customizável baseado nos princípios de *design atômico* (AKVEO, 2019, tradução do autor). Como não é foco deste trabalho explicar estes conceitos, é importante apenas destacar que a biblioteca UI Kitten - especificada na Seção 2.5.1 - é baseada neste *design system* e, além de vantagens como o padrão de estilo e a reutilização de componentes pré-prontos, disponibiliza os arquivos de prototipação para desenvolvedores, o que acabou facilitando muito esta metodologia.

Dentre todas as ferramentas compatíveis com os recursos disponibilizados pela biblioteca - tais como *Sketch* e *Adobe XD* - optou-se por utilizar o *Figma* para dar suporte à esta etapa do trabalho, devido à sua gratuidade e facilidade de uso. O *Figma* é uma ferramenta completa de prototipagem e *design* baseada na *web* (FIGMA, 2020, tradução do autor). Através desta plataforma, foi possível reunir os componentes e vislumbrar o resultado dessa junção. A Figura 26 apresenta como é a interface do *Figma* e a quantidade total de protótipos produzidos para o **DeafEnd**.

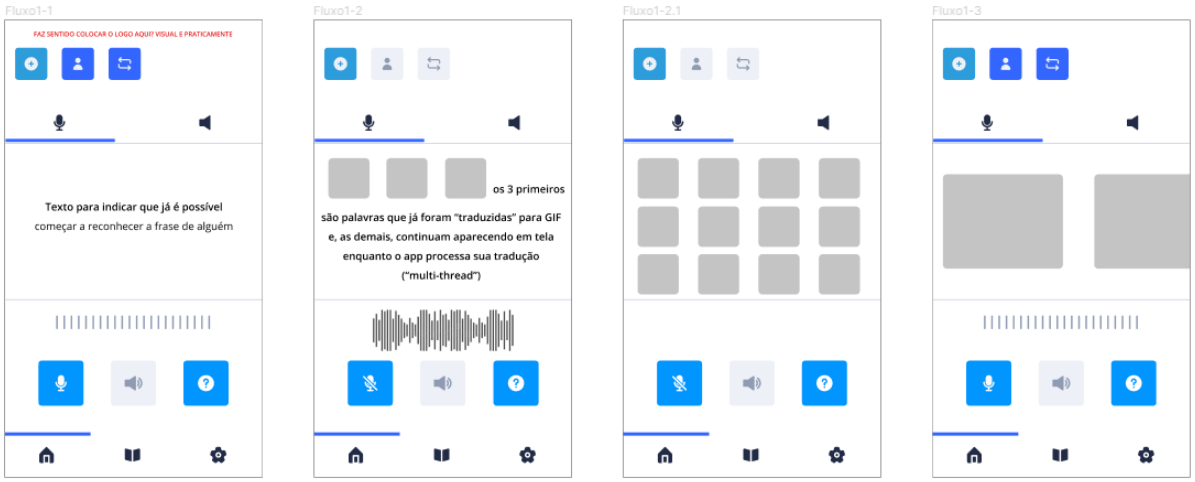
Figura 26 – Prototipação: Visão geral do projeto completo



Fonte: Autor.

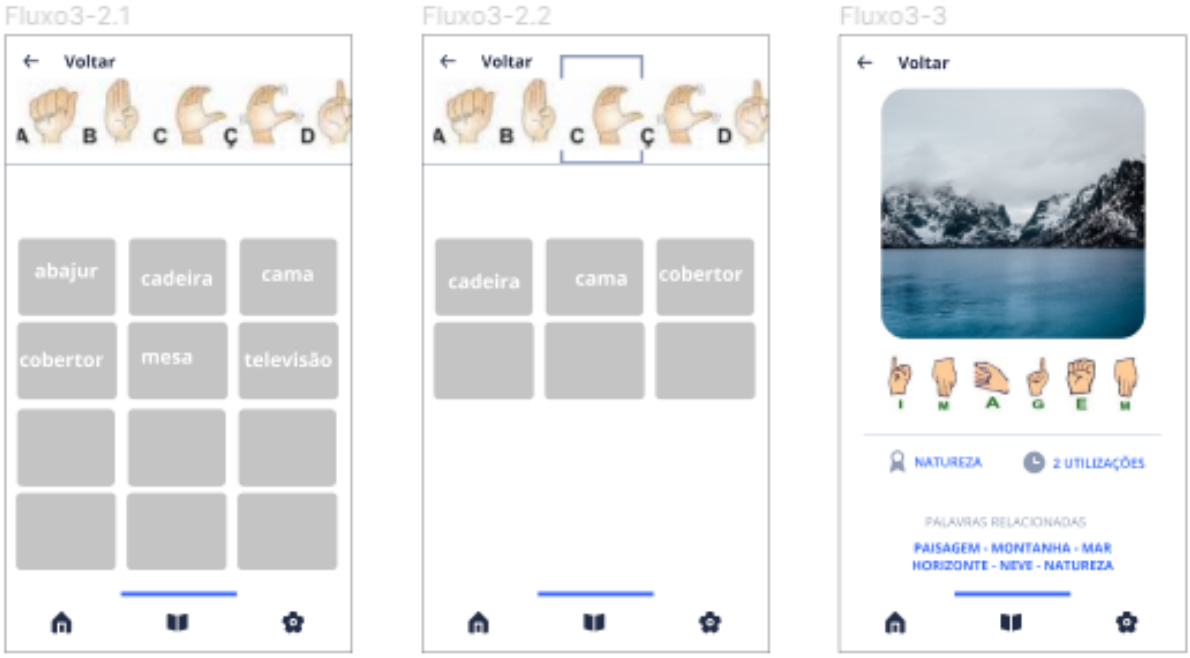
Para evidenciar a importância que este processo teve nas futuras decisões de desenvolvimento, ampliou-se alguns protótipos da Figura 26 para expor, com detalhes, os artefatos que originaram os menus do aplicativo. Na Figura 27, planejou-se o *layout* da Página Inicial, definindo o posicionamento de botões essenciais da aplicação, com funcionalidades como reconhecer a fala e emitir a resposta. Já na Figura 28, é possível reconhecer o surgimento da ideia de filtro com *scroll* horizontal por letras e do catálogo de palavras. E, por fim, a Figura 29 representa o controle dos atalhos customizados do usuário - tanto a listagem, quanto a criação.

Figura 27 – Prototipação: Página inicial



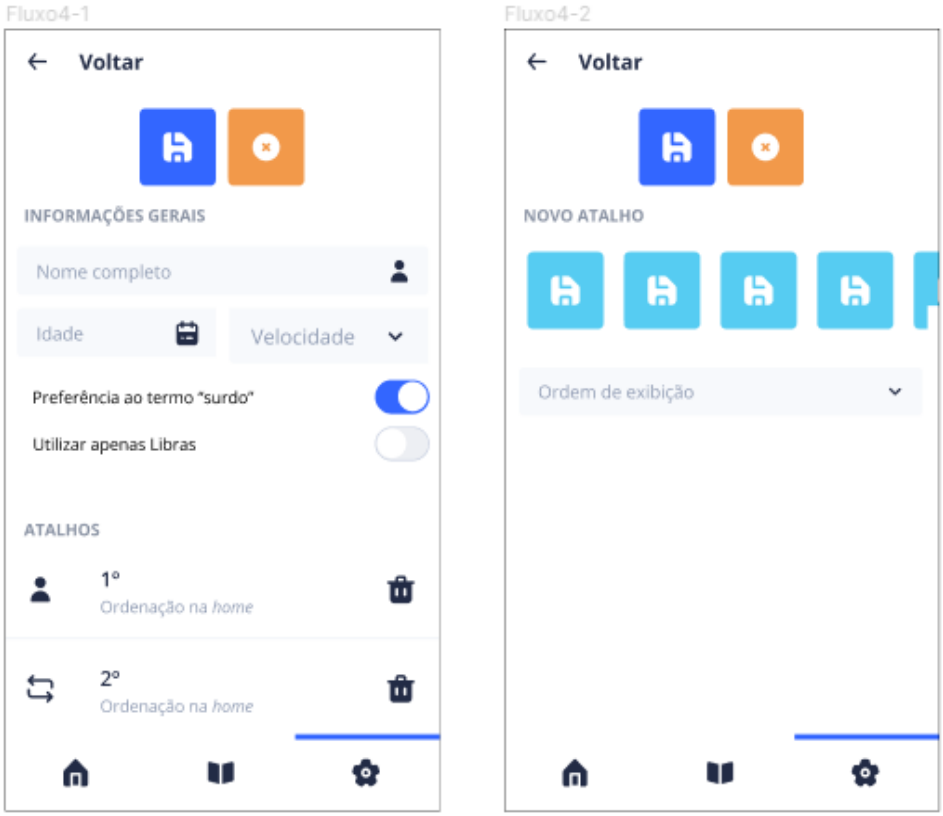
Fonte: Autor.

Figura 28 – Prototipação: Glossário



Fonte: Autor.

Figura 29 – Prototipação: Configurações



Fonte: Autor.

4.3 Implementação do aplicativo

Esta Seção será voltada à parte prática do presente projeto, buscando elucidar as decisões que foram tomadas, os procedimentos realizados e os resultados obtidos. Todo o conteúdo que será exposto neste trecho baseia-se no conhecimento adquirido ao longo do documento, especialmente na Seção 2.5 - que conceitua as tecnologias que serão mencionadas agora. Para facilitar o entendimento do trabalho, dividiu-se os detalhes da implementação em duas frentes principais: a API, responsável por manusear os dados e aplicar, de fato, o processamento de linguagem natural; e o aplicativo, ponto de contato do usuário com a solução completa e responsável por ser o acesso às funcionalidades fornecidas.

Tendo em vista que é necessário uma execução inicial da API para que as informações estejam prontas para serem exibidas na aplicação móvel, a descrição iniciará por ela. Vale ressaltar que os comandos demonstrados foram executados em um sistema operacional Linux (Ubuntu) e que, apesar disso implicar pouca diferença nas ferramentas utilizadas caso os escolhidos fossem o Windows ou o macOS, ainda assim podem haver algumas particularidades.

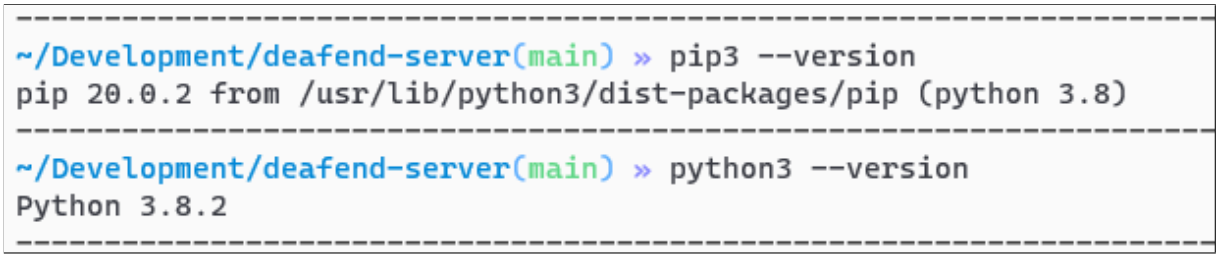
4.3.1 API

Para criar um novo projeto com o FastAPI, é bastante simples. Porém, antes de executar seus comandos específicos, é necessário instalar um gerenciador de pacotes para o Python e garantir que as versões são suportadas. Esse processo pode ser feitos através da execução dos comandos abaixo no *Terminal* (o sinal $ indica o início de um novo comando e não precisa ser digitado):

```
$ sudo apt install python3-venv python3-pip
$ pip3 --version
$ python3 --version
```

Espera-se que o resultado dos dois comandos finais exiba a versão do Python e do PIP que acabaram de ser instaladas, conforme mostra a Figura 30.

Figura 30 – API: Verificação de versão



Fonte: Autor.

A próxima etapa é escolher um local para salvar o projeto - recomenda-se algo como a pasta `Documentos`, dentro do diretório do usuário - e criar a estrutura mais simples possível para

receber uma requisição HTTP (o exemplo disposto abaixo consta na própria documentação). Sendo assim, basta criar um arquivo chamado `main.py` e colocar o seguinte conteúdo:

```
from fastapi import FastAPI
app = FastAPI()
@app.get("/")
def exemplo():
    return {"Projeto": "DeafEnd"}
```

Nesse momento, já é possível instalar as dependências do FastAPI, criar o projeto e disponibilizar o servidor da aplicação que irá retornar o JSON (JavaScript Object Notation) definido no código anterior. Caso os comandos listados abaixo sejam reproduzidos com êxito, as mensagens do *Terminal* devem ser semelhantes às demonstradas na Figura 31 - não serão iguais pois, no exemplo, as bibliotecas já haviam sido instaladas. Além disso, deveria ser possível acessar o endereço http://localhost:8000/ e visualizar o conteúdo da página exibindo `{"Projeto": "DeafEnd"}`.

```
# Dentro do diretório definido:
$ python3 -m venv .env
$ source .env/bin/activate
$ pip3 install -U fastapi
$ pip3 install -U uvicorn
$ uvicorn main:app --reload # 'main' indica o nome do arquivo Python
```

Figura 31 – API: Instalação de dependências e *start* do servidor

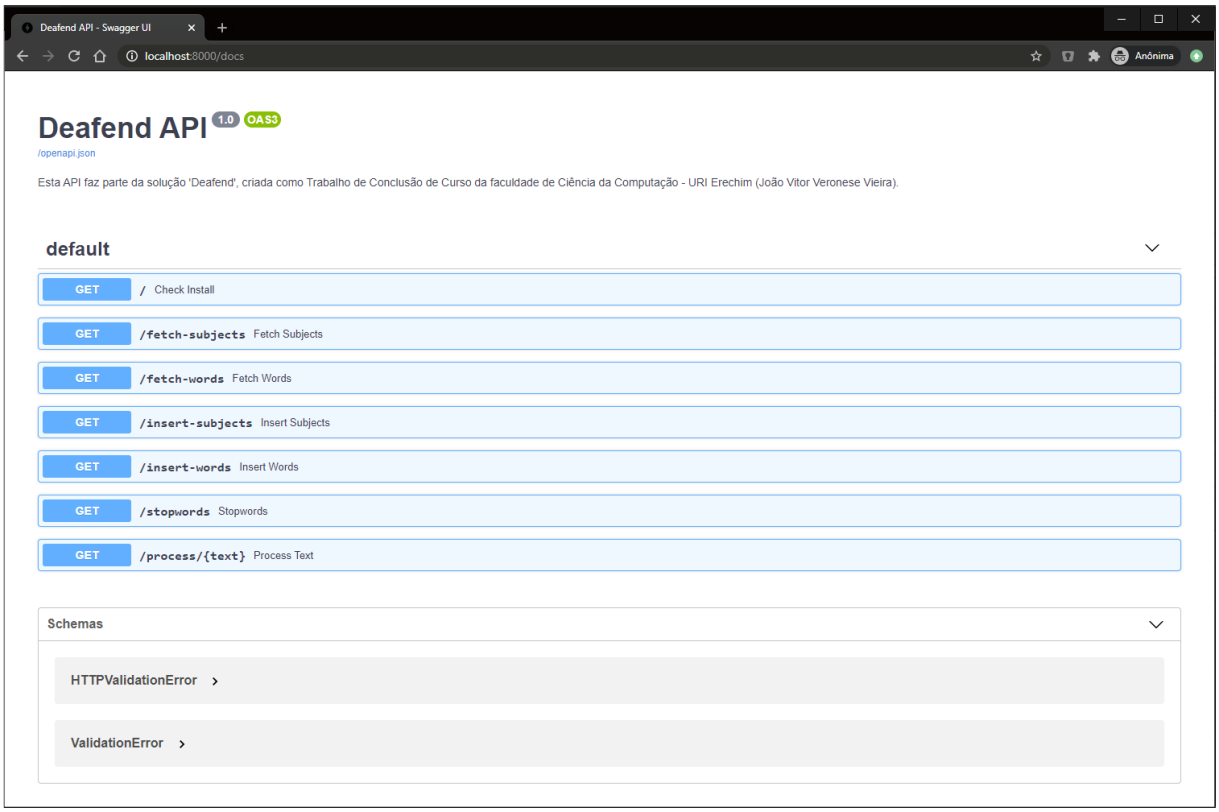
```
~/Development/deafend-server(main) » source .env/bin/activate
(.env) ~/Development/deafend-server(main) » pip3 install -U fastapi
Requirement already up-to-date: fastapi in ./.env/lib/python3.8/site-packages (0.61.2)
Requirement already satisfied, skipping upgrade: pydantic<2.0.0,>=1.0.0 in ./.env/lib/python3.8/site-packages (from fastapi) (1.6.1)
Requirement already satisfied, skipping upgrade: starlette==0.13.6 in ./.env/lib/python3.8/site-packages (from fastapi) (0.13.6)
(.env) ~/Development/deafend-server(main) » pip3 install -U uvicorn
Requirement already up-to-date: uvicorn in ./.env/lib/python3.8/site-packages (0.12.2)
Requirement already satisfied, skipping upgrade: click==7.* in ./.env/lib/python3.8/site-packages (from uvicorn) (7.1.2)
Requirement already satisfied, skipping upgrade: h11>=0.8 in ./.env/lib/python3.8/site-packages (from uvicorn) (0.10.0)
(.env) ~/Development/deafend-server(main) » uvicorn main:app --reload
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [930] using statreload
INFO:     Started server process [932]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Fonte: Autor.

Após estas etapas, a API está criada e pronta para que novos métodos sejam incrementados. Contudo, uma das funcionalidades mais interessantes do FastAPI - além da simplicidade demonstrada - é a documentações interativas que ele possui nativamente. Para visualizá-las, basta acessar o endereço da API que, por padrão é http://localhost:8000/, e adicionar os sufixos

`/docs` ou `/redoc`. As duas opções são muito interessantes, mas a primeira possibilita a execução dos métodos disponibilizados e, por isso, será utilizada para demonstrar as requisições. Na Figura 32, estão listados todos os métodos criados no presente projeto.

Figura 32 – API: Documentação interativa dos métodos disponíveis



Fonte: Autor.

Cada um dos 7 métodos apresentados atua em uma parcela específica da solução e tem papel fundamental no resultado final. O método `Check Install` (`/`) serve exatamente para o mesmo propósito que o exemplo apresentado no início desta Subseção: checar o correto funcionamento do projeto. Este tipo de prática é bastante comum no momento de disponibilizar uma API em um servidor *cloud*, por exemplo, pois acaba sendo uma forma rápida de checar que as respostas estão sendo retornadas. Como este é o único método genérico do programa, os demais serão explanados, em detalhes, em dois momentos distintos: a pré-carga e o processamento dos dados.

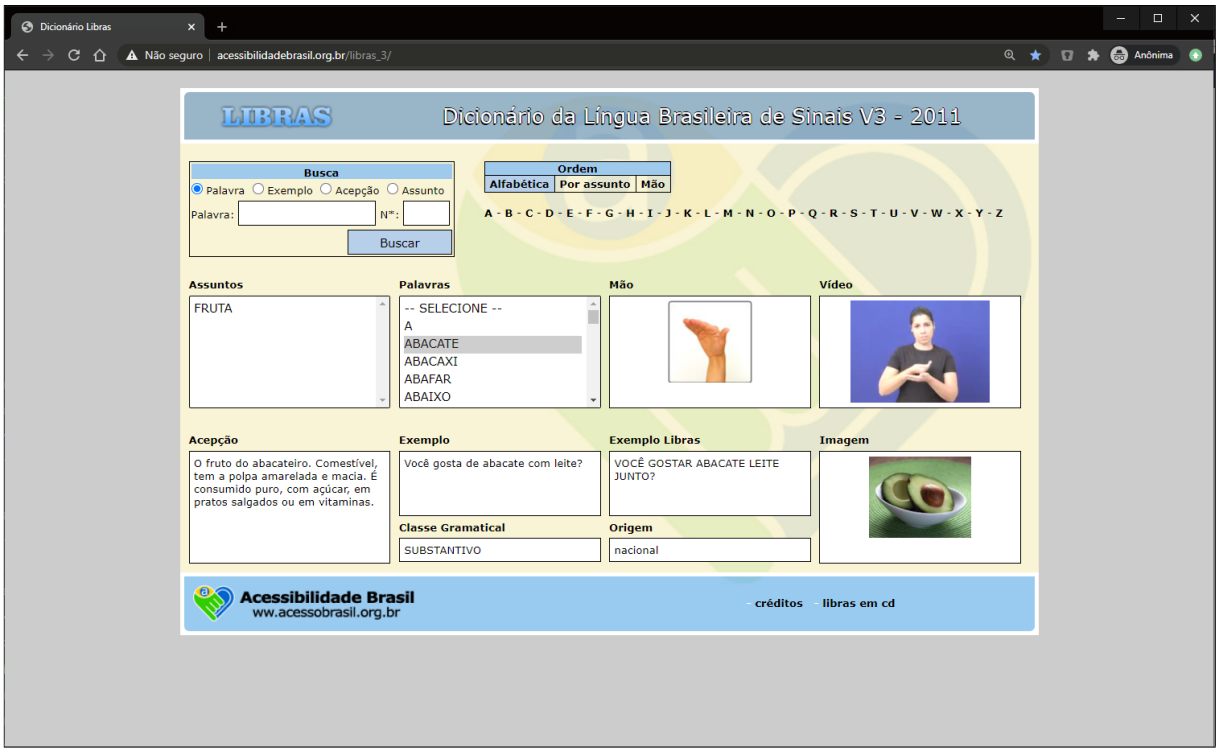
4.3.1.1 Pré-carga de dados

A etapa de pré-carga é, de fato, o período em que o conteúdo utilizado nesta aplicação móvel é preparado. É justamente nesta fase que há a inserção das informações no Amazon DynamoDB e no Amazon S3. Recordando o modelo ER adaptado exposto na Seção 4.1.5, as agremiações tratadas são assuntos (`subjects`) e palavras (`words`), sendo que um assunto pode

contar zero ou muitas palavras. Esse detalhe é crucial, pois indica que a hierarquia de elementos inicia pelos `subjects`. Logo, essa é a primeira informação que será manuseada.

E, durante este momento de implementação da API, o principal objeto de estudo são os GIFs dos sinais da Libras. Entretanto, foi bastante difícil encontrar um local que disponibilizasse esses recursos, em formato GIF, e de forma aberta. Este foi, inclusive, o motivo deste tratamento existir pois, sem ele, não haveria material para apresentar a tradução dos textos que seriam reconhecidos pelo aplicativo. Em virtude disso, antes de entrar nos detalhes da implementação, é necessário destacar a fonte de dados utilizada. Todos os sinais usados no aplicativo provêm do Dicionário da Língua Brasileira de Sinais V3 (http://www.acessibilidadebrasil.org.br/libras_3/), que tem sua interface exibida na Figura 33.

Figura 33 – Dicionário de Libras: Fonte dos vídeos utilizados



Fonte: Lira e Souza (2011)

Encontrar esta base de dados - financiado pelo Ministério da Ciência, Tecnologia e Inovação (LIRA; SOUZA, 2011) - foi fundamental para o êxito deste projeto. Como é possível ver na Figura 33 exibida anteriormente, o *site* contém muitas informações sobre cada um dos sinais catalogados na plataforma (assunto, forma da mão, acepção, exemplo de frases). Além disso, todos os dados mostrados em tela são obtidos através de chamadas REST, o que facilita uma possível integração com esse sistema. Durante o período de construção deste trabalho, tentou-se diversas vezes obter uma permissão formal para utilizar estas informações pois, mesmo que este *site* seja público e na época de seu lançamento houvesse distribuição deste material em CD, seria importante possuir o consentimento dos autores.

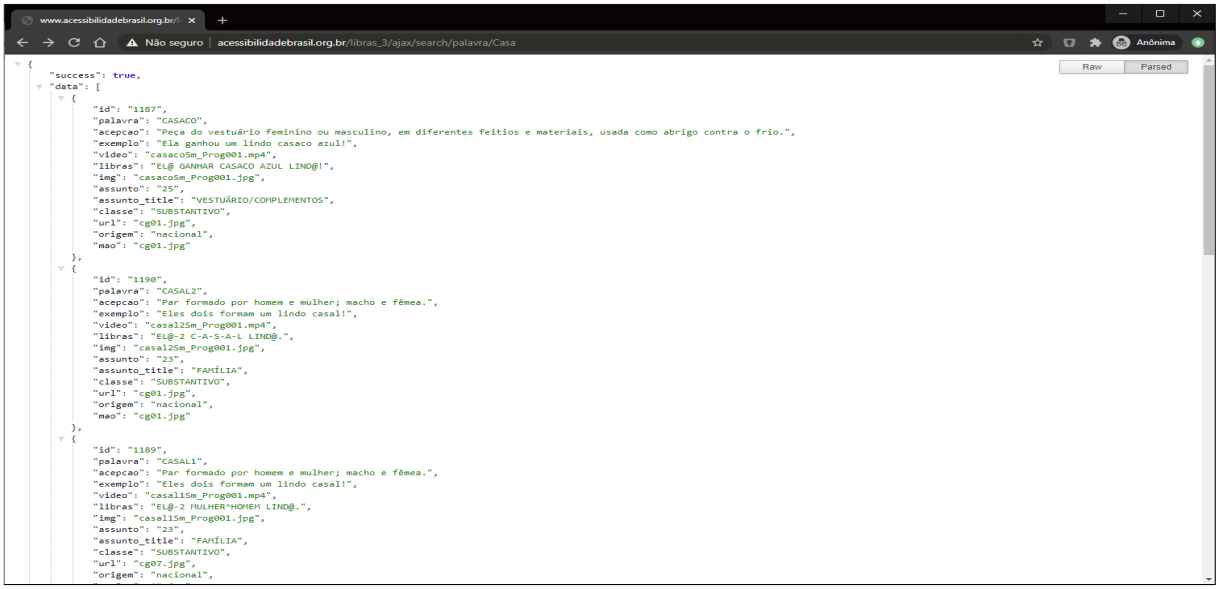
Infelizmente, mesmo após o contato - relatado no Apêndice B - com três setores diferentes do INES (Instituto Nacional de Educação de Surdos), o pedido não recebeu nenhuma resposta até o dia 06/11/2020. Sendo assim, é fundamental deixar claro que estas informações foram utilizadas apenas para fins de pesquisa e validação de hipótese e se, em algum momento, esta concordância explícita for obtida, este documento será imediatamente incrementado com tal parecer. No que tange à parte técnica, havia outro entrave para completar esta integração: os sinais apresentados no *site* são vídeos MP4. Fica claro, portanto, a necessidade de conversão desse conteúdo considerando o objetivo de utilizar GIFs no aplicativo e esta é, rigorosamente, a principal responsabilidade dos métodos `Fetch Subjects` (`/fetch-subjects`) e `Fetch Words` (`/fetch-words`).

Abordando primeiramente a rota de obtenção de assuntos, seu encargo inicial é garantir que a estrutura de diretórios projetada para a execução do *script* está correta, já que ela é a primeira a ser chamada. Na sequência, ocorre a busca dos registros guiada por um escopo controlado (manual). Esta limitação foi utilizada pois, para essa prova de conceito, não era necessário importar todos os sinais agrupados nos 21 assuntos que o dicionário oferece. Sendo assim, escolheu-se apenas 6 considerados suficientes para montar frases comuns do cotidiano com base nas palavras mais usadas da língua portuguesa (NEVES, 2019), que totalizam um conjunto de 741 palavras convertidas, conforme detalhado no bloco de código subsequente.

