

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**APLICAÇÃO MULTIMODAL
PARA PESSOAS COM
NECESSIDADES COMPLEXAS
DE COMUNICAÇÃO**

BRUNO RAUPP KIELING

Trabalho de Conclusão I apresentado
como requisito parcial à obtenção
do grau de Bacharel em Ciência da
Computação na Pontifícia Universidade
Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Michael da Costa Mora

**Porto Alegre
2020**

DEDICATÓRIA

Dedico este trabalho a meus pais, Rejane Maria Stumpf Raupp e Aurélio Gageiro Kieling, que sempre acreditaram em mim independente de meus diversos defeitos. Também dedico este trabalho a todos outros integrantes de minha família, avós, tios, tias, primos e meu irmão Arthur Raupp Kieling.

E meu coração nunca esquecera meu melhor amigo e confidente, meu avô Léo Antônio Coelho Raupp que sempre me apoiou em momentos de alegria e tristeza e minha amada tia e madrinha Rosane Maria Stumpf Raupp que me ensinou a dar meus primeiros passos, e que diante de dificuldades, demonstrou força e alegria.

“Vale muito a pena viver!”
(Paulo Henrique Machado)

AGRADECIMENTOS

Gostaria de agradecer a minha mãe Rejane Maria Stumpf Raupp, meu pai Aurélio Gageiro Kieling pelas oportunidades proporcionadas em toda minha vida.

Também gostaria de agradecer aos meus amigos Maurício Gomes Alexandre, Bernardo Furtado e Renata Urbanski que disponibilizaram seu tempo ajudando a encontrar erros neste trabalho e dando ideias para o mesmo.

APLICAÇÃO MULTIMODAL PARA PESSOAS COM NECESSIDADES COMPLEXAS DE COMUNICAÇÃO

RESUMO

A Comunicação Alternativa Aumentativa de alta tecnologia procura facilitar a comunicação e o desenvolvimento de pessoas com diferentes tipos de deficiências através de dispositivos e aplicações moldados para específicas necessidades. O desenvolvimento desta aplicação procura encontrar uma alternativa para atingir o máximo possível de usuários com necessidades complexas de comunicação através de um *layout* modular. Sendo desenvolvido para iOS e Android. A aplicação explora as vantagens de uma tela de tamanho grande para facilitar a edição dos *layouts*.

Palavras-Chave: comunicação, alternativa, aumentativa, CAA, tablet, deficiências, iOS, android, multimodal .

A COMPLEX COMMUNICATION NEEDS MULTI-MODAL APPLICATION

ABSTRACT

High tech Alternative Augmentative Communication seeks to facilitate communication and development of people with different types of disabilities through devices and applications molded for specific needs. The development of this application seeks to find an alternative to achieve the maximum possible number of users with complex communication needs through a modular layout. Developed for the iOS and Android, the application explores the advantages of a large screen to facilitate the editing of layouts.

Keywords: communication, AAC, tablet, needs, iOS, android, multi-modal.

LISTA DE FIGURAS

Figura 2.1 – Exemplo de foto usada em um VSD. Fonte: (Akshar Dave from Pexels)	15
Figura 2.2 – Imagem de um <i>layout</i> com <i>VSDs</i> para adultos. Fonte: [LWT ⁺ 19]	17
Figura 2.3 – Captura de tela tirada do APP Sono Flex	17
Figura 3.1 – Captura de tela do Sono Flex no iPad [Fle19a].....	18
Figura 3.2 – Captura de tela do Proloquo2go em iPad[Ass19b]	19
Figura 3.3 – Captura de tela do MyTalkTools em iPad [nHEL19b]	20
Figura 4.1 – Seleção de layout	21
Figura 4.2 – Layout para deficiência em desenvolvimento	22
Figura 4.3 – Layout para crianças com Síndrome de Down	23
Figura 4.4 – Layout para Adultos com deficiência adquirida	24
Figura 4.5 – Alternativas a serem editadas.	25
Figura 4.6 – Seleção de caixa de texto para atualização de frases.	25
Figura 4.7 – Layout para Adultos com deficiência adquirida	26
Figura 5.1 – <i>Canvas</i>	27
Figura 5.2 – Criação de VSD.	28
Figura 5.3 – Barra de Navegação Horizontal.	30
Figura 6.1 – Exemplo de árvore de Widgets retirada de um artigo por Andrea Bizzotto [Biz20]	32
Figura 7.1 – Campo de escrita com T2S	35
Figura 7.2 – VSD com lista com componentes com T2S	35
Figura 7.3 – Representação de uma grid.	36
Figura 7.4 – Canvas sem <i>Widgets</i> posicionados.	37
Figura 7.5 – Em vermelho a barra de navegação.....	38
Figura 7.6 – VSD para crianças com deficiência ao selecionar um morango no bolo.	39
Figura 7.7 – <i>Grid</i> para crianças com deficiência.	40
Figura 7.8 – Exemplo do flutter staggered grid view.....	41
Figura 7.9 – Exemplo de <i>Widgets</i> em uma <i>Stack</i>	42
Figura 7.10 – Exemplo retirado da documentação do <i>expanded_grid</i>	43