```
[
  {"id": "3", "nome": "Animal"},       // 151 palavras
  {"id": "22", "nome": "Casa"},        // 107 palavras
  {"id": "23", "nome": "Família"},     // 46 palavras
  {"id": "5", "nome": "Fruta"},        // 37 palavras
  {"id": "1", "nome": "Outro"},        // 318 palavras
  {"id": "30", "nome": "Sentimento"}, // 82 palavras
]
```

Depois da definição dos assuntos que serão manipulados, um arquivo `_subjects.json` com a estrutura disposta anteriormente é criado, objetivando diminuir a quantia de chamadas HTTP que serão realizadas ao longo desta pré-carga. Após, é possível iniciar os *requests* para o *endpoint* do dicionário. Esse endereço - e todos os demais que serão apresentados nesta Seção - possuem o mesmo domínio (http://www.acessibilidadebrasil.org.br/libras_3/ajax/), que será denominado `URL_BASE`. Dessa forma, a primeira requisição HTTP GET realizada utiliza a URL <URL_BASE>/search/palavra/<assunto>. Caso a propriedade `palavra`, contida no JSON de resposta exibido na Figura 34, seja idêntica ao parâmetro `<assunto>` que foi buscado na URL, é possível garantir que aquele sinal representa o assunto atual. Com isso, é possível distinguir o atributo `video` deste mesmo objeto e utilizar este valor para fazer o *download* da mídia por meio do método `urlretrieve` da biblioteca Python `urllib.request`.

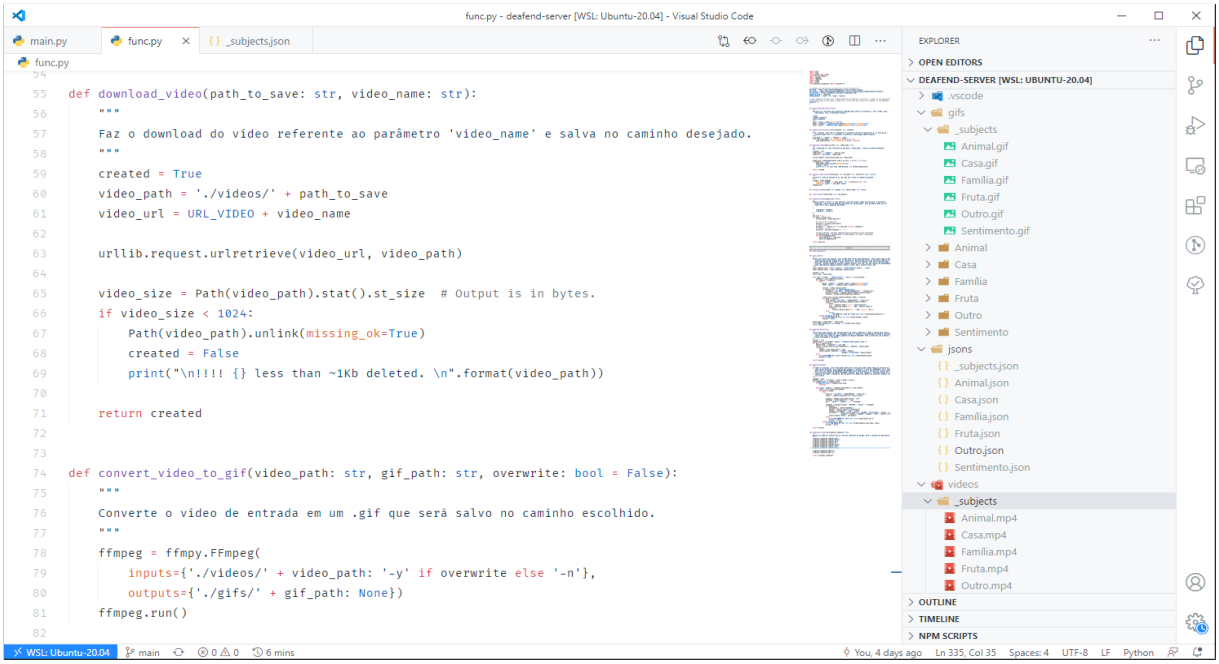
Figura 34 – Dicionário de Libras: Estrutura JSON de resposta para assunto



Fonte: Autor.

Todos os vídeos obtidos por este *script* serão depositados no diretório `videos` para que, no passo seguinte, possam ser convertidos para GIF através do método `FFmpeg` da biblioteca Python `ffmpy`. A Figura 35 apresenta, em seu lado esquerdo, os dois métodos que, respectivamente, salva o vídeo no diretório escolhido e converte o MP4 em GIF. No lado direito da mesma figura, é possível visualizar como fica a hierarquia de arquivos e diretórios quando a execução dos métodos é finalizada.

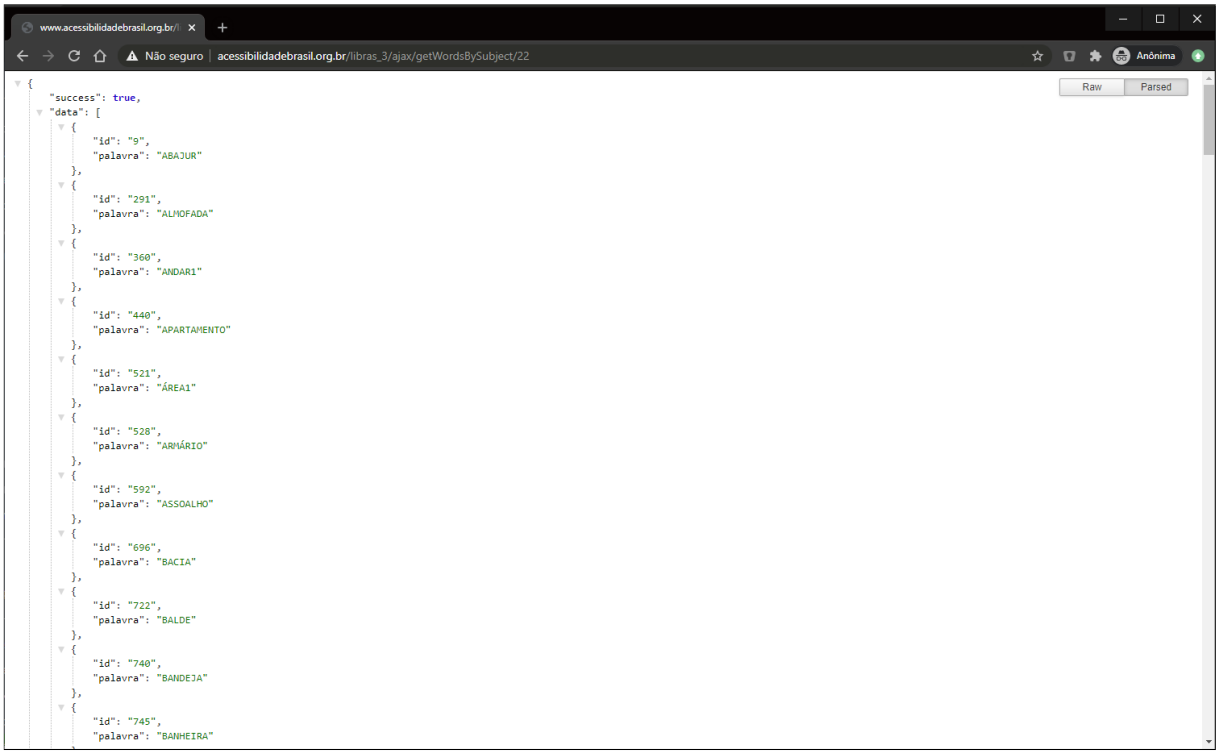
Figura 35 – API: Funções de *download* e conversão e lista de diretórios



Fonte: Autor.

Para a obtenção das palavras, o fluxo de trabalho é bastante similar. Um dos pontos divergentes diz respeito ao escopo de execução que, diferentemente da tratativa manual feita para os assuntos, agora é dinâmico e engloba todos os registros salvos como conteúdo do arquivo `_subjects.json`. O outro é o *endpoint* utilizado que, neste caso, captura as palavras pelo seu assunto (<URL_BASE>/getWordsBySubject/<id_assunto>). Com a resposta obtida - exibida na Figura 36 - também é criado um arquivo JSON com o nome do assunto (ex: `Casa.json`) e, individualmente, é realizado o *download* do vídeo com o sinal da palavra e sua conversão, de MP4 para GIF.

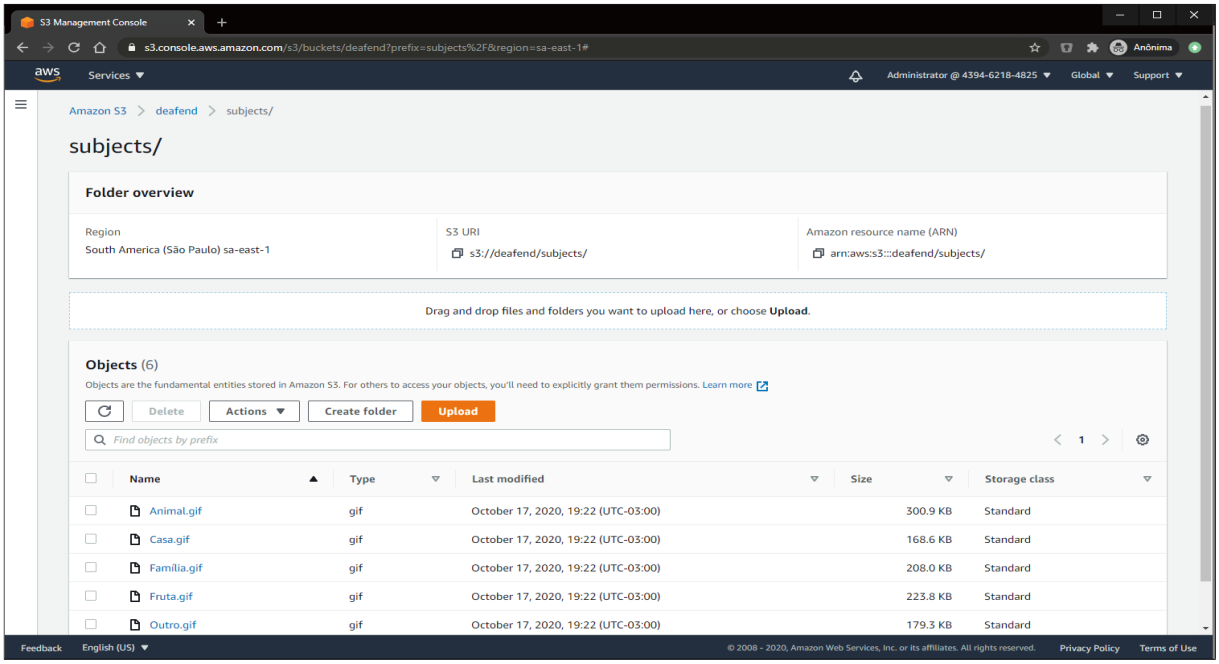
Figura 36 – Dicionário de Libras: Estrutura JSON de resposta para palavras por assunto



Fonte: Autor.

Neste momento, todos os sinais que serão disponibilizados no aplicativo - tanto para os assuntos, quanto para as palavras - foram obtidos e estão guardados, localmente, na estrutura de pastas esperada. Portanto, as próximas chamadas à API, dos métodos `Insert Subjects` (`/insert-subjects`) e `Insert Words` (`/insert-words`), tratam essencialmente da *inserção* desse conteúdo na plataforma AWS. Iniciando pela rota de criação dos assuntos, o *script* itera sobre cada arquivo GIF contido dentro do diretório `/gifs/_subjects/`. Imediatamente, o método que realiza o *upload* desses arquivos no Amazon S3 é invocado, passando como parâmetro o *bucket* `deafend`. A Figura 37 exibe a listagem de GIFs inseridos no diretório `subjects`, dentro do *bucket* `deafend` do Amazon S3.

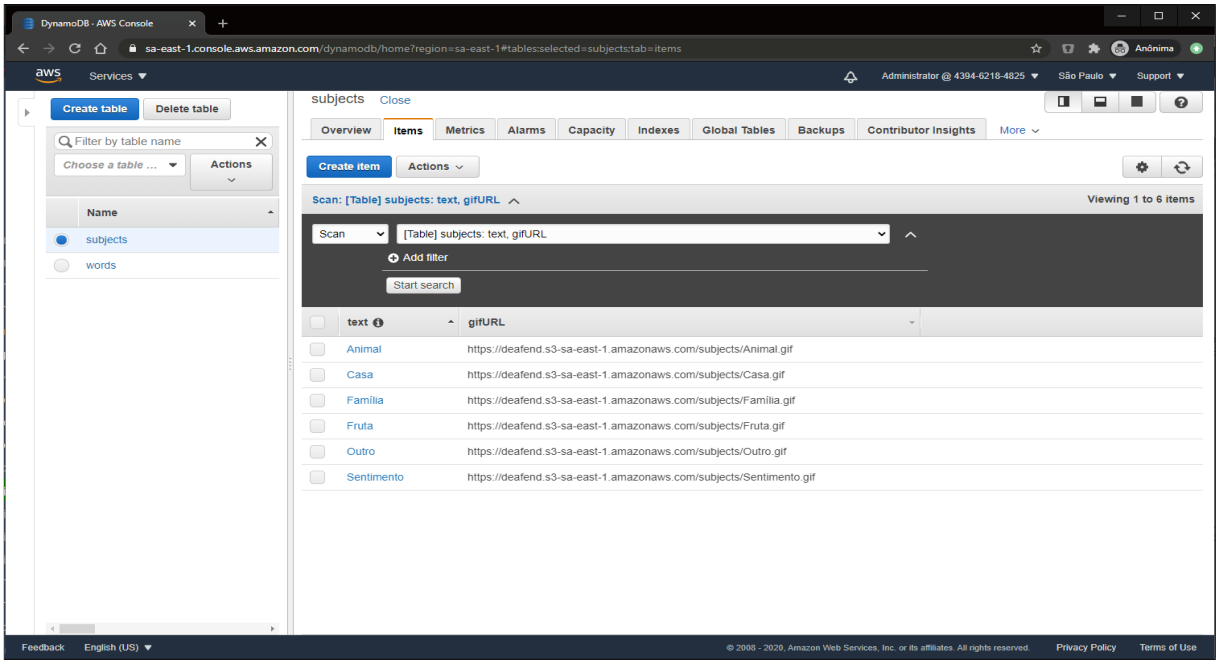
Figura 37 – API: Diretório *subjects* no Amazon S3



Fonte: Autor.

Caso o *upload* - e a definição de permissão de leitura como pública - tenha ocorrido com sucesso, um novo registro (respectivo ao objeto criado) é inserida no Amazon DynamoDB, na tabela `subjects`. Essa tabela será útil para agrupar os sinais disponíveis no *app* e facilitar a procura do usuário. A Figura 38 apresenta todos os assuntos já cadastrados na ferramenta.

Figura 38 – API: Tabela *subjects* no Amazon DynamoDB



Fonte: Autor.

Novamente, o fluxo de trabalho para a inserção de palavras é muito semelhante ao realizado para os assuntos. A única diferença relevante entre os dois processos é que, para as palavras, o endereço <URL_BASE>/getWordById/<id_palavra> recebe uma chamada HTTP GET extra. O retorno dessa requisição permite que os campos `class` e `example` também sejam populados na tabela `words` do Amazon DynamoDB, uma vez que essas informações não estão armazenadas localmente. No trecho de código seguinte, é possível visualizar a estrutura JSON que cria um novo registro na tabela `subjects` (à esquerda) e na tabela `words` (à direita).

```
{
  "text": <assunto>,
  "gifURL": <url_pública_do_gif>,
}
```

```
{
  "text": <palavra>,
  "gifURL": <url_pública_do_gif>,
  "subject": <assunto_da_palavra>,
  "example": <frase_em_português>,
  "class": <classe_gramatical>,
}
```

Isto posto, toda a pré-carga de dados pode ser considerada finalizada e o aplicativo está apto a exibir o conteúdo disponível. É importante destacar, porém, que este processamento não necessariamente precisaria ser executado através de métodos da API. Contudo, considerou-se essa abordagem pertinente, uma vez que é possível controlar cada etapa de execução deste processamento através de chamadas HTTP feitas pela documentação interativa da própria ferramenta, que acaba diminuindo a necessidade de intervenção manual no procedimento e, se necessário, permite a inclusão de parâmetros condicionais para a expansão das opções.

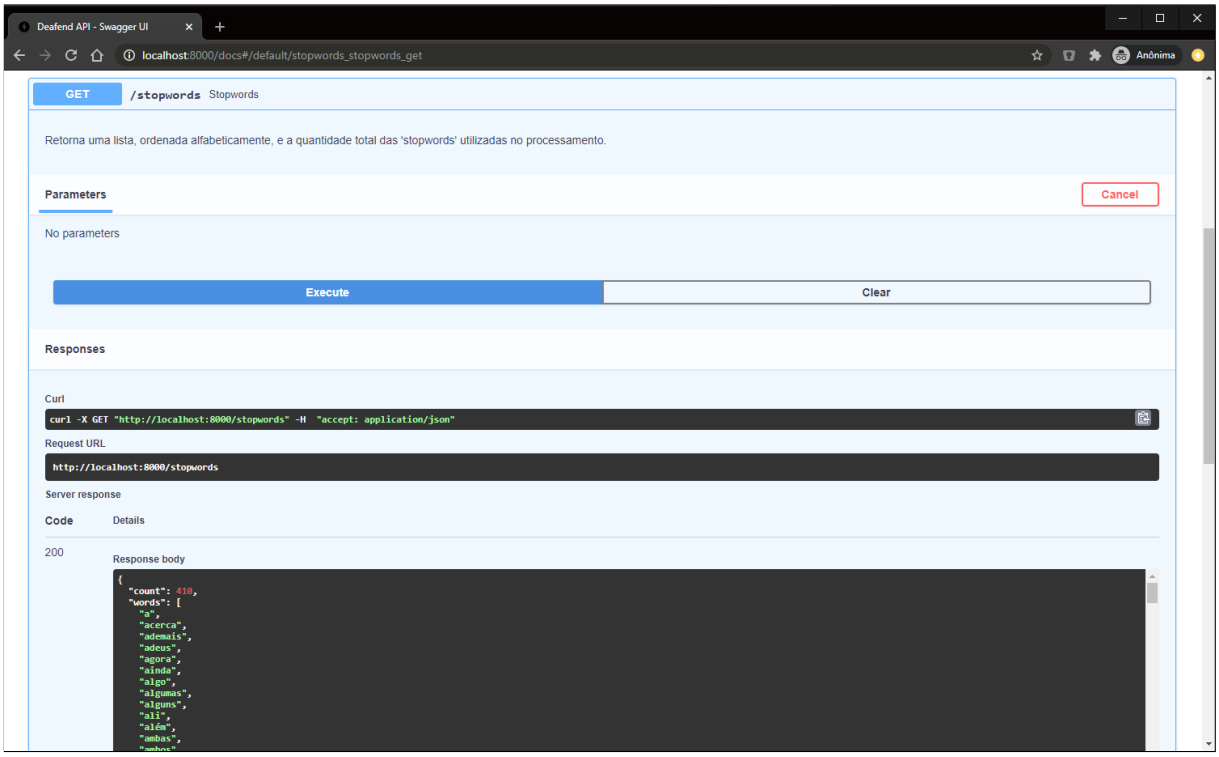
4.3.1.2 Processamento de dados

Os métodos `Stopwords` (`/stopwords`) e `Process Text` (`/process/{text}`) foram apartados para esta Seção pois o propósito de ambos é completamente diferente dos anteriores. Os dois são focados no processamento de linguagem natural e utilizam a biblioteca Python `spaCy` para executar, sem depender de conexão com a AWS ou realizar qualquer tipo de conversão. Com o intuito de não atentar-se exageradamente aos procedimentos desta biblioteca, mas sem deixar de destacar a relevância que possui no presente projeto, o Capítulo 5 irá dissertar sobre suas particularidades e apresentar uma referência de pesquisa para obter todos os detalhes de implementação e exemplos de funcionalidades da mesma.

Stopwords (ou palavras de parada) são palavras comuns que geralmente não contribuem para o significado de uma sentença, pelo menos para fins de recuperação de informações e processamento de linguagem natural. São palavras como *o/a* e *um/uma* (PERKINS, 2014, tradução do autor). O método `Stopwords`, no contexto desta API, apenas retorna o conjunto (propriedade `words`) e o total (propriedade `count`) de palavras consideradas desnecessárias para o PLN, servindo como uma ferramenta de conferência. A Figura 39 demonstra a execução deste

método através da documentação interativa e o início da resposta obtida, com destaque para o total de 410 termos que compõe a coleção de palavras de parada.

Figura 39 – API: Execução do método *Stopwords*



Fonte: Autor.

Já o método `Process Text`, pela responsabilidade efetiva de processar uma frase dentro da *pipeline* estabelecida de PLN, mostra-se como o mais impactante dos 7 disponibilizados na API deste trabalho. Inicialmente, uma instância da biblioteca `spaCy` é criada, enviando como parâmetro o texto recebido no valor da chave `text` da requisição executada. Tal parâmetro é aplicado sob um modelo da língua portuguesa já carregado e esta simples ação, automaticamente, executa o processo de *tokenização*. Este processo, que tem sua complexidade variando de acordo com a complexidade da própria linguagem, é caracterizado por dividir a sentença bruta em *tokens* significativos que, por sua vez, são a unidade mínima que uma máquina pode compreender e processar (HARDENIYA, 2015, tradução do autor).

Com o cumprimento desta etapa, a lista de *tokens* da sentença já está disponível. Sendo assim, é feito um laço de iteração sobre cada item, de modo que seja possível avançar para as próximas fases do processamento. Neste ponto, é interessante reavivar os conceitos evidenciados na Subseção 2.1.3, que expõe certas semelhanças entre a Libras e o Português, especialmente no que refere-se à estrutura gramatical. Tal característica revela que é possível fazer a transposição de frases simples - desconsiderando particularidades como a topicalização - sem modificar a organização da sentença.

Deste modo, optou-se por não alterar a frase recebida, mas apenas tratar os termos

variantes para que a busca no banco de dados, posterior ao processamento, possa ser executada com sucesso. Os termos variantes, na maioria das vezes, são os verbos. Por isso, na interação dos *tokens*, a primeira validação executada é confirmar se a categoria gramatical (ou *Part of Speech* - POS) é igual a `VERB`. Apesar da tarefa de diferenciar um advérbio de um adjetivo remeter à infância e, atualmente, parecer uma tarefa simples, codificar esse conhecimento em um modelo de *machine learning* foi uma tarefa muito complicada na área de PLN por várias décadas (HARDENIYA, 2015, tradução do autor). Felizmente, os algoritmos conhecidos como *POS Tagger* evoluíram muito e, no caso da biblioteca utilizada, esta categorização é incorporada ao *token* automaticamente.

Quando um verbo é reconhecido, executa-se o processo de lematização. Esta técnica mapeia as várias formas de uma palavra (como *apareceu, aparece*) para a forma canônica ou de citação da palavra, também conhecida como forma raiz ou *lema* (por exemplo, *aparecer*) (BIRD; KLEIN; LOPER, 2009, tradução do autor). Após substituir o *token* pelo seu *lema*, a checagem final é verificar se o *token* atual é um sinal de pontuação ou uma *stopword*. Apesar da *lib* `spaCy` fornecer o método `is_stop` (similar ao `is_punct`), que indica se o termo faz parte da "lista de parada", o conjunto utilizado no presente projeto possui pequenas customizações e, para garantir que não ocorram erros, uma comparação `não-está-em` é aplicada no sentido `termo -> lista`. O *token* que passar por estas validações, será adicionado ao *array* que é retornado como resposta do método. A Figura 40 apresenta o código responsável por todas as fases do processamento descritas.

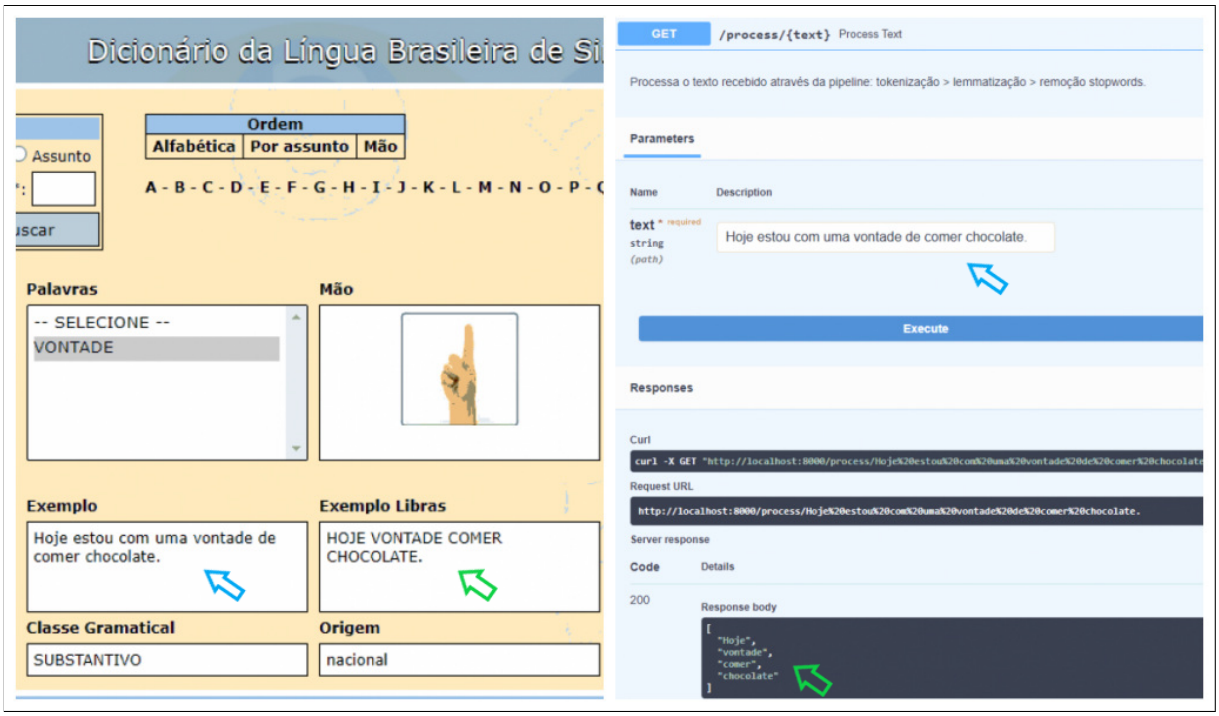
Figura 40 – API: Código-fonte do método *Process Text*

```python
@app.get("/process/{text}")
def process_text(text: str):
    """
    Processa o texto recebido através da pipeline: tokenização > lemmatização > remoção stopwords.
    """
    doc = nlp(text)
    print('## DEAFEND ## Tokens found: {}'.format(len(doc)))

    tokens = []
    for token in doc:
        value = token.text
        if token.pos_ == 'VERB':
            lemma = token.lemma_
            print('## DEAFEND ## Convert [{}] → [{}]'.format(value, lemma))
            value = lemma

        if value not in all_stopwords and not token.is_punct:
            tokens.append(value)
        else:
            print('## DEAFEND ## Removed stopword/punct → [{}]'.format(value))

    return tokens
```

Fonte: Autor.

Aproveitando-se dos exemplos de frases na língua brasileira de sinais e na língua portuguesa que são apresentados no dicionário utilizado como fonte dos GIFs deste projeto, é possível realizar uma conferência de resultados. Selecionando as sentenças vinculadas à palavra `VONTADE`, por exemplo, o *site* exibe: *"Hoje estou com uma vontade de comer chocolate."* (Português) e *"HOJE VONTADE COMER CHOCOLATE."* (Libras). O outro ângulo da comparação é invocar o método `Process Text` através da documentação interativa e enviar o parâmetro `texto` com a sentença na língua portuguesa que deseja-se processar. A Figura 41 é uma montagem que apresenta, lado a lado, os dois cenários propostos. Importante ressaltar que os registros do dicionário foram preparados manualmente dentro de um escopo limitado, enquanto o PLN disponibilizado através da API poderá lidar com infinitas frases nessa mesma estrutura.

Figura 41 – API: Comparação de resultados



Fonte: Autor.

4.3.2 Aplicativo

Antes de criar o aplicativo React Native, é necessário garantir que o `Node.js` está instalado na máquina. Caso ele esteja instalado, automaticamente o gerenciador de pacotes `npm` - utilizado nesta aplicação - também estará e, através dele, será possível fazer a adição das bibliotecas necessárias para o trabalho. Para realizar esta conferência, deve-se executar alguns comandos no *Terminal*:

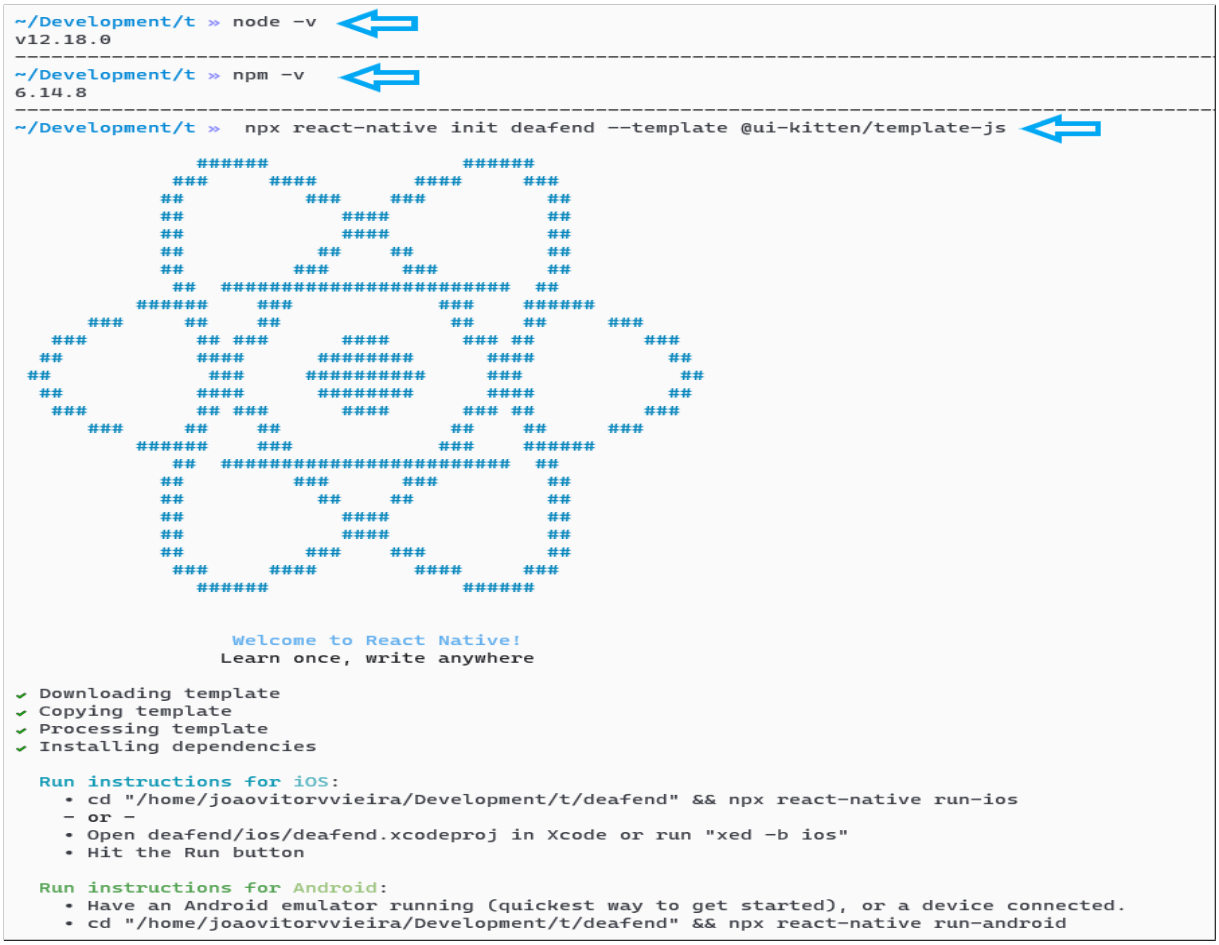
```
$ node -v
$ npm -v
```

Caso o resultado dos dois comandos exibidos apresente um resultado como `v12.18.0` e `6.14.8`, respectivamente, significa que os arquivos binários do `Node` e do `npm` estão instalados com sucesso. Antes de seguir para a próxima etapa, é válido destacar que as versões citadas podem não ser exatamente iguais às referidas anteriormente, pois a própria versão *long-term support* (LTS) - considerada estável - passou por uma atualização durante a criação deste *app*. Entretanto, desde que não exista nenhum *upgrade* que torne as funcionalidades anteriores incompatíveis com a nova versão, uma instalação acima da `v12.x.x` vai funcionar corretamente.

No momento de iniciar o projeto, é importante atentar-se para a possibilidade de indicar o *template* que será adotado. Esta é a alternativa ideal para realizar a instalação da *lib* `UI Kitten` que, como já citado, é a responsável pelo *layout* e parte visual desta aplicação móvel. Para tanto, é necessário executar um comando que garante a disponibilidade do `React Native CLI` (utilitário de linha de comando do *framework*) e outro que, de fato, gera os arquivos do novo sistema em um diretório de preferência do desenvolvedor:

```
$ npm un -g react-native-cli && npm i -g @react-native-community/cli
$ npx react-native init <nome_projeto> --template @ui-kitten/template-js
```

Figura 42 – Aplicativo: Criação do projeto



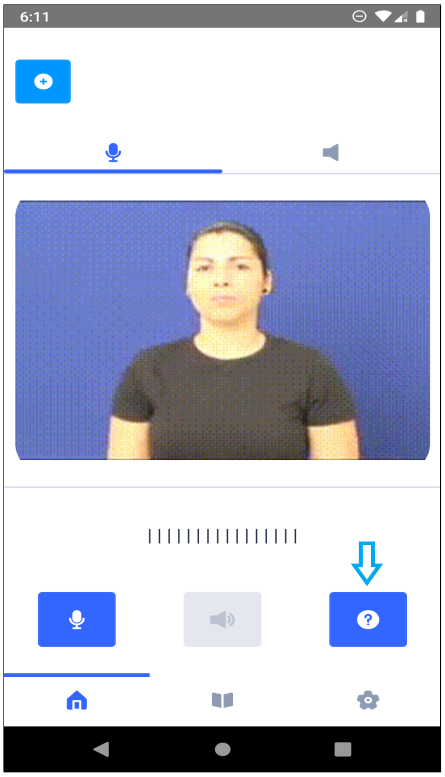
Fonte: Autor.

A partir deste momento, a base para o desenvolvimento da aplicação está pronta. Portanto, as seções seguintes serão delimitadas pelos três menus do aplicativo - Página Inicial (*Home*), Glossário (*Glossary*) e Configurações (*Settings*) - de modo a facilitar a compreensão e, também, permitir um detalhamento completo das especificidades de cada um.

4.3.2.1 Página Inicial

Esta é, sem dúvidas, a parte mais importante da solução final, uma vez que engloba as funcionalidades de reconhecimento de fala, emissão de resposta e criação de atalhos. Sendo assim, é nesta tela que, de fato, pode ocorrer a comunicação entre surdo e ouvinte. Porém, o primeiro item que será destacado é o *Botão de ajuda* - evidenciado na Figura 43a - que estará sempre habilitado enquanto este menu estiver em foco. Ao clicar neste botão, a página *web* (https://deafend-documentation.vercel.app/) que contém a documentação feita para o usuário final será aberta no navegador padrão do dispositivo (demonstrado na Figura 43b). Além disso, cada menu possui uma página dedicada especificamente a explicar as possibilidades de uso e funcionalidades que ele contém e, portanto, a seção referida conterá uma nota de rodapé[2] com o *link* para este apontamento.

Figura 43 – Aplicativo: Acesso à documentação



(a) Botão de acesso

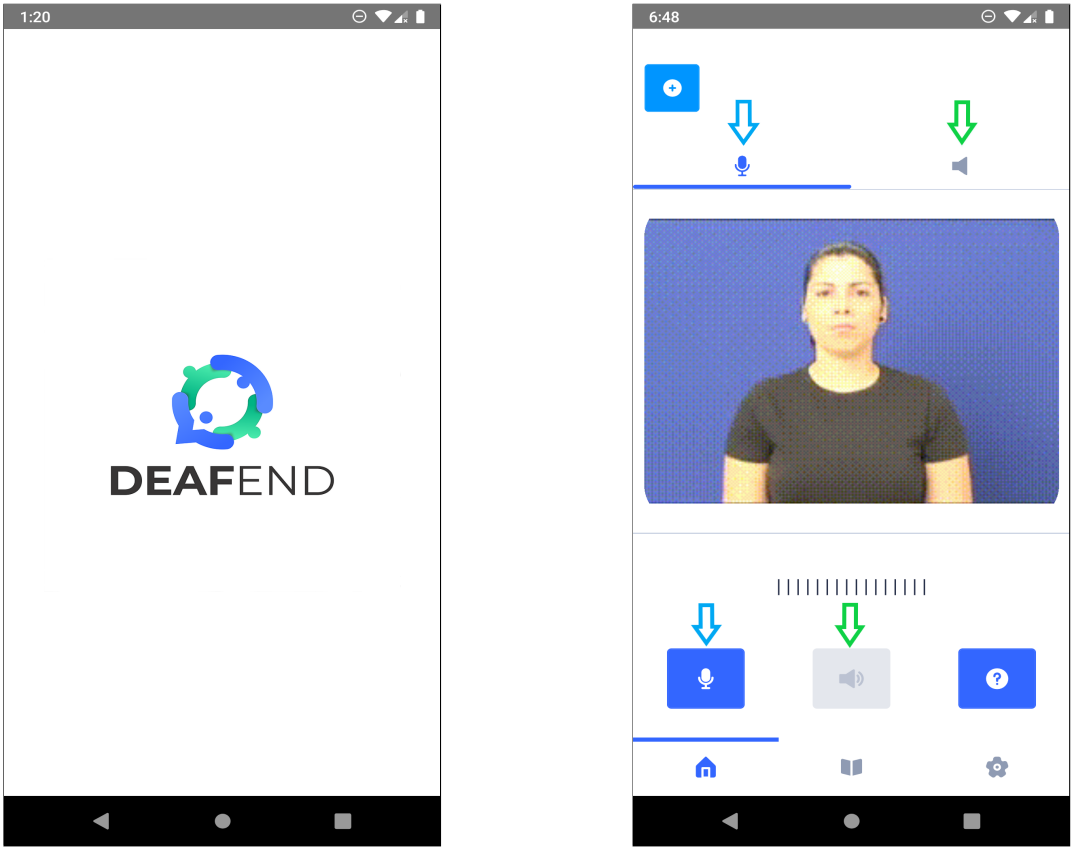
(b) Documentação *online* do usuário

Fonte: Autor

[2] `Documentação - Página Inicial:` https://deafend-documentation.vercel.app/docs/pagina-inicial

Ao abrir o aplicativo, o logo do mesmo será exibido até que as funções disponíveis estejam prontas para uso - comportamento conhecido como *splash screen* e exibido na Figura 44a. Dado que a Página Inicial é a primeira rota acessada ao abrir a aplicação, o usuário estará automaticamente no menu descrito nesta Seção. Sendo assim, será possível observar que a tela contém duas abas: de *reconhecimento* (ícone de microfone) e de *emissão* (ícone de alto-falante), ambas expostas na Figura 44b. As duas estão diretamente ligadas aos botões da parte inferior da tela com ícones semelhantes e, por isso, sempre que um deles for pressionado, a respectiva aba ficará ativa.

Figura 44 – Aplicativo: Acesso inicial



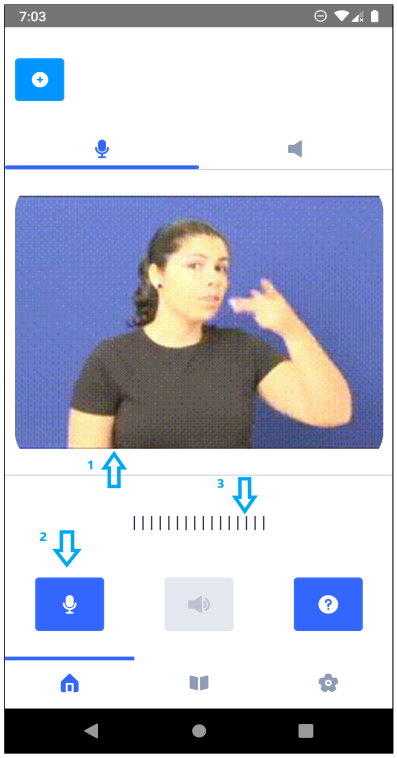
(a) *Splash screen* (b) Abas da Página Inicial

Fonte: Autor

Cada uma das abas citadas pode alterar o conteúdo do painel central da tela conforme é conveniente. Na aba de *reconhecimento*, por exemplo, este conteúdo, inicialmente, é preenchido por um GIF onde a intérprete está sinalizando a palavra *FALAR* - forma de motivar o usuário a utilizar a funcionalidade. Neste mesmo momento, o *Botão de reconhecimento* encontra-se habilitado pois, exceto quando o aplicativo estiver emitindo algum som, a opção de distinguir uma locução estará sempre disponível. Também, o componente de ondas sonoras está demonstrando que não há sons entrando ou saindo do controle do *app* - a alteração visual deste componente, que será exibida ao longo deste detalhamento, é muito importante para conscientizar o usuário

surdo sobre ocorrências sonoras. Todas estas condições estão, respectivamente, sinalizadas na Figura 45.

Figura 45 – Aplicativo: Estrutura da aba *reconhecimento*



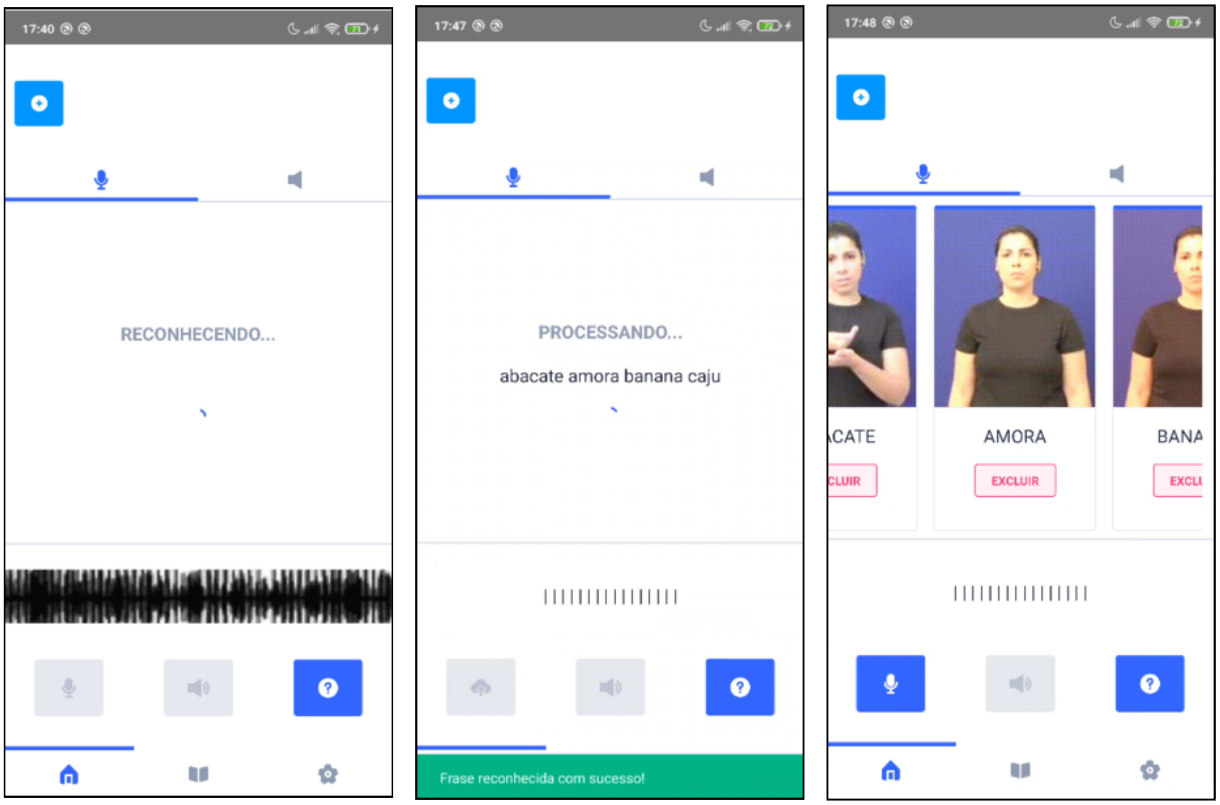
Fonte: Autor.

Quando o *Botão de reconhecimento* é pressionado, o serviço disponibilizado pela biblioteca `@react-native-community/voice` é ativado, definindo a língua que será utilizada como `pt-BR`. O comportamento desta *lib* é orientado a eventos - tais como `onSpeechStart`, `onSpeechPartialResults` e `onSpeechError` - além da disponibilização do *status* do serviço - `isAvailable` e `isRecognizing`, por exemplo. Desta forma, o aplicativo possui um controle de etapas da recognição e, quando o serviço é inicializado, a etapa vigente é `RECORDING`. Neste instante, o GIF com a intérprete é substituído pela palavra *RECONHECENDO* e, enquanto uma fala for detectado, os resultados parciais do reconhecimento serão mostrados em tempo real no painel. Caso o serviço identifique um silêncio de aproximadamente 5 segundos, encerra-se o processo de obtenção e o fluxo segue para a etapa `PROCESSING`.

Nesta fase, a biblioteca citada encerra sua execução disponibilizando a frase reconhecida. Com essa *string* à disposição, é possível invocar o método `Process Text` da API - descrito na Seção 4.3.1.2 - enviando-a como valor para ser processado. Tendo em vista que a resposta da API será um *array* com os termos nos quais empregou-se as técnicas da PLN, é possível basear-se na premissa de que este retorno está qualificado para ser utilizado como parâmetro da *query* no banco de dados. Sendo assim, uma consulta na tabela `words` do Amazon DynamoDB é realizada, buscando todos os registros em que a propriedade `text` *está contida*

no conjunto de termos provindos da resposta da API. Obtendo estes registros, é possível acessar a propriedade `gifURL` de cada um e, em ordem, sobrepor o texto reconhecido pelos GIFs correspondentes. A Figura 46 retrata o comportamento visual do aplicativo durante a computação descrita.

Figura 46 – Aplicativo: Fluxo de *reconhecimento*



(a) Etapa 1: reconhecer (b) Etapa 2: processar (c) Etapa 3: exibir

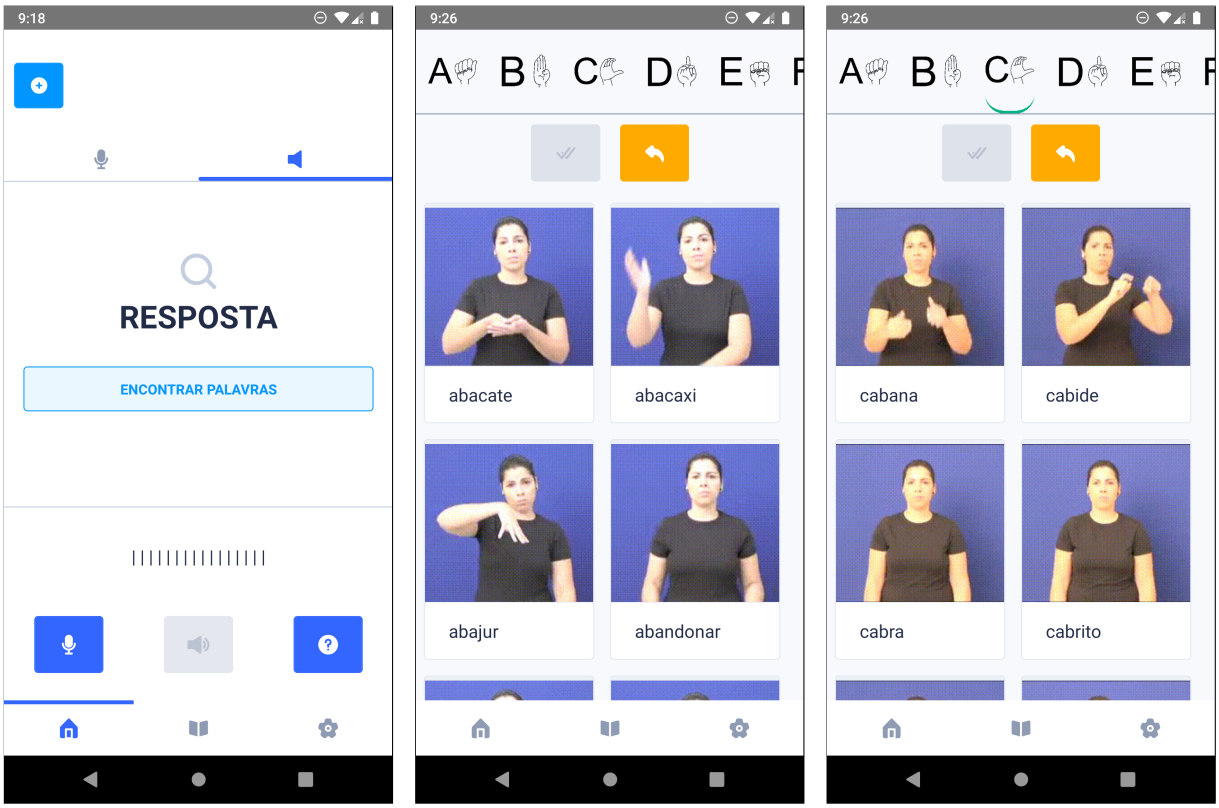
Fonte: Autor.

É interessante destacar a mudança visual que é causada pela ativação do componente de ondas sonoras (Figura 46a). Como já mencionado, este recurso será acionado todas as vezes que o aparelho estiver emitindo ou reconhecendo um som, almejando indicar ao usuário que a operação está ocorrendo com sucesso. Neste exemplo, essa movimentação indica que o aplicativo está aguardando ou já está reconhecendo o que está sendo pronunciado. Existem outros dois momentos - a utilização de um atalho ou a emissão de uma resposta - que ativam este recurso e que serão descritos adiante neste documento.

A função de *emissão* pode ser descrita como o oposto do *reconhecimento* - apesar de apresentar certas similaridades - uma vez que o *input* ocorre pela Libras e o *output* pelo Português. Inicialmente, o conteúdo disposto no painel dessa aba é composto por um ícone de pesquisa com a descrição *Resposta* e um botão com o rótulo *Encontrar palavras* (exibido na Figura 47a). Ao clicar neste botão, o usuário é levado para a tela em que ele poderá escolher as palavras que formarão a sentença para ser emitida. Com o intuito de facilitar esta construção,

existe um filtro na parte superior da tela que consiste em uma lista horizontal com todas as letras do alfabeto. Ao pressionar qualquer uma, um sinal será exibido para deixar claro que o filtro está sendo aplicado e apenas as palavras que iniciarem com esta letra passarão a ser exibidas no aplicativo. As Figuras 47b e 47c retratam este comportamento.

Figura 47 – Aplicativo: Construção da resposta com opção de filtro por letra



(a) Conteúdo inicial da aba (b) Listagem completa (c) Filtro pela letra “C” ativo

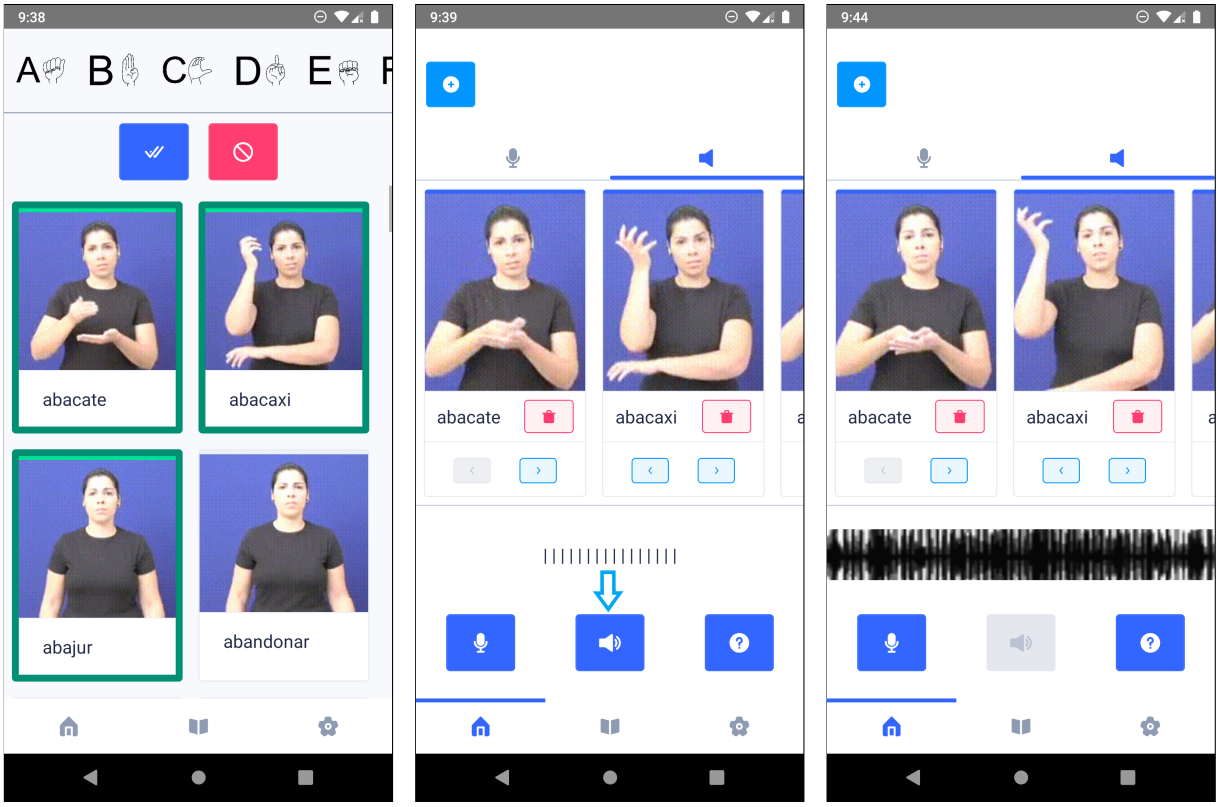
Fonte: Autor.

Todas os termos que são listados nesta tela provém do resultado de uma consulta livre (sem filtros) no Amazon DynamoDB. Cada palavra será exibida em um *card* próprio e, para adicioná-la na sentença que está sendo montada, basta clicar sobre a mesma. Nesse momento, uma borda verde será posta ao redor deste quadro, identificando que aquela palavra já está marcada (Figura 48a). Uma observação relevante sobre este processo é que as palavras serão incorporadas à resposta à medida que são escolhidas, o que favorece o modo como é construída uma frase (sequencialmente). Após ter selecionado qualquer vocábulo, o *Botão escolher* e o *Botão desmarcar* serão habilitados, pois são as duas alternativas que o usuário possui neste momento. Ao pressionar o *Botão desmarcar* (vermelho), como o próprio nome já sugere, todas as palavras selecionadas serão desmarcadas.

Porém, ao utilizar o *Botão escolher* (azul), o usuário será redirecionado para a Página Inicial e o conteúdo do painel apresentará a frase construída. No momento que este retorno automático ocorre, o *Botão de emissão* passa a estar habilitado, como evidencia a Figura 48b.

Na mesma imagem, é possível notar que existem alguns botões abaixo de cada GIF que constitui o conteúdo do painel. Prevendo possíveis erros ou pequenos ajustes que podem ocorrer após ter montado a frase, estes botões fornecem a opção de remover determinado termo da sentença ou alterar seu posicionamento (para frente e para trás). Dessa forma, o usuário não necessita realizar todo o procedimento novamente. Quando a frase estiver concluída, as palavras que a constituem podem ser exteriorizadas através dos alto-falantes do dispositivo apenas pressionando o *Botão de emissão*. Neste momento, conforme já relatado, as ondas sonoras serão ativadas (Figura 48c), indicando que o processo de emissão está em andamento.

Figura 48 – Aplicativo: Montagem da resposta e emissão



(a) Seleção de palavras (b) Botão de *emissão* habilitado (c) Frase em processo de emissão

Fonte: Autor.

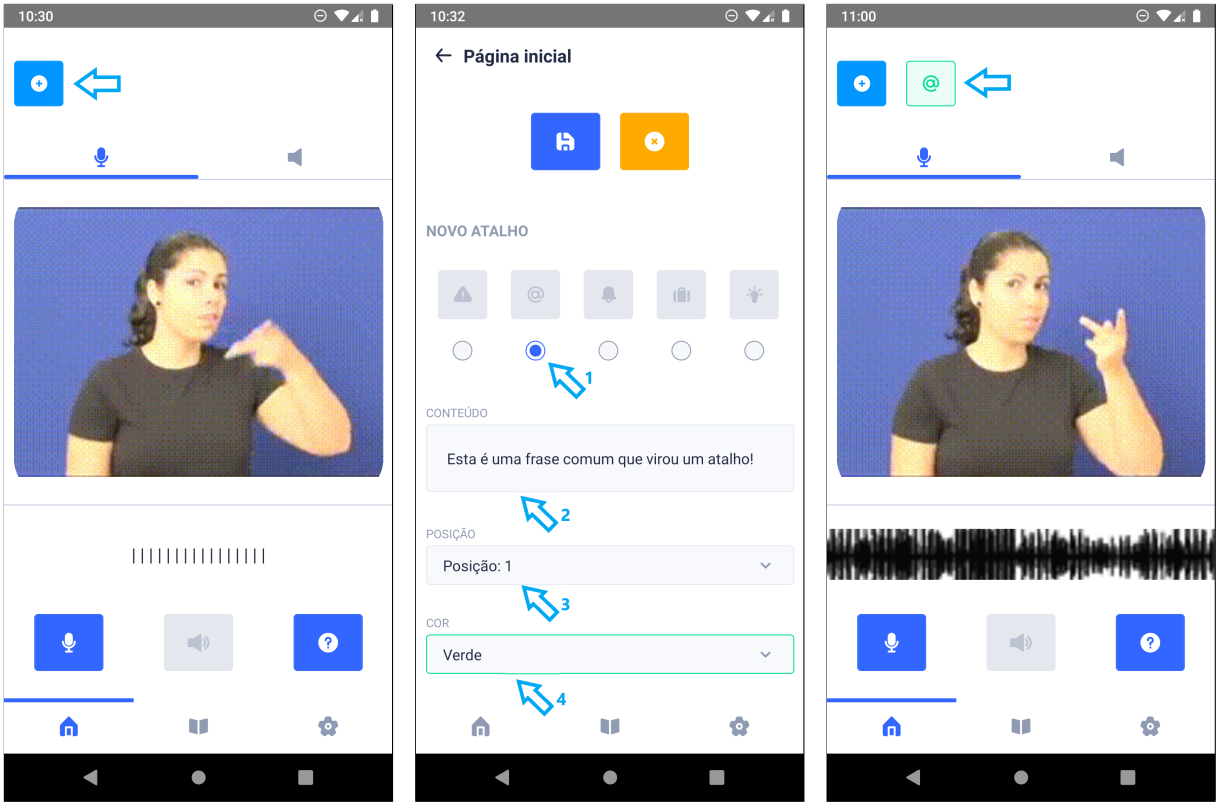
Ainda neste menu, existe outra forma de emitir mensagens: os botões de atalho. Esta funcionalidade foi criada com a intenção de disponibilizar uma forma rápida de expressar uma sentença muito comum ou frequentemente utilizada no dia-a-dia do usuário. Sendo assim, o botão localizado na parte superior esquerda (evidenciado na Figura 49a) redireciona o utilizador para a tela que permite criar um atalho. Essa criação ocorre por meio do preenchimento de um formulário que possibilita a seleção de um ícone, a digitação do conteúdo que será emitido e a escolha da posição e da cor que o futuro botão deverá apresentar (Figura 49b).

Assim que o *Botão salvar* for pressionado, uma mensagem de *feedback* de sucesso é exibida e, ao retornar para a Página Inicial, haverá um novo botão, com as características

definidas sendo apresentado no aplicativo (Figura 49c). Caso ele seja pressionado, o conteúdo será exteriorizado tal qual a emissão de uma resposta e as ondas sonoras serão exibidas. Esta funcionalidade também utiliza a *lib* `@react-native-community/async-storage` - abordada na Subseção 2.5.5 - que viabiliza o armazenamento de registros chave-valor localmente no dispositivo. Para este tópico, a chave escolhida foi `@Deafend_shortcuts` e o valor arquivado consiste em uma estrutura JSON conforme apresentado na sequência:

```
[
  { // Objeto representando o atalho criado
    "content": "Esta é uma frase comum que virou um atalho!",
    "status": "success",
    "iconId": 1,
    "position": 0,
  }
]
```

Figura 49 – Aplicativo: Criação de atalho



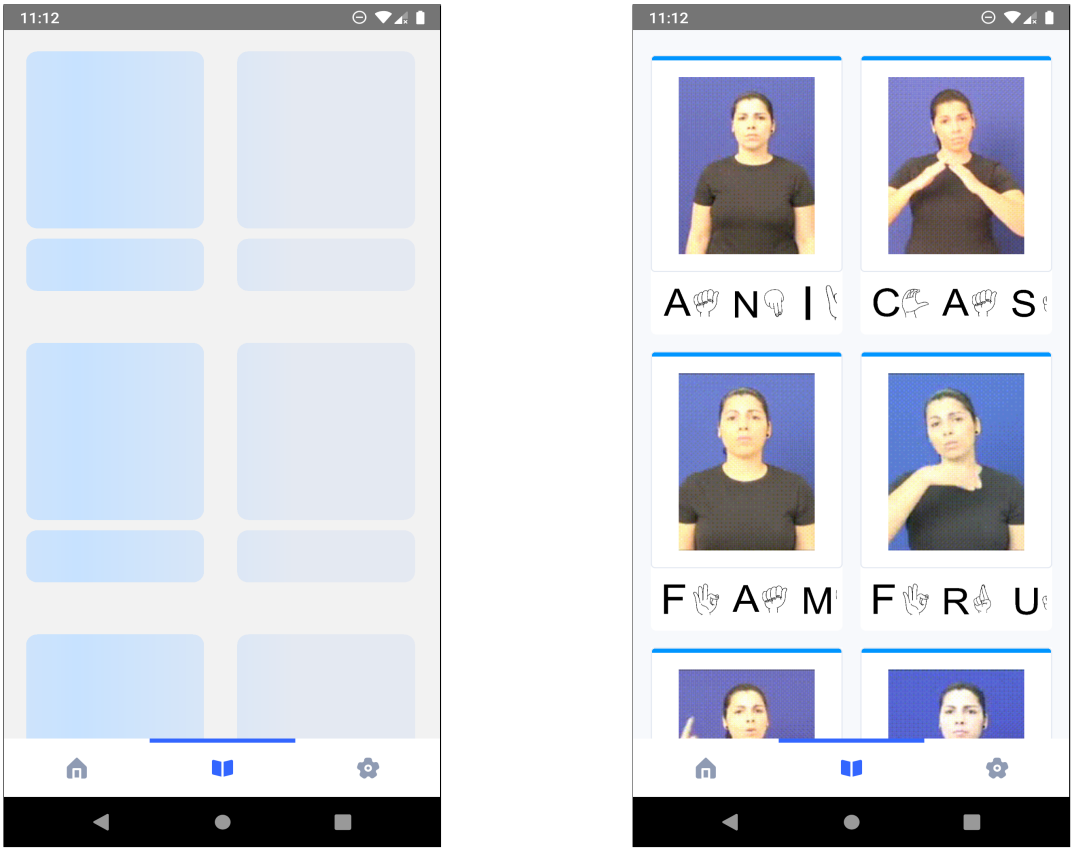
(a) Botão de redirecionamento (b) Campos do formulário (c) Atalho criado

Fonte: Autor.

4.3.2.2 Glossário

Esta tela[3], que pode ser acessada através da opção central do menu, pode ser identificada como puramente informativa, já que sua função é fornecer ao usuário todos os dados disponíveis no aplicativo sobre qualquer sinal em Libras que ele deseje conhecer. Logo que a tela é aberta, uma *query* na tabela `subjects` do Amazon DynamoDB é realizada para que seja possível listar todos os assuntos cadastrados na base de dados. Tendo em vista que a velocidade desse retorno depende de variáveis, tais como a capacidade do dispositivo e a velocidade da conexão utilizada, utilizou-se uma técnica conhecida como *Skeleton Loading* - exposta na Figura 50a. Esta estratégia é um padrão de experiência do usuário que reduz a frustração com o tempo de carregamento (DOLPHIN, 2018, tradução do autor). Quando a resposta é obtida, o conteúdo temporário é imediatamente substituído por *cards* individuais dos assuntos (Figura 50b).

Figura 50 – Aplicativo: Exibição da lista de assuntos



(a) *Skeleton Loading* (b) Carregamento concluído

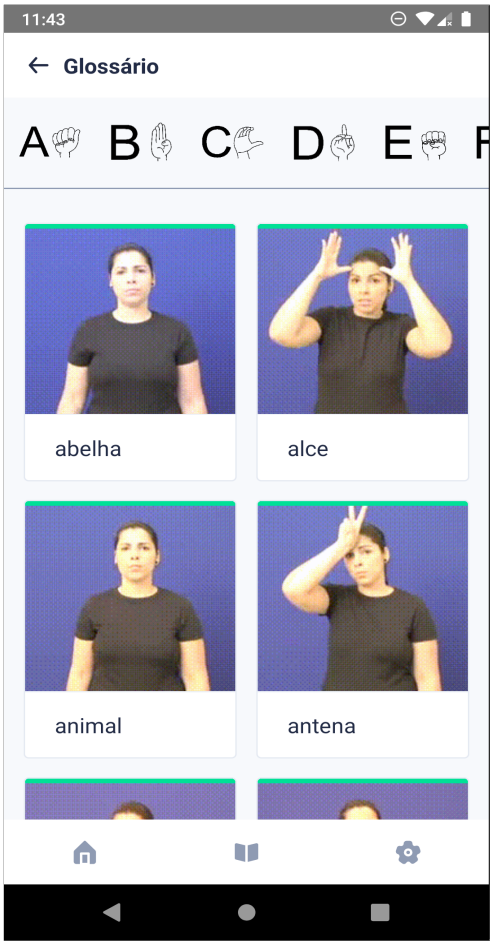
Fonte: Autor

Seguindo o padrão adotado em todo o *app*, caso a descrição do assunto - que é exibida letra por letra, acompanhada do seu respectivo sinal em Libras - não seja comportada pela largura do *card*, basta deslizá-la horizontalmente para visualizar o conteúdo oculto. Ao clicar em

[3] `Documentação - Glossário:` https://deafend-documentation.vercel.app/docs/glossario

qualquer assunto, outra consulta é disparada ao Amazon DynamoDB, porém esta é direcionada para a tabela `words`, filtrando todos os registros que contêm a propriedade `subject` igual ao valor `text` do assunto escolhido. Exemplificando, ao clicar no cartão *ANIMAL*, o retorno da busca no banco de dados será o conjunto de todos os termos que constituem este assunto. Com estes valores, é possível redirecionar o usuário para a tela de *Detalhes do assunto*, conforme exposto na Figura 51.

Figura 51 – Aplicativo: Detalhes do assunto

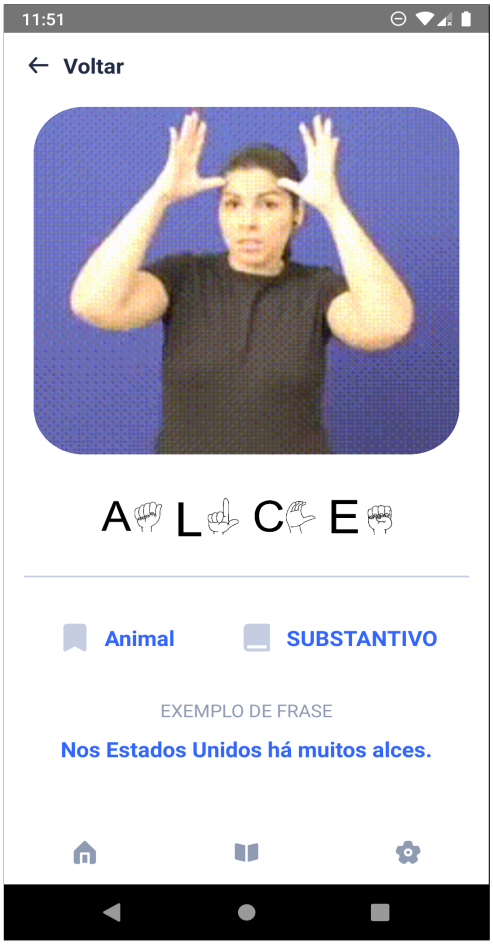


Fonte: Autor.

Ainda visualizando a Figura 51, é possível perceber sua semelhança com a tela de *Construção da resposta*. De fato, a estrutura de ambas consiste no filtro por letra sendo exposto na parte superior e o restante do conteúdo sendo preenchido por uma lista de palavras, com a diferença de que, neste caso, o escopo limita-se às palavras do assunto selecionado. Caso o usuário queira visualizar mais informações sobre uma palavra específica, é possível pressionar o *card* da mesma para acessar a tela de *Detalhes da palavra* - demonstrada na Figura 52. Nela, além do GIF com o sinal, da datilologia e do assunto da palavra, estão contidas algumas propriedades exclusivas, tais como a classe gramatical daquele termo e um exemplo de uso em uma frase real. Importante salientar que estas informações foram obtidas do Dicionário da Língua

Brasileira de Sinais V3 e inseridas na tabela `words` do Amazon DynamoDB durante a pré-carga realizada, descrita na subdivisão 4.3.1.1 do presente documento.

Figura 52 – Aplicativo: Detalhes da palavra



Fonte: Autor.

4.3.2.3 Configurações

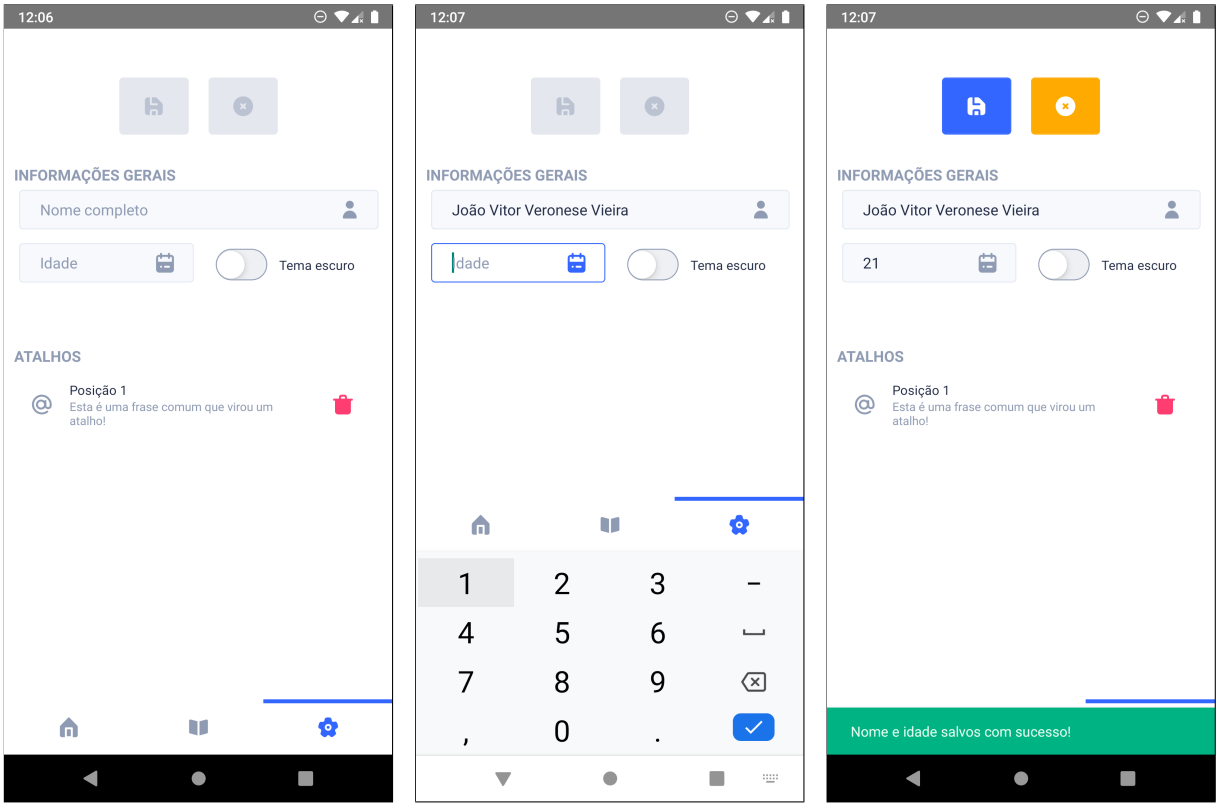
Diferentemente da anterior, a tela de Configurações[4] apresenta um teor mais decisório, uma vez que suas interações remetem à escolhas. É neste local que as informações pessoais do usuário são preenchidas e salvas, que o tema do aplicativo é alterado e que os atalhos - criados na Página Inicial - são listados e, também, excluídos. A Figura 53 apresenta um panorama geral, utilizando o tema *light*, sobre como tais tópicos estão dispostos.

Os dois botões exibidos na parte superior são exclusivamente referentes aos *inputs* *nome completo* e *idade* e, sendo assim, o estado inicial de ambos é desabilitado (como mostra a Figura 53a), pois não há conteúdo dentro dos campos citados. A partir do momentos que os dois campos estiverem preenchidos, tanto o *Botão salvar* quanto o *Botão limpar* são, automaticamente, habilitados (exibido na Figura 53c). Já que estes valores são individuais e não

[4] `Documentação - Configurações:` https://deafend-documentation.vercel.app/docs/configuracoes

são utilizados por outras funcionalidades, considerou-se conveniente salvá-los localmente no dispositivo do usuário.

Figura 53 – Aplicativo: Preenchimento de informações pessoais



(a) Estado inicial dos botões (b) *Inputs* sob validação (c) Botões habilitados

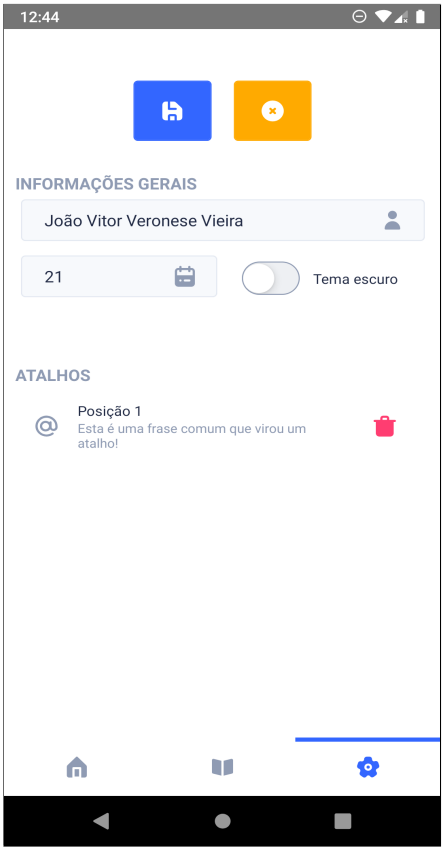
Fonte: Autor.

Para tanto, a biblioteca `@react-native-community/async-storage` é utilizada e a chave utilizada para armazenar a estrutura JSON, disposta na sequência, é `@Deafend_data`.

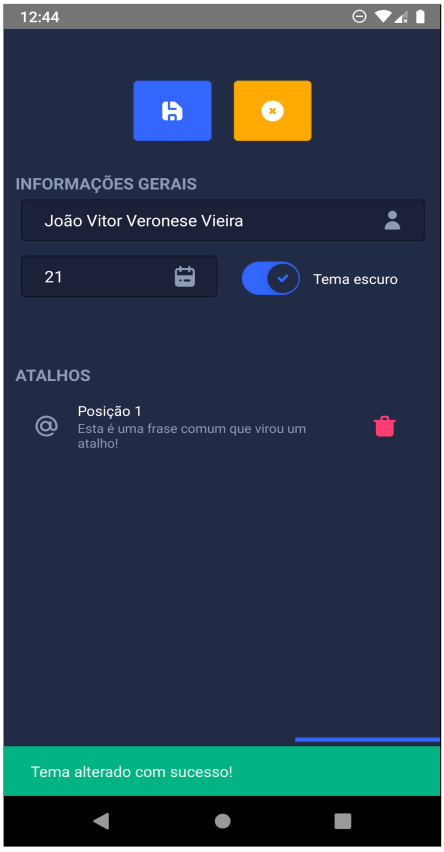
```
{
  "name": "João Vitor Veronese Vieira",
  "age": 21,
}
```

Apesar de serem completamente independentes entre si, a chave liga/desliga também salva seu valor (*light* ou *dark*) neste banco local, porém na chave `@Deafend_theme`. Uma observação interessante é que o aplicativo é compatível com a funcionalidade *tema* nativa dos aparelhos. Por isso, caso o usuário já esteja com o modo escuro habilitado em seu dispositivo e abra o *app*, esta propriedade será reconhecida e, automaticamente, o tema *dark* será definido como padrão. A Figura 54 exibe, lado a lado, as diferenças visuais entre cada uma das alternativas.

Figura 54 – Aplicativo: Temas *light* e *dark*



(a) Tela Configurações no modo *light*



(b) Tela Configurações no modo *dark*

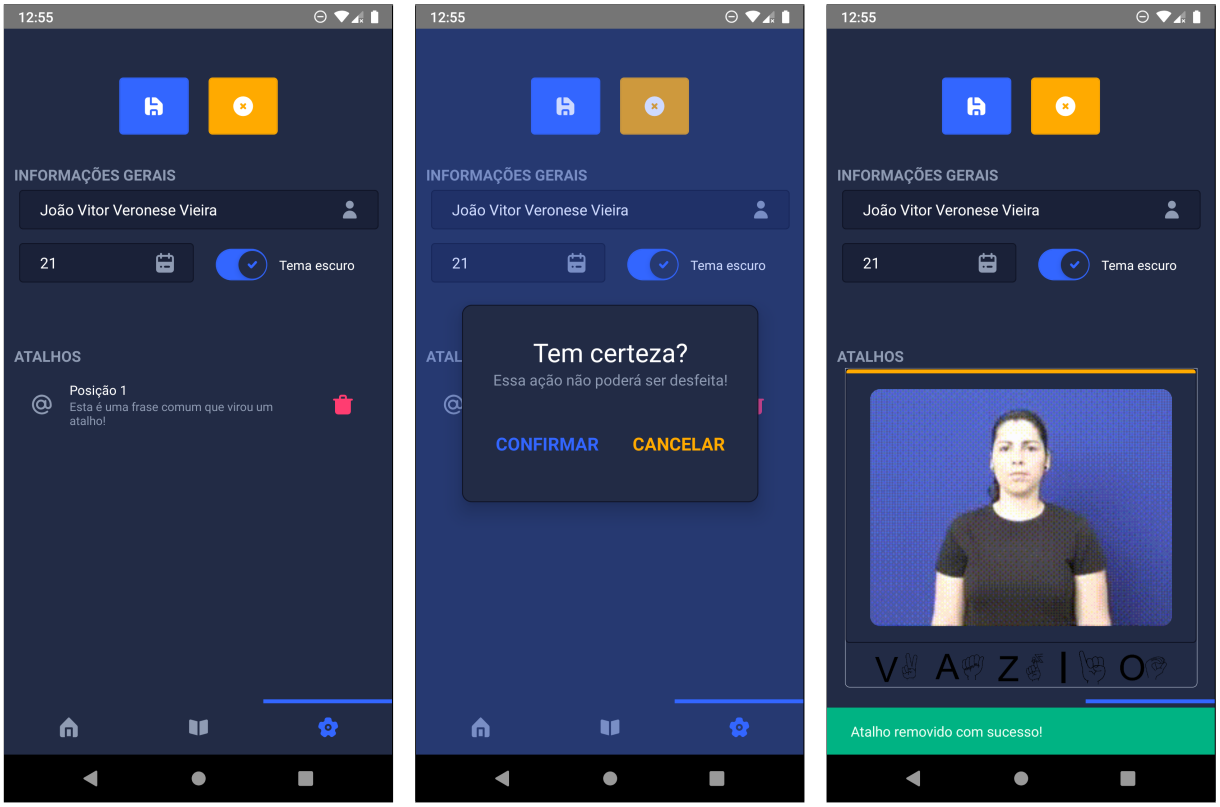
Fonte: Autor

E, por fim, nota-se a listagem de atalhos, disposta na Figura 55a. Por praticidade e buscando evitar ao máximo que qualquer informação adicional atrapalhasse os principais processos da aplicação móvel - reconhecimento e emissão - optou-se por permitir que estes itens sejam excluídos na tela de Configurações (mesmo que sua criação ocorra na Página Inicial). Cada elemento que constitui a lista possui, na parte direita da linha, um ícone de lixeira que, ao ser pressionado, abre um *modal* (Figura 55b). Este componente questiona o usuário se ele deseja, realmente, excluir aquele atalho. Caso o *click* tenha ocorrido por engano ou o utilizador desista de excluí-lo, basta pressionar em *cancelar* que o *modal* será fechado e nenhuma ação será realizada.

Caso contrário, a exclusão é efetivada. E, para que a chave `@Deafend_shortcuts` seja atualizada com o valor correto, é necessário realizar uma reordenação da lista de atalhos e alterar a propriedade `position` de cada item, pois este atributo é utilizado para disponibilizar a lista de alternativas de posição no momento de criar outro atalho. Quando esta computação é finalizada e a atualização no banco ocorre, a lista exibida é sincronizada. Se, como é o caso da situação retratada na Figura 55c, o atalho excluído era o último elemento, o espaço passa a ser preenchido por um GIF que representa o sinal *VAZIO*. Além de manter o padrão do *app*, exibir

a imagem em detrimento de um texto, por exemplo, favorece o sujeito surdo que, geralmente, possui dificuldades com a interpretação da língua portuguesa.

Figura 55 – Aplicativo: Exclusão de atalho



(a) Lista de atalhos (b) Confirmação de exclusão (c) Atalho excluído

Fonte: Autor.

5 RESULTADOS E DISCUSSÕES

Inicialmente, este documento discorreu sobre a opinião de diversos autores, trouxe o embasamento teórico acerca da surdez e descreveu como os dispositivos móveis e seus recursos têm impactado toda a sociedade. Em seguida, relatou-se a existência de *softwares* que atuam especificamente na extensão que situa-se este trabalho. Logo após, explanou-se a fundação técnica do aplicativo construído, as bibliotecas externas que viabilizam seu funcionamento e como cada fragmento da solução comunica-se entre si. Desse modo, mostra-se oportuno conceber as interligações existentes entre estas seções e apresentar quais foram as contribuições geradas por este projeto, propósito deste Capítulo.

5.1 Prova de conceito

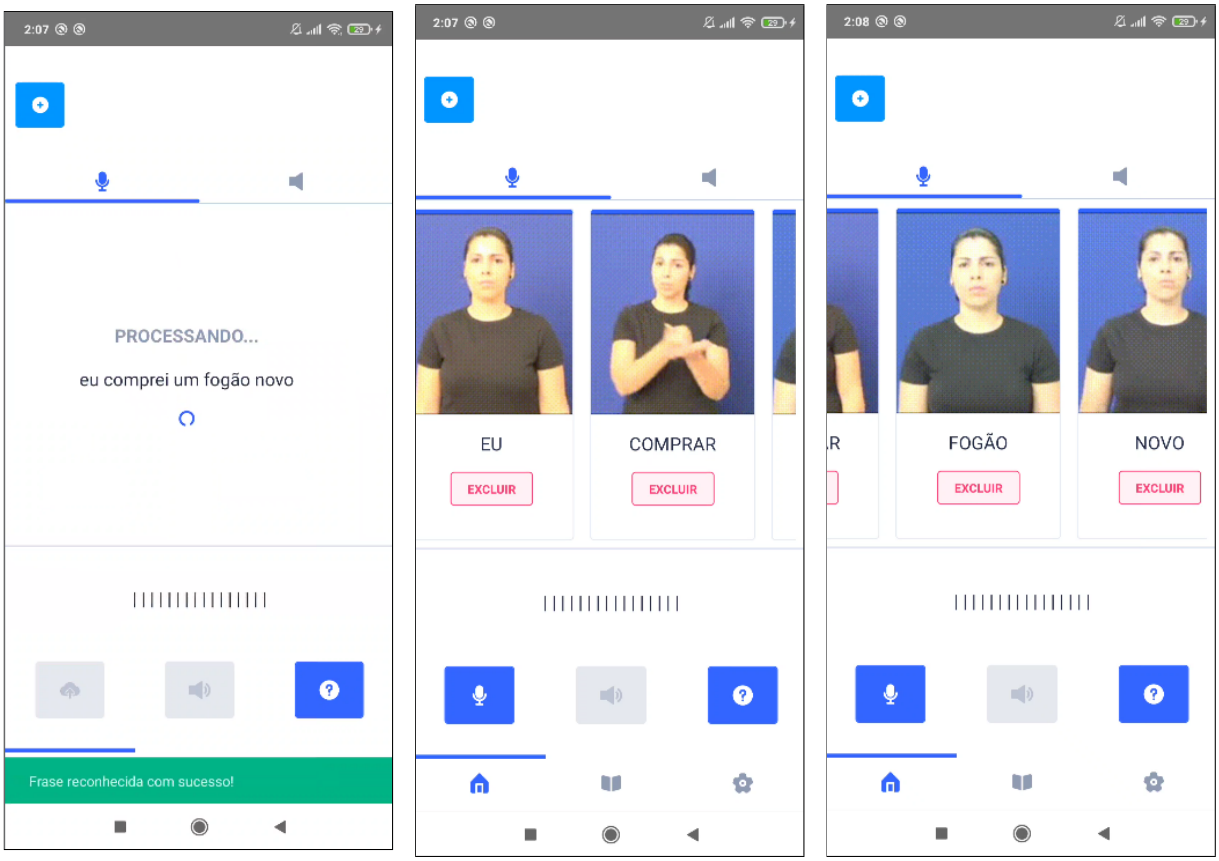
Com base no objetivo principal desta atividade, o primeiro tópico que deve ser ressaltado é o estreitamento entre as línguas brasileira de sinais e portuguesa. Conforme exposto na Seção 2.1.3, os dois idiomas podem ser considerados similares, uma vez que a estrutura gramática principal é sujeito-verbo-objeto (SVO). Este é um conceito-chave que favoreceu a trajetória até a versão final do sistema, pois foi possível migrar o tempo que seria investido na adaptação da ordem dos termos em outras *features* - tais como a emissão de resposta e a criação de atalhos.

Figura 56 – Exemplo 1: *EU COMPREI UM FOGÃO NOVO.*

Fonte: Autor.

Procurando facilitar o entendimento, além de fornecer mais clareza à conferência entre resultados, a Figura 56 recorre ao Dicionário da Língua Brasileira de Sinais V3 para servir-se de um exemplo de frase (destacada na imagem). Na sentença afirmativa escolhida - *EU COMPREI UM FOGÃO NOVO.* - percebe-se que existem diversas classes gramaticais, tais como sujeito (*EU*), verbo (*COMPREI*) e adjetivo (*NOVO*). Tais variedades aumentam a dificuldade do processamento de linguagem natural em virtude da quantidade de cenários que tornam-se viáveis com suas combinações. Apesar disto, seria importante que esta pluralidade fosse suportada pelo aplicativo, para que o objetivo traçado seja alcançado. Para fins de análise de exatidão, pronunciou-se a mesma frase colhida do dicionário utilizando a funcionalidade de reconhecimento de fala. O resultado obtido é exibido na Figura 57.

Figura 57 – Exemplo 1: Abordagem **DeafEnd**



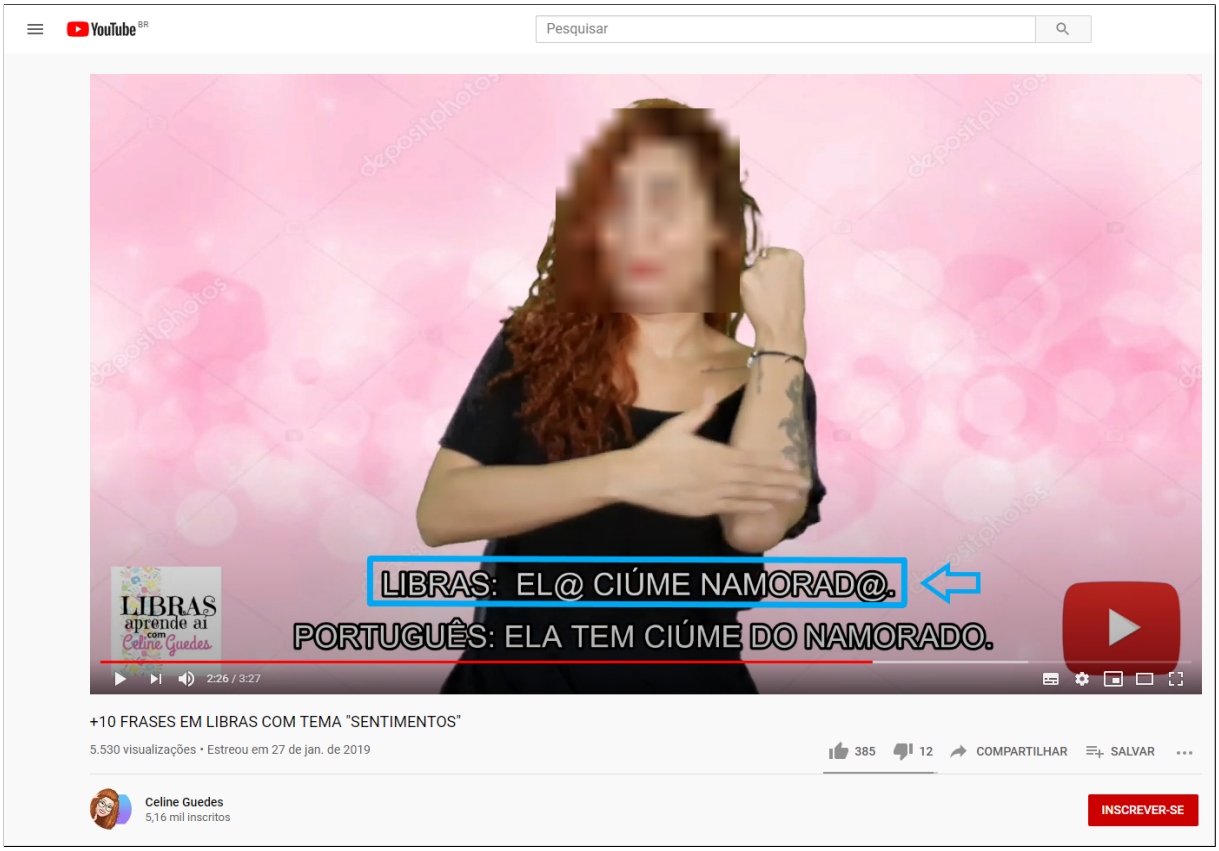
(a) Processando frase (b) Primeira parte da exibição (c) Segunda parte da exibição

Fonte: Autor.

Ao visualizar os GIFs exibidos no painel da página inicial, é possível contemplar as mesmas palavras, na mesma posição, que a frase tomada como modelo. Vale ressaltar que, enquanto o material disponibilizado no dicionário foi inserido manualmente (através de uma extensa pesquisa financiada pelo Ministério da Ciência, Tecnologia e Inovação do país), a transposição demonstrada no aplicativo deriva de um fluxo dinâmico de tratamento de mensagens - admitindo, portanto, infinitas possibilidades similares à amostra. Todavia, a plataforma citada

já foi utilizada com o propósito de comparação, além de ser a fonte das imagens apresentadas, o que poderia transparecer um certo viés. Para ampliar a confiabilidade no processo, decidiu-se encontrar informações de outra origem para continuar a avaliação. Por isso, aplicou-se a sentença *"frases em libras com legenda"* em uma pesquisa no YouTube (https://www.youtube.com/), objetivando encontrar um vídeo aleatório que fosse condizente com o escopo buscado.

Figura 58 – Exemplo 2: *ELA TEM CIÚME DO NAMORADO.*

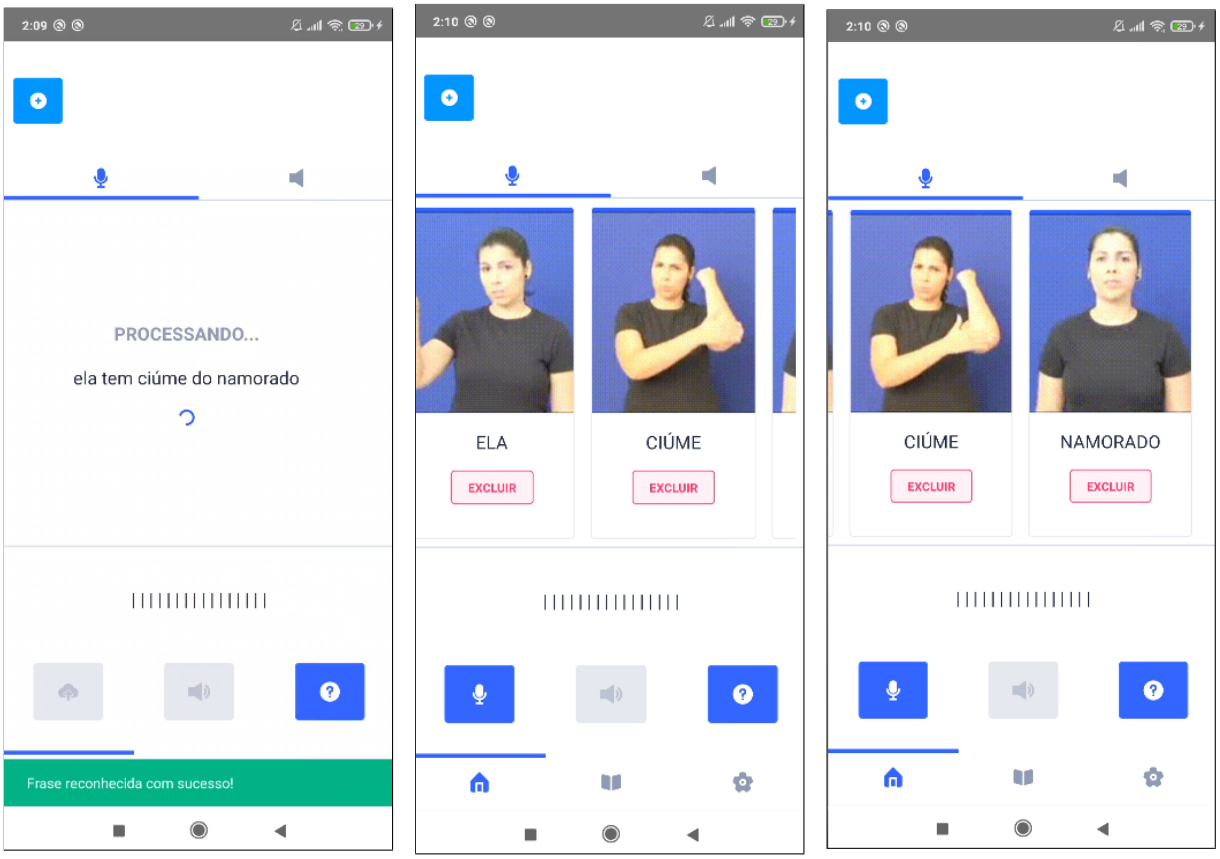


Fonte: Autor.

A Figura 58 - borrada propositalmente pois, apesar do conteúdo ser público, não houve um contato formal com a autora para utilização deste exemplo - apresenta a resposta considerada mais adequada à procura. Nela, é possível contrastar o formato de outra frase afirmativa - porém mais sinuosa - em Libras e em Português. Recordando que a intenção deste projeto, citada no Capítulo 1, é desenvolver uma POC que ateste a viabilidade da criação de um aplicativo que facilite a comunicação da comunidade surda por meio da união de técnicas e recursos, realizar comparações com conteúdo já disponíveis na internet, por exemplo, aparenta ser fundamental.

Em virtude disso, o intuito deste paralelo é constatar que a aplicação móvel, de fato, produz resoluções semelhantes às de outras plataformas ou profissionais. Para tal, da mesma forma que no primeiro exemplo, proferiu-se a oração encontrada enquanto a funcionalidade de recognição estava em uso. O retorno obtido está exposto na Figura 59.

Figura 59 – Exemplo 2: Abordagem **DeafEnd**



(a) Processando frase (b) Primeira parte da exibição (c) Segunda parte da exibição

Fonte: Autor.

5.2 Estímulo ao domínio

O segundo item contributivo refere-se ao ponto de vista técnico, visto que o Autor não possuía experiência com os campos de inteligência artificial e processamento de linguagem natural. Entendendo esta carência, seguramente seria necessário adquirir conhecimento acerca dos temas citados. Felizmente, atualmente existem diversos materiais - livros, artigos e palestras - que, impulsionados pela abrangência da internet, estão acessíveis para qualquer indivíduo que possua uma conexão com a rede. No entanto, ainda são esporádicos os tutoriais (principalmente em Português) que, enquanto demonstram quais comandos devem ser executados e os resultados esperados, empenham-se em esclarecer os conceitos de determinada técnica.

Pensando em entusiastas do ramo ou pessoas interessadas em, rapidamente, experimentar os resultados de uma simples interação com algum procedimento relacionado ao tema, este trabalho elaborou um tutorial[1] introdutório ao PLN com a biblioteca `spaCy` (usada neste projeto), exibido parcialmente no Apêndice C. Este estilo de anotação serve como guia para pessoas que, talvez, encontrem-se na mesma situação do Autor relatada no início deste Subcapí-

[1] `Tutorial:` https://github.com/joao-vieira/deafend-nlp-intro

tulo, além de ser um acréscimo interessante para trechos deste documento - tal como a Subseção 4.3.1.2. As instruções referidas foram escritas na linguagem de marcação *Markdown* e contêm, além das demonstrações das alternativas para casos de uso específicos, uma explanação sobre como configurar um ambiente Linux para executar o passo a passo. Ao final, inclui-se uma lista com as referências que embasaram tal conteúdo - como forma de encorajar o leitor a prosseguir seus estudos - e os *links* dos repositórios que integram este trabalho.

5.3 Aplicativo como produto

Naturalmente, por tratar-se de uma atividade prática que, em paralelo à pesquisa, visa criar um produto, o próprio aplicativo pode ser compreendido como a terceira contribuição gerada pelo projeto. Tal afirmação fundamenta-se no fato de que a aplicação móvel elaborada contempla grande parte das funcionalidades disponíveis em outros *softwares* que atuam no campo da surdez, de acordo com a comparação detalhada no Capítulo 3. Isto assegura que a metodologia de desenvolvimento utilizada, que iniciou-se com um extenso levantamento bibliográfico (Capítulo 2), seguido pela análise de requisitos (Subcapítulo 4.1) e criação dos protótipos (Subcapítulo 4.2), culminando na implementação do aplicativo (Subcapítulo 4.3), é efetiva e conduziu, com sucesso, esta ideia rumo às metas traçadas.