LISTA DE TABELAS

Tabela 3.1 – Comparações	18
------------------------------------	----

LISTA DE SIGLAS

AAC – Alternative Augmentative Communication

CAA – Comunicação Aumentativa e Alternativa

CCN – *Complex Communication Needs*

VSD – *Visual Scene Display*

AVC – Acidente Vascular Cerebral

SSD – *Solid State Disk*

HD – *Hard Drive*

GB – *Gigabyte*

T2S – *Text To Speech*

LISTA DE ABREVIATURAS

ASHA. – *American Speech-Language-Hearing Association*

TEA. – Transtorno do Espectro Autista

APP. – *Application*

UI. – *User Interface*

RAM. – *Random Access Memory*

TCC. – Trabalho de Conclusão de Curso

UML. – *Unified Modeling Language*

SUMÁRIO

1	INTRODUÇÃO	13
1.1	OBJETIVO	13
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	COMUNICAÇÃO ALTERNATIVA AUMENTATIVA	14
2.2	DISPOSITIVOS	14
2.3	<i>VISUAL SCENE DISPLAY - VSDS</i>	14
2.4	NAVEGAÇÃO EM VSDS	15
2.4.1	NAVEGAÇÃO PARA CRIANÇAS	16
2.4.2	NAVEGAÇÃO PARA ADULTOS	16
2.5	<i>GRID LAYOUT</i>	17
3	SISTEMAS DE CAA SEMELHANTES	18
3.1	SONO FLEX	18
3.2	PROLOQUO2GO	19
3.3	MYTALKTOOLS	19
4	ESTRUTURA VISUAL	21
4.1	MODELOS PADRÃO	21
4.2	CRIANÇAS COM DEFICIÊNCIA EM DESENVOLVIMENTO	22
4.3	SÍNDROME DE DOWN	22
4.4	ADULTOS	23
4.4.1	EDIÇÃO DE TEXTOS	24
4.4.2	TECLADO	26
5	MODULARIZAÇÃO DO DESIGN	27
5.1	CANVAS	27
5.2	PEÇAS	28
5.2.1	VSD	28
5.2.2	HOTSPOTS	29
5.2.3	COLEÇÃO DE VSDS	29
5.2.4	BLOCO DE TEXTO	29
5.2.5	BARRA DE NAVEGAÇÃO	30

6	DEFINIÇÕES DO AMBIENTE DE DESENVOLVIMENTO	31
6.1	AMBIENTE DO USUÁRIO	31
6.2	FLUTTER	31
6.3	LINGUAGEM DART	32
6.4	AMBIENTE DE DESENVOLVIMENTO	32
7	DESENVOLVIMENTO	34
7.1	FERRAMENTAS	34
7.1.1	PUB.DEV	34
7.1.2	FLUTTER TTS	34
7.1.3	EXPANDED GRID	35
7.2	ELEMENTOS	36
7.2.1	CANVAS	37
7.2.2	BARRA DE NAVEGAÇÃO	38
7.2.3	VSDS	38
7.2.4	CARDS	39
7.3	DIFÍCULDADES	40
7.3.1	TELA DE EDIÇÃO	42
7.4	SOLUÇÕES	42
7.4.1	REFATORAÇÕES	43
8	CONCLUSÃO	44
	REFERÊNCIAS	45

1. INTRODUÇÃO

Este projeto foi criado para o Trabalho de Conclusão do curso de Ciências da Computação da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Nele são analisadas diferentes abordagens de alta tecnologia para a área de Comunicação Alternativa Aumentativa.