Contudo, um sistema não consiste apenas no código que origina-o. Apesar de orientar as definições de *layout* do aplicativo na dificuldade dos surdos de assimilar a língua portuguesa e, na medida do possível, colocar apenas ícones ou imagens nas telas, além de buscar criar uma interface extremamente amigável ao utilizador, de nada adianta um sistema interessante que não seja intuitivo ou cause dúvidas na pessoa que irá usá-lo. É por este motivo que, desde sua concepção, este trabalho mostrou a intenção de conter uma documentação completa e interativa, que pudesse servir de apoio ao usuário - especialmente nos seus primeiros contatos com a aplicação móvel. Portanto, ainda que tenha sido citado diversas vezes ao longo deste documento, considerou-se importante reforçar, neste Subcapítulo, que o manual de usuário hospedado no endereço http://deafend-documentation.vercel.app/ e exibido no Apêndice D, também é um elemento fundamental na composição do *produto* denominado **DeafEnd**.

5.4 Trabalhos futuros

Visando melhorar a versão construída, é possível destacar certas adições que tornarão o aplicativo ainda mais abrangente e, consequentemente, melhor. Inicialmente, seria bastante interessante analisar alternativas que permitam uma tratativa especial para frases interrogativas, uma vez que, até o presente momento, tais sentenças não possuem uma abordagem específica. Esta manipulação faz-se relevante em virtude das possíveis transformações que existem na estrutura gramática deste tipo de oração da língua portuguesa. Para viabilizar esta atualização, será necessário duas atuações: a primeira diz respeito ao reconhecimento de fala, já que a biblioteca `react-native-tts` (detalhada na Seção 2.5.2) - responsável por este papel - não consegue re-

conhecer uma pergunta apenas baseada no tom de voz do usuário; a segunda refere-se à emissão de resposta, pois a *lib* `voice` (descrita na Seção 2.5.6), encarregada desta funcionalidade, não altera a entonação do conteúdo enunciado, independente da pontuação.

Outro incremento viável e altamente útil é a criação de um histórico de traduções - funcionalidade que já existe no aplicativo *Hand Talk* (Subcapítulo 3.1), mas que está disponível apenas para usuários registrados. A disponibilização desta função, além de tornar a aplicação móvel mais prática, poderá ser a base para a elaboração de uma série de métricas pertinentes, tais como: qual palavra é mais utilizada; qual é o período do dia em que o usuário mais acessa o aplicativo; qual estrutura frasal é mais montada pelo usuário; entre outras. Todas estas informações, caso armazenadas em uma estrutura pesquisável ou após passarem por um processo de limpeza, poderiam propiciar o desenvolvimento de outras *features* ou serem usadas como inspiração para prováveis melhorias.

Além destes, outro ponto passível de expansão corresponde às experiências de listagem da aplicação móvel. Atualmente, as telas que exibem a lista de termos cadastrados contam com o filtro horizontal por letras, que permite ao usuário visualizar apenas as palavras que iniciarem com a letra selecionada. Contudo, para esta prova de conceito - conforme explicado na Subseção 4.3.1.1 - não foi preciso importar todos os sinais catalogados no Dicionário da Língua Brasileira de Sinais V3, o que significa que a quantidade de itens pode aumentar consideravelmente. Sendo assim, seria proveitoso acrescentar outro tipo de filtro e, especialmente, formas de ordenação destas palavras, como por exemplo: ordem alfabética crescente (A-Z); ordem alfabética decrescente (Z-A); palavras mais usadas e menos usadas; entre outras.

Por fim, seria interessante explorar as opções disponíveis para serem empregadas no tratamento da sentença que é emitida como resposta. Embora seja legítimo dizer que o aplicativo cumpriu seu propósito inicial, no momento ele apenas exterioriza, através dos alto-falantes do aparelho, as palavras que o próprio usuário escolheu. Logo, visando a evolução da funcionalidade de emissão, recursos como redes neurais ou *ElasticSearch* poderiam potencializar a qualidade destas frases viabilizando, por exemplo, a inserção de conectivos coesos entre os termos selecionados. Esta melhoria, desde que seja dinâmica, agregará naturalidade à oração, que ficará mais compatível com as regras gramaticais do Português e, consequentemente, será compreendida com mais facilidade pelos ouvintes.

6 CONSIDERAÇÕES FINAIS

Observando o projeto elaborado, nota-se a importância individual de cada uma das etapas que compõe a metodologia de desenvolvimento adotada. Apesar de todas as novas ferramentas disponíveis nos dias de hoje, fica claro que o planejamento ainda mostra-se como elemento decisivo para o sucesso de qualquer atividade. Desta forma, vale ressaltar a contribuição que os períodos de levantamento de requisitos e prototipação apresentaram no que tange à redução de desvios que costumam aparecer ao longo da codificação de um aplicativo. Cada diagrama concebido que, após passar pelas fases de planejamento e algumas validações pontuais, transformou-se em um esboço de tela, possibilitou o mapeamento de todos os dados e recursos que seriam indispensáveis à aplicação móvel - antes mesmo da primeira linha de código ser escrita. Além disso, considerando a inexperiência do Autor em várias tecnologias que tornaram-se cruciais para o resultado obtido, a oportunidade de antecipar possíveis problemas foi, indiscutivelmente, uma das decisões mais corretas deste trabalho.

Com uma base de conhecimento sólida e objetivos bem definidos, existe a confiança necessária para iniciar a etapa de implementação. Sem dúvidas, a experiência prévia como desenvolvedor e o contato com a comunidade tecnológica auxiliou a tomada de certas decisões, tal como optar pela técnica de desenvolvimento híbrido - `React Native` - por conhecer o *framework* `React` e ter ciência da quantidade de desenvolvedores ativos nos fóruns desta tecnologia. Porém, o momento de firmamento das definições arquiteturais ocorreu, de fato, na etapa de pesquisa sobre bibliotecas existentes e formas viáveis de realizar determinada requisição, abordada principalmente no Capítulo 2. Após entender qual seria a responsabilidade de cada um destes agentes, foi possível orquestrar suas atuações em conjunto.

A visão integral da comunicação entre cada parte da solução possibilitou, por exemplo, identificar a necessidade de criar uma API REST para centralizar o processamento das informações e não sobrecarregar o aplicativo. Novamente, a investigação de características conduziu à escolha do `FastAPI` - baseado na linguagem de programação Python - para este papel. Neste mesmo formato, optou-se por utilizar: o *cloud* da AWS para armazenar a maior quantidade de informações; a biblioteca `UI Kitten` para fornecer o padrão visual e componentes pré-prontos que agilizassem o desenvolvimento; a *lib* `Voice` para emitir frases do dispositivo; o pacote `react-native-tts` para converter fala em texto; entre outros.

Conforme cada uma destas ferramentas era adotada e um novo conhecimento era fixado, mais perto chegava-se do propósito desta atividade. É por isso que, remetendo-se ao Resumo e ao Capítulo 1, é factível afirmar que o aplicativo **DeafEnd** provou ser um conceito válido. O mais interessante é perceber que criar uma aplicação móvel com bom potencial de crescimento dependeu, somente, de dedicação e, mais do que o fruto instalável através de uma loja de *apps*, fica a satisfação com os conhecimentos adquiridos e a ponte que possa ter sido criada entre surdos e ouvintes.

REFERÊNCIAS

AKVEO. **UI Kitten - React Native UI Library based on Eva Design System**. 2019. Disponível em: <https://akveo.github.io/react-native-ui-kitten/docs/design-system/eva-design-system-intro#eva-design-system>. Acesso em: 01 de novembro de 2020. Citado 2 vezes nas páginas 27 e 51.

ALMEIDA, W. G. **Introdução à língua brasileira de sinais**. 1. ed. Bahia: UAB/UESC, 2013. Citado 4 vezes nas páginas 9, 12, 13 e 14.

ALVES, T. S.; TONIOLO, C. M. **Tecnologias para web e para dispositivos móveis**. 1. ed. Londrina: Editora e Distribuidora Educacional S.A., 2016. Citado na página 14.

ANDRADE, C. A. A. de et al. Avaliação de um módulo de reconhecimento de fala na plataforma arduino. n. 29, 2016. Citado na página 26.

ANTUNES, M. **API Restful: conceito, princípios e como criar**. 2019. Disponível em: <https://www.hostgator.com.br/blog/api-restful/>. Acesso em: 24 de outubro de 2020. Citado na página 4.

ARAGON, C. A.; SANTOS, I. B. Deficiência auditiva/surdez: conceitos, legislações e escolarizações. v. 5, n. 2, p. 119–140, 2015. Citado na página 2.

ARAÚJO, E. C. de. **Xamarin Forms e MVVM: persistência local com Entity Framework Core**. [S.l.]: Casa do Código, 2019. Citado na página 17.

AWS. **O que é NoSQL?** 2019. Disponível em: <https://aws.amazon.com/pt/nosql/>. Acesso em: 18 de outubro de 2020. Citado na página 4.

AWS. **Amazon DynamoDB**. 2020. Disponível em: <https://aws.amazon.com/pt/dynamodb>. Acesso em: 17 de outubro de 2020. Citado na página 30.

AWS. **Amazon S3**. 2020. Disponível em: <https://aws.amazon.com/pt/s3/>. Acesso em: 17 de outubro de 2020. Citado na página 30.

AWS. **AWS SDK para JavaScript no Node.js**. 2020. Disponível em: <https://aws.amazon.com/pt/sdk-for-node-js/>. Acesso em: 17 de outubro de 2020. Citado na página 29.

AWS. **What Is the AWS SDK for JavaScript?** 2020. Disponível em: <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/welcome.html>. Acesso em: 17 de outubro de 2020. Citado na página 29.

BARBOZA, F. F. M.; FREITAS, P. H. C. **Modelagem e desenvolvimento de banco de dados**. Porto Alegre: SAGAH, 2018. Citado na página 49.

BERNARDES, T. F.; MIYAKE, M. Cross-platform mobile development approaches: A systematic review. **IEEE Latin America Transactions**, v. 14, p. 1892–1898, 04 2016. Citado na página 18.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. 3. ed. Rio de Janeiro: Elsevier, 2015. Citado 4 vezes nas páginas 39, 40, 46 e 49.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with Python**. 1. ed. Califórnia: O'Reilly, 2009. Citado na página 64.

BOTELHO, B. P. Acessibilidade para surdo: novas tecnologias para a comunicação em libras. 2015. Citado na página 8.

BOTO3. **Documentação oficial**. 2020. Disponível em: <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>. Acesso em: 24 de outubro de 2020. Citado na página 30.

BRASIL. **LEI Nº 10.436**. 2002. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/2002/l10436.htm>. Acesso em: 08 de outubro de 2020. Citado na página 2.

BRASIL, H. **O que são GIFs e como usá-los na estratégia de comunicação da sua empresa**. 2018. Disponível em: <https://www.hostgator.com.br/blog/o-que-sao-gifs-e-como-usar/>. Acesso em: 01 de maio de 2020. Citado na página 16.

BRASILEIRO, S. P. **VLibras**. 2019. Disponível em: <https://softwarepublico.gov.br/gitlab/groups/vlibras>. Acesso em: 05 de novembro de 2020. Citado na página 38.

CAPELAS, B. **Veja como os GIFs evoluíram ao longo do tempo**. 2017. Disponível em: <https://link.estadao.com.br/noticias/cultura-digital,veja-como-os-gifs-evoluiram-ao-longo-do-tempo,70002015626>. Acesso em: 01 de maio de 2020. Citado na página 16.

CGI.BR. **Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros: TIC domicílios 2018**. 14. ed. São Paulo: Núcleo de Informação e Coordenação do Ponto BR, 2019. Citado na página 1.

CINEMA, M. **A História da gravação de áudio**. 2019. Disponível em: <https://medium.com/@montagem/a-hist%C3%B3ria-da-gravac%CC%A7a%CC%83o-de-a%CC%81udio-d8d2b0fd6c71>. Acesso em: 19 de abril de 2020. Citado na página 15.

COELHO, B. **Inclusão é direito: as principais leis de acessibilidade no Brasil**. 2018. Disponível em: <http://blog.handtalk.me/leis-de-acessibilidade/>. Acesso em: 11 de abril de 2020. Citado na página 8.

COELHO, R. **Natural Language Processing. Um pouco de história.** 2019. Disponível em: <https://developer.ibm.com/br/technologies/natural-language-processing/articles/natural-language-processing-um-pouco-de-historia/>. Acesso em: 11 de junho de 2020. Citado na página 24.

COPPIN, B. **Inteligência Artificial**. 1. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda, 2013. Citado 3 vezes nas páginas 22, 23 e 25.

COSTA, M. B. **O que é GIF e como usá-lo**. 2020. Disponível em: <https://canaltech.com.br/software/o-que-e-gif-e-como-usa-lo/>. Acesso em: 01 de maio de 2020. Citado na página 16.

CRISTIANO, A. **Bill Clinton**. 2020. Disponível em: <https://www.libras.com.br/surdos-famosos-bill-clinton>. Acesso em: 10 de abril de 2020. Citado na página 2.

DILLI, K. S. A inclusão do surdo na educação brasileira. Trabalho de conclusão de curso apresentado como requisito parcial para obtenção do título de bacharel em Serviço Social pela UFSC. 2010. Citado na página 6.

DOCUSAURUS. **Documentação oficial**. 2020. Disponível em: <https://v2.docusaurus.io/docs/>. Acesso em: 27 de outubro de 2020. Citado na página 32.

DOLPHIN, C. **Skeleton screens with React and React Native**. 2018. Disponível em: <https://www.digitalocean.com/community/tutorials/react-skeleton-screens-react-and-react-native>. Acesso em: 01 de novembro de 2020. Citado na página 74.

DONATO, A. D.; DINIZ, S. Libras i. Material da disciplina de Libras I da UFPB, focado em nos tópicos: o Cérebro e a Língua de Sinais; Processos Cognitivos e Linguísticos; Linguística Aplicada: Fonologia, Morfologia e Sintaxe. 2010. Citado 2 vezes nas páginas 9 e 10.

DUARTE, S. B. R. et al. Aspectos históricos e socioculturais da população surda. v. 20, n. 4, p. 1713–1734, 2013. Citado 2 vezes nas páginas 2 e 6.

DURÃES, G. **Híbrido ou nativo: qual desenvolvimento escolher?** 2019. Disponível em: <https://blog.tecnospeed.com.br/aplicativo-hibrido-ou-nativo/>. Acesso em: 04 de novembro de 2020. Citado na página 18.

DURÃES, G. **Desenvolvimento mobile: desafios, tendências e dicas**. 2020. Disponível em: <https://blog.tecnospeed.com.br/desafios-no-desenvolvimento-mobile/>. Acesso em: 04 de novembro de 2020. Citado na página 17.

EISENMAN, B. **Learning React Native: Building native mobile apps with JavaScript**. 2. ed. Califórnia: O'Reilly, 2018. Citado na página 21.

FACELI, K. et al. **Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina**. 1. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda, 2011. Citado na página 23.