Pessoas que sofrem de deficiências muitas vezes precisam de ajuda para se sentirem parte da sociedade [CBH17]. A Comunicação Alternativa Aumentativa procura ajudar aqueles que não possuem voz, aqueles com dificuldades de se expressarem, devido a vários motivos, como deficiências de nascença ou adquirida. Essas pessoas acabam precisando de meios diferentes para se encaixarem, e é através de *softwares* ou *hardwares* que tornam seus relacionamentos mais confortáveis.

Neste trabalho procuramos entender o significado de Comunicação Alternativa Aumentativa, suas diferentes aplicações e estudos, os diferentes tipos de *design* sugeridos por pesquisadores para diferentes grupos e finalizamos com a aplicação dessas pesquisas e sugestões através de um APP multimodal que pode ser usado por usuários com qualquer tipo de afasia, que não afete sua coordenação motora.

1.1 Objetivo

O objetivo é o desenvolvimento de uma aplicação multimodal com a capacidade de se adaptar as principais necessidades daqueles que possuem necessidades complexas de comunicação (*Complex Communication Needs-CCN*).

Através de um *layout* responsivo será possível trabalhar em cima de uma *grid*, organizando objetos em células com diferentes tamanhos e formas conforme as necessidades ou preferências de seus usuários. Será possível adicionar a esta *grid* outros elementos também, como caixas de texto redimensionáveis, inserir dentro das células imagens, VSDs e textos acompanhados de possíveis pontos de referência, dividir os elementos em categorias (ex.: célula 2x2 com 4 células menores representando informações semelhantes), criar pastas direcionáveis para diferentes *displays* e selecionar textos para o uso de T2S.

A aplicação também disponibilizará *layouts* padrões sugeridos no artigo [LWT⁺19] para crianças com diferentes tipos de necessidades e adultos. O padrão será estruturado com base nas diferentes necessidades desse grupo e separados por suas específicas deficiências. Também será possível editar os *layouts* padrões e criar novos para se acomodar ao usuário.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Comunicação Alternativa Aumentativa

Alternative Augmentative Communication (AAC) são diversas maneiras, ou alternativas para facilitar ou compensar dificuldades encontradas em pacientes com necessidades complexas de comunicação. Essa área abrange uma grande gama de deficiências, tanto aqueles que possuem de nascença, autismo, síndrome de down e aqueles que adquiriram causadas por problemas clínicos ou acidentes. Para cada uma dessas situações existem necessidades diferentes que precisam ser identificadas e tratadas da maneira adequada por pesquisadores[CBH17].

Existem várias formas de AAC, aquelas que se utilizam de dispositivos tecnológicos como computadores, celulares, *tablets*, até as mais básicas sem o uso de tecnologias ou o mínimo possível, como libras e placas com símbolos.

2.2 Dispositivos

Com a evolução da tecnologia, novas alternativas foram criadas para facilitar a inclusão dessas pessoas na sociedade. Dispositivos como *tablets* e celulares tem o atrativo de serem pessoais, de fácil manuseio e acessíveis. Computadores por sua vez trazem diversas formas de personalização para o seu uso, e ainda existem hardwares específicos como os da Telepatix que se utilizam da tecnologia de Eye-Tracking em conjunto a um *display*.

Esses dispositivos dependem de aplicativos capazes de suprirem as necessidades de seus usuários, entre eles crianças com deficiências de desenvolvimento, síndrome de down, pessoas que sofreram algum tipo de lesão cerebral dificultando suas habilidades de comunicação. Grande parte desses aplicativos utiliza especificações de layout tentando atingir um público específico, a partir de VSDs e *grid layout* com imagens significativas para os usuários.

2.3 *Visual Scene Display - VSDs*

Os VSDs, Visual Scene Display, são representados por imagens como fotos, desenhos ou vídeos que são responsáveis por comunicar uma mensagem significativa[LWT⁺19].

Mensagens de texto também podem ser pré-programadas para identificar um local, um objeto ou uma ação.