FASTAPI. **Documentação oficial**. 2020. Disponível em: <https://fastapi.tiangolo.com/>. Acesso em: 01 de novembro de 2020. Citado na página 31.

FFMPEG. **Documentação oficial**. 2019. Disponível em: <https://ffmpeg.org/>. Acesso em: 13 de outubro de 2020. Citado na página 31.

FFMPY. **Documentação oficial**. 2020. Disponível em: <https://pypi.org/project/ffmpy/#history>. Acesso em: 13 de outubro de 2020. Citado na página 31.

FIGMA. **Free Prototyping Tool to Create Clickable Prototypes**. 2020. Disponível em: <https://www.figma.com/prototyping-tool/>. Acesso em: 09 de setembro de 2020. Citado na página 51.

FLUTTER. **Documentação oficial**. 2020. Disponível em: <https://flutter.dev/>. Acesso em: 21 de outubro de 2020. Citado na página 20.

FOWLER, M. **UML Essencial: um breve guia para a linguagem-padrão de modelagem de objetos**. 3. ed. São Paulo: Bookman, 2005. Citado na página 40.

FRAMEWORK, I. **Documentação oficial**. 2020. Disponível em: <https://ionicframework.com/docs>. Acesso em: 18 de outubro de 2020. Citado na página 19.

GANDRA, A. **País tem 10,7 milhões de pessoas com deficiência auditiva, diz estudo**. 2019. Disponível em: <https://agenciabrasil.ebc.com.br/geral/noticia/2019-10/brasil-tem-107-milhoes-de-deficientes-auditivos-diz-estudo>. Acesso em: 19 de abril de 2020. Citado na página 2.

GLOBO, O. **Celebridades que têm deficiência auditiva contam como superaram os obstáculos**. 2017. Disponível em: <https://oglobo.globo.com/sociedade/educacao/celebridades-que-tem-deficiencia-auditiva-contam-como-superaram-os-obstaculos-22031605>. Acesso em: 10 de abril de 2020. Citado na página 2.

GONÇALVES, A. **O que é URL, como localizá-la e qual a sua importância**. 2020. Disponível em: <https://www.hostinger.com.br/tutoriais/url/#O-que-e-URL>. Acesso em: 20 de setembro de 2020. Citado na página 29.

GUEDES, M. **App nativo x app híbrido: existe o melhor?** 2017. Disponível em: <https://www.treinaweb.com.br/blog/app-nativo-x-app-hibrido-existe-o-melhor/>. Acesso em: 04 de novembro de 2020. Citado 2 vezes nas páginas 17 e 18.

HAMANN, R. **GIFs animados: 25 anos de história**. 2012. Disponível em: <https://www.tecmundo.com.br/internet/25164-gifs-animados-25-anos-de-historia.htm>. Acesso em: 01 de maio de 2020. Citado na página 16.

HARDENIYA, N. **NLTK Essentials**. 1. ed. Birmingham: Packt, 2015. Citado 2 vezes nas páginas 63 e 64.

HONORA, M.; FRIZANCO, M. L. E. **Livro ilustrado de Língua Brasileira de Sinais: desvendando a comunicação usada pelas pessoas com surdez**. São Paulo: Ciranda Cultural, 2009. Citado na página 12.

IBGEEDUCA. **Uso de internet, televisão e celular no Brasil**. 2017. Disponível em: <https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html>. Acesso em: 16 de maio de 2020. Citado na página 17.

ICTS, G. **Rybená - Tradutor de Português - Libras**. 2020. Disponível em: <https://portal.rybena.com.br/site-rybena/index.html>. Acesso em: 29 de outubro de 2020. Citado na página 34.

ICTS, G. **Rybená Tradutor Libras Voz**. 2020. Disponível em: <https://play.google.com/store/apps/details?id=br.com.icts.rybenatorandroid&hl=pt_BR&gl=US>. Acesso em: 28 de outubro de 2020. Citado na página 35.

IDASZAK, D. **Comparison of Text to Speech Solutions for React Native**. 2020. Disponível em: <https://www.netguru.com/codestories/react-native-text-to-speech>. Acesso em: 03 de novembro de 2020. Citado na página 28.

IMAGE react-native-fast. **Documentação oficial**. 2020. Disponível em: <https://www.npmjs.com/package/react-native-fast-image>. Acesso em: 01 de novembro de 2020. Citado na página 28.

INEP. **Enem 2017 tem 7,6 milhões de inscritos**. 2017. Disponível em: <http://portal.inep.gov.br/artigo/-/asset_publisher/B4AQV9zFY7Bv/content/enem-2017-tem-7-6-milhoes-de-inscritos/21206#:~:text=O%20Exame%20Nacional%20do%20Ensino,5%20e%2012%20de%20novembro.> Acesso em: 09 de outubro de 2020. Citado na página 3.

INES. **Acessibilidade**. 2020. Disponível em: <http://www.ines.gov.br/acessibidade-menu>. Acesso em: 10 de abril de 2020. Citado na página 8.

JABBAR, G. **Speech to Text Conversion in React Native - Using Hooks (Use useState and useEffect)**. 2020. Disponível em: <https://medium.com/@gilshaan/speech-to-text-conversion-in-react-native-using-hooks-use-usestate-and-useeffect-c2b9de6b8a8a>. Acesso em: 05 de novembro de 2020. Citado na página 29.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. The unified software development process. In: . [S.l.: s.n.], 1999. Citado na página 39.

JUNIOR, A. T. da C. Processamento de linguagem natural para indexação automática semântico-ontológica. Tese apresentada à banca examinadora como requisito parcial à obtenção do Título de Doutor em Ciência da Informação pela Universidade de Brasília. 2013. Citado na página 24.

JUSTIÇA, O. S. N. de. **A classificação indicativa na Língua Brasileira de Sinais**. 1. ed. Brasília: SNJ, 2009. Citado na página 5.

KOROLOV, M. **Estas são as principais aplicações de inteligência artificial hoje**. 2019. Disponível em: <https://cio.com.br/estas-sao-as-principais-aplicacoes-de-inteligencia-artificial-hoje/>. Acesso em: 02 de maio de 2020. Citado na página 23.

LARMAN, C. **Utilizando UML e Padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo**. 3. ed. Porto Alegre: Bookman, 2007. Citado na página 39.

LEE, V.; SCHNEIDER, H.; SCHELL, R. **Aplicações móveis: arquitetura, projeto e desenvolvimento**. 1. ed. São Paulo: Pearson Education do Brasil, 2005. Citado na página 14.

LEVITIN, D. J. **A mente organizada - Como pensar com clareza na era da sobrecarga de informação**. 1. ed. Rio de Janeiro: Objetiva Ltda, 2015. Citado na página 1.

LIMA, V. **Desenvolvimento para mobile: Ionic, React-Native ou Flutter, qual usar?** 2020. Disponível em: <https://blog.geekhunter.com.br/desenvolvimento-para-mobile-reactnative-flutter-ionic/>. Acesso em: 30 de outubro de 2020. Citado 3 vezes nas páginas 19, 20 e 21.

LINS, N. **O que é o Benchmark e Benchmarking?** 2019. Disponível em: <https://www.psiuproducoes.com.br/o-que-e-o-benchmark-e-benchmarking/>. Acesso em: 28 de outubro de 2020. Citado na página 33.

LIRA, G. de A.; SOUZA, T. A. F. de. **Dicionário da Língua Brasileira de Sinais V3**. 2011. Disponível em: <http://www.acessibilidadebrasil.org.br/libras_3/>. Acesso em: 15 de julho de 2020. Citado na página 57.

LOCAWEB. **Framework: usar ou não usar? Eis a questão!** 2016. Disponível em: <https://blog.locaweb.com.br/artigos/desenvolvimento-artigos/framework-usar-ou-nao-usar-eis-questao/>. Acesso em: 28 de outubro de 2020. Citado na página 27.

LOPES, S. **Aplicações mobile híbridas com Cordova e PhoneGap**. 1. ed. São Paulo: Casa do Código, 2016. Citado na página 18.

LOPEZ, B. **Estatísticas de Celulares no Brasil**. 2019. Disponível em: <https://www.pagbrasil.com/pt-br/insights/relatorio-digital-in-2019-brasil/>. Acesso em: 19 de abril de 2020. Citado na página 14.

LUGER, G. F. **Inteligência Artificial**. 6. ed. São Paulo: Pearson Education do Brasil, 2013. Citado na página 22.

MACHADO, F. N. R. **Banco de dados: projeto e implementação**. 4. ed. São Paulo: Érica, 2020. Citado na página 49.

MAINKAR, P.; GIORDANO, S. **Google Flutter mobile development quick start guide**. 1. ed. Mumbai: Packt, 2019. Citado 3 vezes nas páginas 19, 20 e 21.

MALINOSQUI, G. **Aplicativo híbrido: O que é e porque você deveria conhecer**. 2019. Disponível em: <https://ezdevs.com.br/aplicativo-hibrido-porque-voce-deveria-conhecer/>. Acesso em: 24 de outubro de 2020. Citado na página 4.

MANFIO, E. R. Relação entre reconhecimento e compreensão de voz: experimento para análise linguística. v. 17, n. 1, p. 281–300, 2017. Citado na página 26.

MARKETEAM. **Aplicativo nativo, web app ou aplicativo híbrido?** 2020. Disponível em: <https://usemobile.com.br/aplicativo-nativo-web-hibrido/>. Acesso em: 27 de outubro de 2020. Citado na página 18.

MARTIN, S. **React Native vs. Flutter vs. Ionic**. 2020. Disponível em: <https://medium.com/better-programming/react-native-vs-flutter-vs-ionic-46d3350f96ee>. Acesso em: 04 de novembro de 2020. Citado 3 vezes nas páginas 19, 20 e 21.

MEDEIROS, L. F. de. **Inteligência artificial aplicada: uma abordagem introdutória**. 1. ed. Curitiba: InterSaberes, 2018. Citado na página 22.

MORAIS, C. E. L. de et al. **Libras**. 2. ed. Porto Alegre: SAGAH EDUCAÇÃO S.A., 2018. Citado 2 vezes nas páginas 6 e 7.

MUNDO-E. **A Importância da Comunicação em Nossas Vidas**. 2018. Disponível em: <https://mundoe.net.br/comunicacao/>. Acesso em: 07 de outubro de 2020. Citado na página 1.

NAVIGATION, R. **Documentação oficial**. 2020. Disponível em: <https://reactnavigation.org/docs/hello-react-navigation>. Acesso em: 01 de novembro de 2020. Citado na página 29.

NETO, J. M. de O.; TONIN, S. D.; PRIETCH, S. S. Processamento de linguagem natural e suas aplicações computacionais. 2010. Citado na página 25.

NETO, N.; SILVA Ênio; SOUSA, E. Software usando reconhecimento e síntese de voz: O estado da arte para o português brasileiro. p. 326–331, 2005. Citado na página 15.

NEVES, F. **As palavras mais usadas na língua portuguesa**. 2019. Disponível em: <https://www.dicio.com.br/as-palavras-mais-usadas-na-lingua-portuguesa/>. Acesso em: 07 de outubro de 2020. Citado na página 58.

NUNES, S. da S. et al. Surdez e educação: escolas inclusivas e/ou bilíngues? **Psicologia Escolar e Educacional**, scielo, v. 19, p. 537 – 545, 12 2015. ISSN 1413-8557. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-85572015000300537&nrm=iso>. Citado na página 5.

OLIVEIRA, D. H. D. de et al. Prototipação de interfaces de aplicativos para dispositivos móveis: Estado da arte e desafios de ihc. v. 14, p. 315–324, 2012. Citado na página 15.

OVERBYE, D. **Mystery of Big Data's Parallel Universe Brings Fear, and a Thrill**. 2012. Disponível em: <https://www.nytimes.com/2012/06/05/science/big-datas-parallel-universe-brings-fears-and-a-thrill.html>. Acesso em: 08 de outubro de 2020. Citado na página 1.

PEREIRA, A. P. **Como funciona o reconhecimento de voz?** 2009. Disponível em: <https://www.tecmundo.com.br/curiosidade/3144-como-funciona-o-reconhecimento-de-voz-.htm>. Acesso em: 22 de abril de 2020. Citado na página 26.

PEREIRA, L. A. de M. **Análise e Modelagem de Sistemas com a UML: com dicas e exercícios resolvidos**. 1. ed. Rio de Janeiro: Luiz Antônio de Moraes Pereira, 2011. Citado na página 42.

PEREIRA, S. do L. Processamento de linguagem natural. v. 31, 2011. Citado na página 24.

PERKINS, J. **Python 3 text processing with NLTK 3 cookbook**. 2. ed. Birmingham: Packt, 2014. Citado na página 62.

PINHEIRO, D. C. de S. O papel do plano de comunicação preventivo em momento de crise na organização. Monografia apresentada como requisito parcial para a graduação no curso de Comunicação Social, habilitação em Jornalismo pela UFG. 2005. Citado na página 1.

PRESSMAN, R. S.; MAXIM, B. R. **Engenheria de software: uma abordagem profissional**. 8. ed. Porto Alegre: AMGH, 2016. Citado 2 vezes nas páginas 39 e 41.

QUADROS, R. M. de; KARNOPP, L. B. **Língua de sinais brasileira: estudos linguísticos**. Porto Alegre: Artmed, 2004. Citado 2 vezes nas páginas 10 e 12.

QUADROS, R. M. de; PIZZIO, A. L.; REZENDE, P. L. F. Língua brasileira de sinais i. Material da disciplina de Língua Brasileira de Sinais I da UFSC. 2009. Citado 2 vezes nas páginas 9 e 10.

RODRIGUES, J. **O que é o Processamento de Linguagem Natural?** 2017. Disponível em: <https://medium.com/botsbrasil/o-que-%C3%A9-o-processamento-de-linguagem-natural-49ece9371cff>. Acesso em: 21 de abril de 2020. Citado na página 24.

SACKS, O. **Vendo vozes: uma viagem ao mundo dos surdos**. São Paulo: Companhia das Letras, 2010. Citado na página 6.

SANTOS, R. E. S. et al. Técnicas de processamento de linguagem natural aplicadas ao processo de mineração de textos: Resultados preliminares de um mapeamento sistemático. v. 4, n. 2, p. 116–125, 2014. Citado na página 24.

SAS. **Processamento de Linguagem Natural: O que é e qual sua importância?** 2019. Disponível em: <https://www.sas.com/pt_br/insights/analytics/processamento-de-linguagem-natural.html>. Acesso em: 24 de outubro de 2020. Citado na página 4.

SCHOPENHAUER, A. **O mundo como vontade e representação**. Rio de Janeiro: Contraponto, 2001. Citado na página 3.

SILVA, C. F. da; FERREIRA, S. B. L.; SACRAMENTO, C. Mobile application accessibility in the context of visually impaired users. n. 32, p. 1–10, 2018. Citado na página 15.

SILVA, F. M. da et al. **Inteligência Artificial**. 1. ed. Porto Alegre: SAGAH EDUCAÇÃO S.A., 2019. Citado na página 23.

SILVA, M. J. V. e et al. **Design thinking: inovação em negócios**. 1. ed. Rio de Janeiro: MJV Press, 2012. Citado na página 51.

SNACKBAR react-native. **Documentação oficial**. 2020. Disponível em: <https://www.npmjs.com/package/react-native-snackbar>. Acesso em: 01 de novembro de 2020. Citado na página 28.

SONZA, A. P. et al. **Acessibilidade e tecnologia assistiva: pensando a inclusão sociodigital de pessoas com necessidades especiais**. 20. ed. Bento Gonçalves: Ministério da Educação, 2013. Citado 2 vezes nas páginas 6 e 33.

SORAL, R. **React Native vs Ionic: Which Framework is best and why?** 2020. Disponível em: <https://www.simform.com/react-native-vs-ionic>. Acesso em: 06 de novembro de 2020. Citado na página 21.

SOUZA, D. A. de et al. Estratégias inteligentes para desenvolvimento de aplicativos mobile multiplataforma. 2016. Disponível em: <https://www.aedb.br/seget/arquivos/artigos17/12425177.pdf>. Citado na página 17.

SOUZA, I. de. **Framework: descubra o que é, para que serve e por que você precisa de um para o seu site**. 2019. Disponível em: <https://rockcontent.com/br/blog/framework/>. Acesso em: 20 de outubro de 2020. Citado na página 4.

SOUZA, I. de. **Saiba o que é UI (User Interface) e a importância dele para os clientes**. 2020. Disponível em: <https://rockcontent.com/br/blog/o-que-e-ui/>. Acesso em: 02 de novembro de 2020. Citado na página 51.

SPACY. **spaCy 101: Everything you need to know**. 2020. Disponível em: <https://spacy.io/usage/spacy-101>. Acesso em: 24 de outubro de 2020. Citado na página 31.

STACKOVERFLOW. **Stack Overflow Annual Developer Survey - 2020**. 2020. Disponível em: <https://insights.stackoverflow.com/survey/2020>. Acesso em: 06 de novembro de 2020. Citado 2 vezes nas páginas 21 e 22.

STENO, G. **"Setembro Azul": vamos falar de acessibilidade para surdos?** 2018. Disponível em: <http://steno.com.br/setembro-azul-vamos-falar-de-acessibilidade-para-surdos/>. Acesso em: 02 de maio de 2020. Citado na página 8.

STORAGE, R. N. C. A. **Documentação oficial**. 2020. Disponível em: <https://react-native-async-storage.github.io/async-storage/>. Acesso em: 30 de outubro de 2020. Citado na página 28.