Em uma VSD básica como a da Figura 2.1, o usuário pode selecionar pontos de interesse para expressar:

- Felicidade ao apontar para o rosto;
- Indicar o violão para se referir a música;
- Demonstrar interesse em tocar um instrumento;



Figura 2.1 – Exemplo de foto usada em um VSD. Fonte: (Akshar Dave from Pexels)

2.4 Navegação em VSDs

Um dos maiores desafios é como navegar por múltiplos VSDs. A inserção de informações pode poluir a tela confundindo o usuário além de prejudicar o avanço do aprendizado caso esse seja o objetivo. Estudos com crianças e adultos, com diferentes dificuldades complexas de comunicação decorrente de problemas de nascença ou adquiridos, demonstrou a necessidade de layouts com uma variação de posicionamento de seus elementos.

2.4.1 Navegação para Crianças

Geralmente o menu de navegação dos APPs de AAC com VSDs estão presentes em uma página diferente, organizados com o *layout* em *grid*. A criança acessa essas páginas apertando no símbolo do contexto que deseja navegar para abrir o VSD[LWT⁺¹⁹].

Para crianças que estão aprendendo a se comunicar, este *layout* apresentou dificuldades em sua navegação, já que o próximo *display* está escondido do campo de visão, os símbolos representando navegação podem não ser reconhecidos e/ou pode ser difícil lembrar a sequência de VSDs [LWT⁺¹⁹]. Em pesquisa feita por Drager, com crianças de três anos de idade sem problemas de desenvolvimento, demonstrou-se que a navegação por *thumbnails* de VSDs era mais eficiente do que símbolos em um *display* separado. [DLC⁺⁰⁴]

Foi constatado por O'Neill que grupos de crianças com deficiências de desenvolvimento (síndrome de down, deficiência intelectual, TEA), passavam mais tempo visualizando o VSD principal ($M = 61\text{-}67\%$) do que a barra de navegação($M = 30\text{-}33\%$), mesmo com o foco no VSD elas demonstraram consciência dos *thumbnails* presentes na barra de navegação. [OWL19]

Outro ponto importante é a localização dos VSDs e da barra de navegação, sendo as principais funcionalidades de um APP de AAC, é necessário que suas posições se adaptem as diferentes características das crianças, como por exemplo, crianças com deficiência de desenvolvimento passavam mais tempo olhando os VSDs, com atividades em conjunto, quando a barra de navegação ficava localizada na extremidade inferior da tela enquanto crianças com síndrome de down, ASD e deficiência intelectual concentravam-se no rosto das pessoas quando a barra de navegação ficava localizada na extremidade superior da tela.[OWL19]

2.4.2 Navegação para Adultos

Não somente para crianças, APPs de AAC podem ser utilizados para adultos com afasia decorrente de traumatismo craniano, AVCs e tumores cerebrais. No entanto existem algumas diferenças de *layout* comum entre os APPs, entre elas VSDs tendem a possuir caixas de texto a seu lado sem o uso de pontos de interesse (*hotspots*) [LWT⁺¹⁹], a organização dos *thumbnails* tendem a ser em volta do VSDs principal, e não só os texto são personalizados como os VSDs são preferivelmente compostos por fotos ou imagens fornecidas pelo usuário.[MDH⁺⁰⁷]

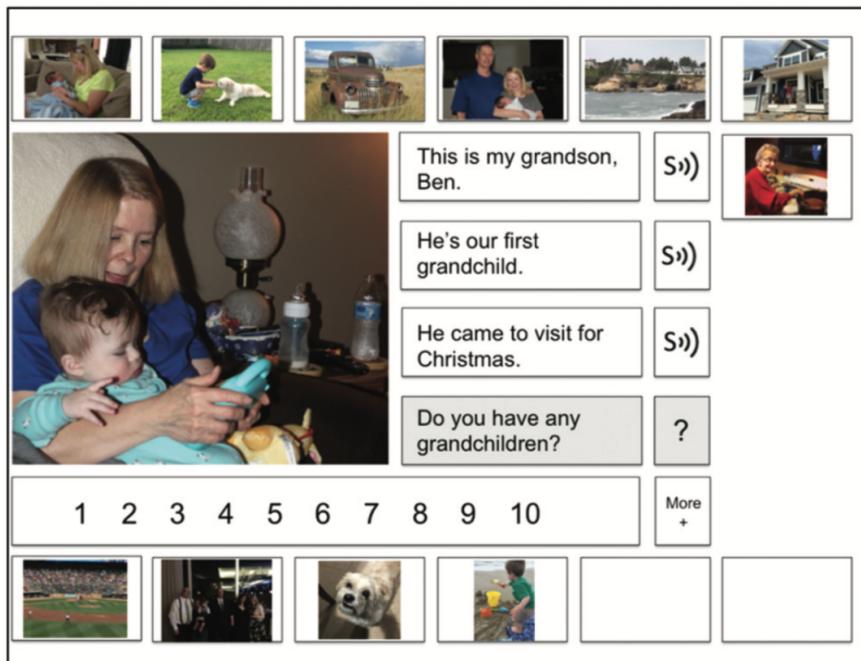


Figura 2.2 – Imagem de um *layout* com *VSDs* para adultos. Fonte: [LWT⁺19]

2.5 Grid Layout

O *layout* mais utilizado entre os APPs de AAC é o *Grid Layout*. Ele disponibiliza um *overview* de múltiplos contextos que levam para outros *displays* com imagens significativas referente ao contexto escolhido, principalmente acompanhados de palavras. Em sua grande maioria é encontrado uma caixa de texto que grava as células selecionadas do *grid* (Figura 2.3).

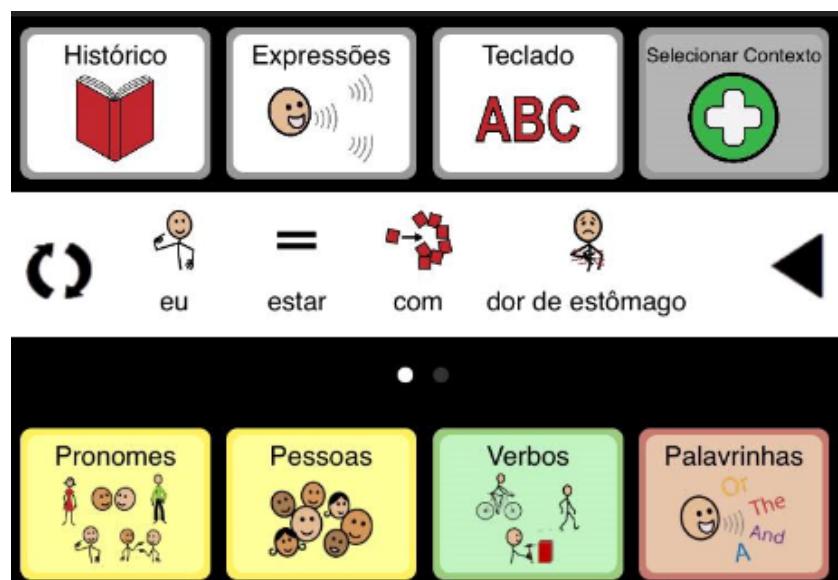


Figura 2.3 – Captura de tela tirada do APP Sono Flex

3. SISTEMAS DE CAA SEMELHANTES

Na busca por referências que abrangem a área de CAA de alta tecnologia, foram encontradas várias alternativas pertinentes, entre elas Sono Flex, Proloquo2go e MyTalkTools. A pesquisa se limitou a aplicativos com disponibilidade em plataformas *mobile*, principalmente para dispositivos como *tablet* e *smartphones* em lojas virtuais como a App Store, nos dispositivos da APPLE, e na Google Play, nos dispositivos que possuem sistema operacional Android. Todas essas aplicações possuem o mesmo princípio que é o uso do *touchscreen* para a interação do usuário com o *software*, seus *designs* seguem padrões semelhantes, entre eles o uso de texto e símbolos simples para a comunicação.

Tabela 3.1 – Comparações

Nome APP	Plataforma	Valor*
Sono Flex	iOS e Android	\$99,99
Proloquo2go	iOS	\$249,99
MyTalkTools	iOS e Android	\$99,99

3.1 Sono Flex

Desenvolvido para a plataforma iOS, o Sono Flex está disponível em quatro línguas, inglês, português, sueco e alemão, todos com uma versão grátis. Este APP faz parte de uma coleção de sistemas de CAA desenvolvidos pela empresa Tobii Dynavox LLC em diferentes tipos de plataformas incluindo *hardware* proprietário. Seu principal objetivo é ajudar usuários com dificuldades de fala, promete uma interface simples em forma de *grid* com uso de *touchscreen* [Fle19b] como na figura 3.1.