STROBEL, K. L. Surdos: vestígios culturais não registrados na história. Tese apresentada ao programa de pós-graduação doutorado em educação da UFSC como requisito para obtenção do título de doutorado. 2008. Citado na página 2.

STROBEL, K. L.; FERNANDES, S. **Aspectos linguísticos da língua brasileira de sinais**. Curitiba: Secretaria de Estado da Educação - Superintendência de Educação - Departamento de Educação Especial, 1998. Citado 5 vezes nas páginas 9, 10, 11, 12 e 13.

TALK, H. **Hand Talk Tradutor para Libras**. 2020. Disponível em: <https://play.google.com/store/apps/details?id=br.com.handtalk&hl=pt_BR&gl=US>. Acesso em: 28 de outubro de 2020. Citado 2 vezes nas páginas 33 e 34.

TEVAH, R. T. Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro. Dissertação submetida como parte dos requisitos necessários para a obtenção do grau de mestre em Ciências em Engenharia Elétrica pela UFRJ. 2006. Citado na página 26.

TTS react-native. **Documentação oficial**. 2020. Disponível em: <https://www.npmjs.com/package/react-native-tts>. Acesso em: 01 de novembro de 2020. Citado na página 27.

VEJA. **5,1 bilhão de pessoas têm celular no planeta, sendo 204 milhões no Brasil**. 2019. Disponível em: <https://veja.abril.com.br/economia/51-bilhao-de-pessoas-tem-celular-no-planeta-sendo-204-milhoes-no-brasil/>. Acesso em: 19 de abril de 2020. Citado na página 15.

VIEIRA, M. S. "ok google, já chega": privacidade e reconhecimento de fala ininterrupto em celulares. v. 12, n. 2, p. 299–307, 2016. Citado 2 vezes nas páginas 25 e 26.

VLIBRAS. **VLibras - Tradução de Português para Libras**. 2016. Disponível em: <https://www.vlibras.gov.br/>. Acesso em: 02 de maio de 2020. Citado na página 35.

VLIBRAS. **VLibras**. 2020. Disponível em: <https://play.google.com/store/apps/details?id=com.lavid.vlibrasdroid>. Acesso em: 28 de outubro de 2020. Citado na página 36.

VOICE, R. N. C. **Documentação oficial**. 2020. Disponível em: <https://www.npmjs.com/package/@react-native-community/voice>. Acesso em: 26 de outubro de 2020. Citado na página 29.

WAZLAWICK, R. S. **Análise e design orientados a objetos para sistemas de informação**. 3. ed. Rio de Janeiro: Elsevier, 2015. Citado 2 vezes nas páginas 42 e 46.

WERNECK, V. R. Sobre o processo de construção do conhecimento: O papel do ensino e da pesquisa. v. 14, n. 51, p. 179–196, 2006. Citado na página 5.

WILDT, D. et al. **eXtreme Programming: Práticas para o dia a dia no desenvolvimento ágil de software**. 1. ed. [S.l.]: Casa do Código, 2015. Citado na página 39.

X-APPS. **Aplicativos nativos ou híbridos: qual escolher para a sua startup?** 2019. Disponível em: <https://www.x-apps.com.br/aplicativos-nativos-ou-hibridos-qual-escolher-para-a-sua-startup>. Acesso em: 04 de novembro de 2020. Citado na página 17.

ZABONI, Z. C.; IORIO, M. C. M. Reconhecimento de fala no nível de máximo conforto em pacientes adultos com perda auditiva neurossensorial. v. 3, n. 14, p. 491–497, 2009. Citado na página 25.

Apêndices

APÊNDICE A – Descrições dos casos de uso

Figura 60 – Descrição: Reconhecer fala

Caso de Uso:	Reconhecer Fala
Ator(es):	Usuário
Pré-condições:	Possuir acesso à internet.
Pós-condições:	Fala reconhecida com sucesso e reproduzida através de sinais em Libras.

	Ator		**Sistema**	
1	Abrir o aplicativo			
2	Pressionar o botão *Reconhecer* da Página Inicial.			A1, E1
3	Pronunciar a frase que deverá ser reconhecida			E2
		4	Utilizar biblioteca *react-native-tts* para transcrever a pronúncia em texto	
		5	Selecionar uma das alternativas de transcrição e enviar para a API */process-text*	
		6	Aplicar técnicas de processamento de linguagem natural e retornar o texto processado	
		7	Utilizar o retorno para realizar uma busca no Amazon DynamoDB	R1
		8	Exibir na tela os GIFs associados à frase reconhecida, obtidos através da busca	E3
		9	Encerra caso de uso	

A1	Caso seja a primeira utilização, deverá solicitar ao usuário permissão para utilizar o microfone do dispositivo.
E1	O usuário não concedeu permissão à aplicação para utilização do microfone do dispositivo. O caso de uso é encerrado.
E2	Usuário não pronunciou nenhuma frase. O caso de uso é encerrado.
R1	A consulta deve ser realizada na tabela *words,* aplicando, como filtro, os termos retornados após o processamento de linguagem natural da API.
E3	Não foram encontrados registros de sinais correspondentes às palavras reconhecidas no banco de dados Amazon DynamoDB. O aplicativo exibe uma mensagem que informa o ocorrido ao usuário e encerra o caso de uso.

Fonte: Autor.

Figura 61 – Descrição: Manter resposta

Caso de Uso:	Manter Resposta
Ator(es):	Usuário
Pré-condições:	Possuir acesso à internet; Garantir um volume mínimo de reprodução dos sons do dispositivo.
Pós-condições:	Resposta construída emitida com sucesso.

	Ator		**Sistema**	
1	Abrir o aplicativo			
2	Navegar para a aba *Emissão* na Página Inicial			
3	Pressionar o botão *Encontrar Palavras*			A1
		4	Realizar consulta no Amazon DynamoDB para retornar todas palavras com sinal associado	R1, E1
		5	Exibir um filtro horizontal por letras do alfabeto e as palavras obtidas em formato de lista	
6	Selecionar os termos que constituirão a resposta			
7	Pressionar o botão *Confirmar*			A2
		8	Montar uma lista interna com as palavras escolhidas e exibir os sinais correspondentes	R2
9	Pressionar o botão *Emitir*			
		10	Exteriorizar os termos que compõe a resposta através dos alto-falantes do dispositivo e ativar o componente de ondas sonoras durante a emissão	R3
		11	Encerra caso de uso	

A1	O usuário clica em “Voltar” (ou qualquer outro botão com ação de retrocesso, como o próprio botão físico de retorno). O caso de uso é encerrado.
R1	A consulta deve ser realizada na tabela *words* e não possuir filtros.
E1	Não foram cadastradas palavras no banco de dados Amazon DynamoDB. O aplicativo exibe uma mensagem que informa o ocorrido ao usuário e encerra o caso de uso.
A2	O usuário pressiona o botão *Desmarcar*. Retorna ao passo 6.
R2	A lista deve ser montada respeitando a ordem em que o usuário selecionou as palavras.
R3	A língua de emissão deve ser Português e a velocidade de reprodução deve ser *normal*.

Fonte: Autor.

APÊNDICE B – Solicitação de permissão de uso

Figura 62 – Relato da solicitação de uso: Início

Gmail — João Vitor Veronese Vieira <joaovitorvv@gmail.com>

Permissão de Uso - Dicionário de Libras
10 mensagens

João Vitor Veronese Vieira <joaovitorvv@gmail.com> 30 de setembro de 2020 01:05
Para: diesp@ines.gov.br

Olá, tudo bem?

Meu nome é João Vitor Veronese Vieira e sou acadêmico do curso de Ciência da Computação da Universidade Regional Integrada do Alto Uruguai e das Missões (URI - Erechim), 10º semestre. Estou fazendo meu trabalho de conclusão de curso neste momento e o mesmo trata-se de um aplicativo para traduzir Português para Libras através do reconhecimento de fala.

Dito isso, durante as pesquisas, me deparei com o site: *http://www.acessibilidadebrasil.org.br/libras_3/* e vi que existem diversos GIFs ali exibidos. Nos créditos da página, vi que o responsável pelo projeto é o próprio INES e que antigamente uma cópia via CD-Rom era disponibilizada.

Sendo assim, gostaria de pedir, se for possível, permissão para usar o conteúdo do site como fonte de estudo e testes. Peço isso pois meu aplicativo usará GIFs na tradução e o site de vocês é simplesmente incrível, parabéns!

Obs: caso precise de um documento formal, por favor me avisem que eu providencio.

Desde já, agradeço e conto muito com a colaboração de vocês.
Tenham um ótimo dia.

--
Atenciosamente,
João Vitor.

Divisão de Estudo e Pesquisa <diesp@ines.gov.br> 1 de outubro de 2020 11:29
Para: Coordenação COPET <copet@ines.gov.br>, joaovitorvv@gmail.com

Olá Vitor,

Agradecemos o seu contato e seu interesse em atuar na acessibilidade para surdos.

Estou encaminhando seu pedido para a Coordenação de Projetos Especiais e Tecnológicos (COPET) que poderá te responder sobre essa questão e nos lê em cópia.

Pedimos que aguarde o contato do referido setor.

Estamos a disposição para outras dúvidas e inormações

Atenciosamente,

Jean F. de Paiva
Chefe da Divisão de Estudos e Pesquisas
[Texto das mensagens anteriores oculto]

João Vitor Veronese Vieira <joaovitorvv@gmail.com> 1 de outubro de 2020 13:38
Para: Divisão de Estudo e Pesquisa <diesp@ines.gov.br>
Cc: Coordenação COPET <copet@ines.gov.br>

Olá Jean!

Agradeço imensamente o retorno e aguardarei, portanto, o contato do COPET.
Essa permissão, se possível, será de grande ajuda para o sucesso do meu trabalho mas, principalmente, para a criação de uma ferramenta útil que, quem sabe, poderá ser útil em pouco tempo :)

Novamente, reitero meu agradecimento e aguardarei.
Tenham um ótimo dia!
[Texto das mensagens anteriores oculto]
--

Fonte: Autor.

Figura 63 – Relato da solicitação de uso: Meio

Atenciosamente,
João Vitor.

João Vitor Veronese Vieira <joaovitorvv@gmail.com> 12 de outubro de 2020 17:16
Para: Divisão de Estudo e Pesquisa <diesp@ines.gov.br>
Cc: Coordenação COPET <copet@ines.gov.br>

Olá pessoal, tudo bem com vocês?

Houve alguma atualização sobre o *status* da permissão de uso?
Gostaria de reiterar o foco completamente educacional (meu Trabalho de Conclusão de Curso) e o quanto ajudaria ter a permissão de vocês para utilizar um material tão importante.

Muito obrigado desde já.
[Texto das mensagens anteriores oculto]
--
Atenciosamente,
João Vitor.

João Vitor Veronese Vieira <joaovitorvv@gmail.com> 27 de outubro de 2020 20:13
Para: Divisão de Estudo e Pesquisa <diesp@ines.gov.br>
Cc: Coordenação COPET <copet@ines.gov.br>

Boa noite pessoal, tudo bem?

Existe alguma outra forma que eu possa entrar em contato para que possamos seguir com o assunto?
Reforço que a data de entrega do meu trabalho aproxima-se e eu gostaria muito de poder contar com a permissão de uso desse material que, por natureza, é público e tem a intenção de aumentar a acessibilidade para todos.

Caso exista este outro meio, peço que, por gentileza, vocês possam me informar e eu utilizarei para que possamos seguir.
Agradeço imensamente.
[Texto das mensagens anteriores oculto]
--
Atenciosamente,
João Vitor.

Divisão de Estudo e Pesquisa <diesp@ines.gov.br> 29 de outubro de 2020 22:14
Para: João Vitor Veronese Vieira <joaovitorvv@gmail.com>

Olá João Vitor,

Espero que esteja bem.

Sentimos que ainda não tenha conseguido uma resposta. Então, reencaminhamos novamente seu pedido para a Coordenação que citei em nosso último contato e para a Direção Geral do Ines como uma última instância para resolver seu caso.

Caso queira entrar em contato via telefone com o INES, você pode ligar para o telefone (21) 2285-7546 e digitar o ramal 116.

Espero ter auxiliado,

Atenciosamente,

Jean F. de Paiva
Chefe da Divisão de Estudos e Pesquisas
[Texto das mensagens anteriores oculto]

João Vitor Veronese Vieira <joaovitorvv@gmail.com> 29 de outubro de 2020 23:44
Para: Divisão de Estudo e Pesquisa <diesp@ines.gov.br>

Olá Jean,
Grato, mais uma vez, por sua colaboração!

Adoraria falar por telefone. Em que horário e dia fica melhor para você?
[Texto das mensagens anteriores oculto]

Fonte: Autor.

Figura 64 – Relato da solicitação de uso: Fim

Divisão de Estudo e Pesquisa <diesp@ines.gov.br> 5 de novembro de 2020 10:55
Para: João Vitor Veronese Vieira <joaovitorvv@gmail.com>

Bom dia João,

Peço desculpas pela demora em responder, pois estamos envolvidos em um evento on-line que acontecerá hoje no Youtube do INES e não tivemos tempo hábil de responder antes.

Então, com esse número que te passei no e-mail anterior você consegue falar direto com a Direção Geral do INES, facilitando você a resolver a questão da liberação institucional dos GIFs do CD Dicionário de Libras.

O nosso telefone é outro (021 2205-0224) e nele você consegue tirar dúvidas sobre outros assuntos, como por exemplo, publicações, pesquisas e distribuição de materiais. Porém, devido a pandemia e com parte dos servidores em grupo de risco, a instituição está com grande parte de seu efetivo em trabalho remoto, inclusive a nossa equipe. Assim, no momento, você não vai conseguir ter atendimento pelo telefone, somente por esse e-mail.

Mas o telefone da Direção Geral está funcionando normalmente, e lá você consegue ser atendido para essa questão mais rapidamente. Além disso, como já tinha encaminhado seu pedido para eles antes, provavelmente eles já devem ter ciência do seu caso.

Espero ter auxiliado,

Atenciosamente,

Jean F. de Paiva
Chefe da Divisão de Estudos e Pesquisas
[Texto das mensagens anteriores oculto]

João Vitor Veronese Vieira <joaovitorvv@gmail.com> 5 de novembro de 2020 11:08
Para: Divisão de Estudo e Pesquisa <diesp@ines.gov.br>

Bom dia Jean,

Não há dúvidas alguma que auxiliou! Sua ajuda foi fundamental para a resolução dessa situação e eu apenas posso lhe agradecer por isso.
Vou ligar lá neste momento para tentar conseguir essa permissão ainda hoje.

Atualizarei esta *thread* de e-mail conforme evoluir o assunto.
Novamente, gratidão!
[Texto das mensagens anteriores oculto]
--
Atenciosamente,
João Vitor.

João Vitor Veronese Vieira <joaovitorvv@gmail.com> 5 de novembro de 2020 11:28
Para: Divisão de Estudo e Pesquisa <diesp@ines.gov.br>, dirge@ines.gov.br

Bom dia a todos! Tudo bem?
Meu nome é João Vitor Veronese Vieira e sou estudante do curso de Ciência da Computação da URI Erechim - Rio Grande do Sul.

Estou compartilhando com vocês a *thread* de e-mail onde eu solicito, se possível, permissão para usar os GIFs do Dicionário de Libras disponíveis no endereço *http://www.acessibilidadebrasil.org.br/libras_3/* em meu Trabalho de Conclusão de Curso.
Dentro da Divisão de Estudos e Pesquisas, o Jean me auxiliou repassando o contato da Direção Geral. Liguei lá agora pouco e falei com a Renata, que acredito que pode me auxiliar na explicação da situação.

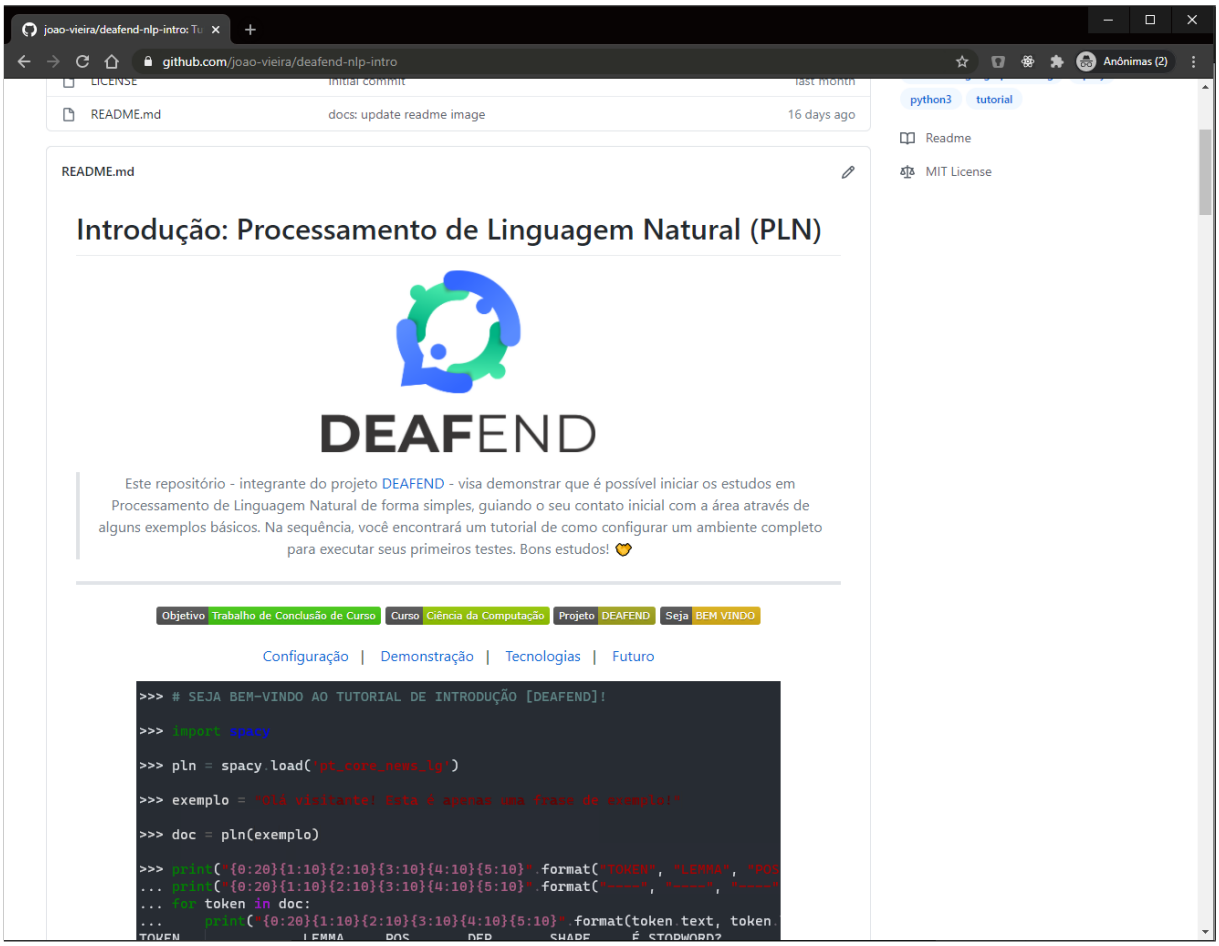
Basicamente, eu apenas preciso da ciência e concordância do INES para poder utilizar as imagens no meu projeto, uma vez que essas imagens apresentam a intérprete.
Como é um *site* público e, há tempos atrás, havia a disponibilização de um CD com essas imagens, acredito que podemos seguir o mesmo protocolo agora, mas de forma digital.

Desde já, agradeço imensamente o entendimento e colaboração. Estou à disposição para qualquer dúvida.
[Texto das mensagens anteriores oculto]
--
Atenciosamente,
João Vitor.

Fonte: Autor.

APÊNDICE C – Tutorial introdutório ao PLN

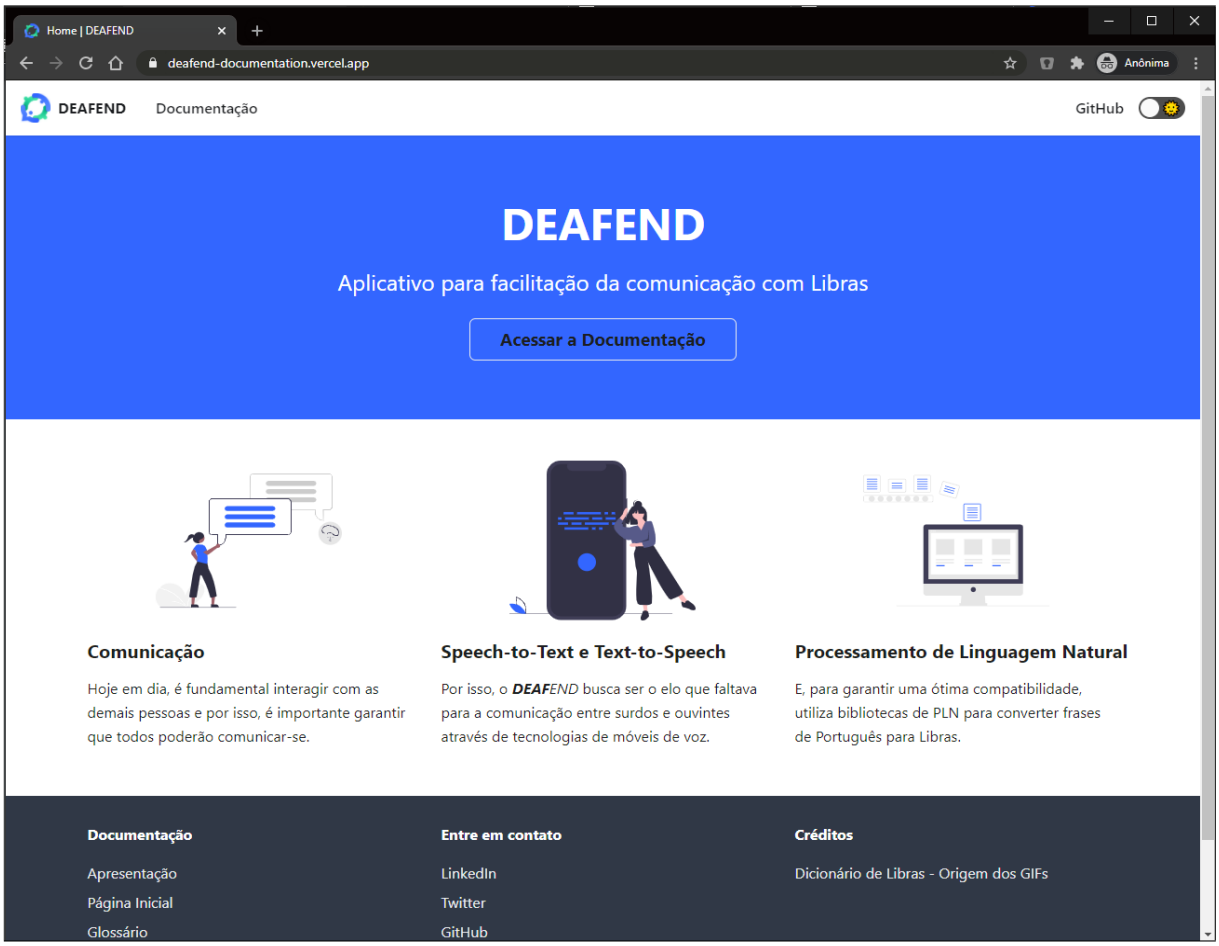
Figura 65 – Apresentação do tutorial introdutório ao PLN



Fonte: Autor.

APÊNDICE D – Manual do usuário *online*

Figura 66 – Apresentação do manual do usuário *online*



Fonte: Autor.