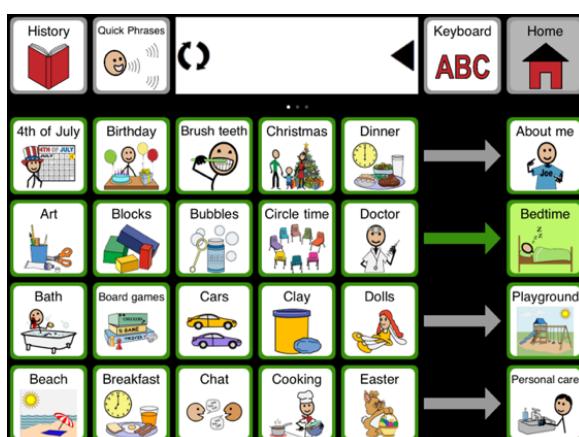


Figura 3.1 – Captura de tela do Sono Flex no iPad [Fle19a]

Este aplicativo ainda disponibiliza um histórico de frases, um teclado com letras em ordem alfabética e a possibilidade de adicionar diferentes contextos já pré-definidos. O aplicativo está disponível na APP Store por 99,99 dólares em versão em inglês e 179,99 reais em português.

3.2 Proloquo2go

Desenvolvido pela AssistiveWare, o Proloquo2go sugere que o uso de palavras chave é significativo para uma comunicação eficiente. O uso delas em posições pré-definidas ainda facilita o treinamento de memória muscular.[Ass19a] Com mais de duzentos mil usuários, Proloquo2go possui mais de dez mil palavras chaves, possibilidade de personalização do vocabulário e *layout*, suporte bilíngue, cem vozes diferentes para várias linguás.[Ass19a]



Figura 3.2 – Captura de tela do Proloquo2go em iPad[Ass19b]

Disponível para dispositivos APPLE, é possível adquirir o aplicativo na loja da APP Store por 249,99 dólares.

3.3 MyTalkTools

MyTalkTools foi o aplicativo que apresentou uma interface de *grid* com melhor visualização. Seu propósito segue o padrão dos aplicativos de CAA: palavras chave organizadas por contexto. Como adicional possui a funcionalidade de enviar mensagens em SMS para contatos da agenda telefônica. A desenvolvedora 2nd Half Enterprises LLC disponibiliza o aplicativo na plataforma Android e iOS (iPad e iPhone). [nHEL19a]

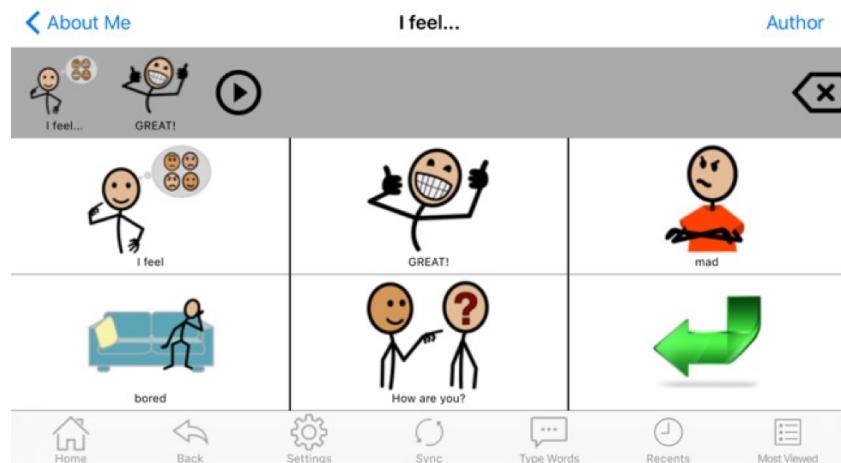


Figura 3.3 – Captura de tela do MyTalkTools em iPad [nHEL19b]

Com acesso a mais de vinte mil símbolos, ainda é possível adquirir uma ferramenta Web para a personalização do aplicativo, adicionando palavras, símbolos, alterando *layout*, além de serem oferecidos quatro tipos de serviço, *Workspace Basic* gratuita, *Workspace Family* por 75 dólares ou 35 dólares anuais, *Workspace DEMO* com 30 dias gratuitos e ao final desses, a possibilidade de usar um certificado da ASHA, e a versão *Workspace Professional* por 120 dólares ou anuidade de 57 dólares.

4. ESTRUTURA VISUAL

4.1 Modelos Padrão

Como visto no Capítulo 2, existem vários estudos e sugestões de como devem ser modelados aplicativos para diferentes tipos de necessidades. Visando buscar a melhor forma para satisfazer as necessidades dessas pessoas, modelos padrões serão disponibilizados para acesso imediato, quando se entra no aplicativo sem que seja preciso criar um layout.

Esses modelos são baseados nos estudos encontrados nos artigos que este trabalho se inspirou. Esses artigos trazem claramente especificações de como as aplicações devem se comportar em cada situação. Por exemplo, nos casos de crianças com deficiência em desenvolvimento, foi avaliado que a melhor posição da barra de navegação deve se encontrar na parte inferior da tela, enquanto o VSD logo acima [LWT⁺19]. Já nos casos de Síndrome de Down, foi identificado que a melhor localização da barra de navegação seria logo acima do VSD, localizado no centro.

Ao iniciar a aplicação o usuário poderá escolher opções conforme suas necessidades, com a opção também de criação de layout que será abordada em seguida aos modelos padrões.

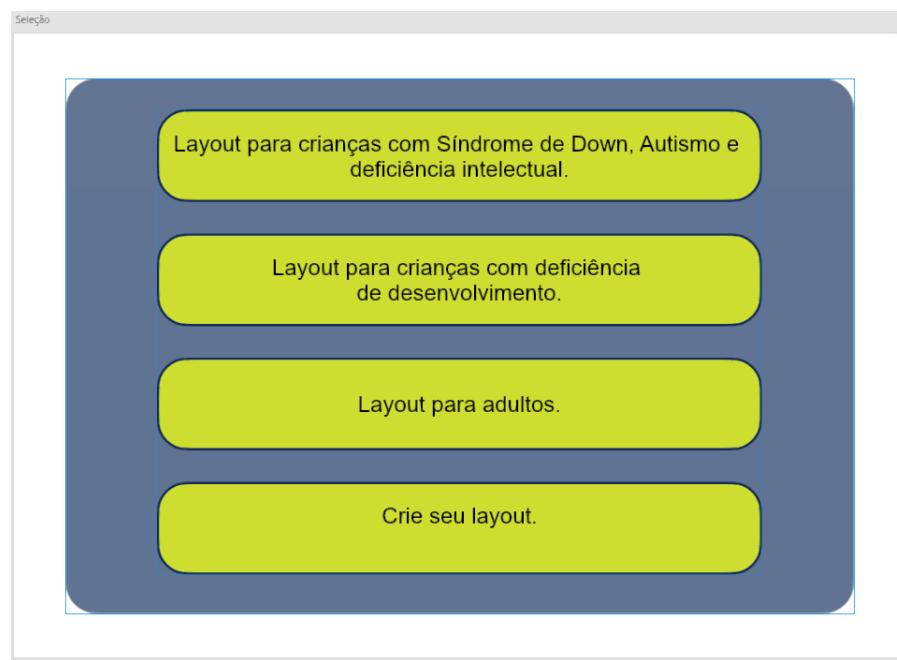


Figura 4.1 – Seleção de layout

4.2 Crianças com Deficiência em Desenvolvimento

Os modelos padrão procuram seguir uma *Grid* para que possam ser editados se caso necessário. No caso de crianças com deficiência em desenvolvimento, seu layout procura seguir a sugestão do artigo [LWT⁺19].

Neste layout, a barra de navegação ficou localizada na parte inferior da tela, logo abaixo do VSD. Dentro desta barra, se encontram outras fotos que representam categorias com contexto social. O uso de texto para identificar essas categorias não é algo eficiente para usuários com deficiência em desenvolvimento, mas podem ser adicionados caso seja constatado que ajude-o.

Com o VSD centralizado, é possível navegar por suas diversas fotos através de *swipe* para direita e esquerda ou clicando nas fotos seguintes ou anteriores. Nesses VSD existem pontos de interesse que ao serem clicados apresentam um texto identificando nome de objetos, pessoas, ações, como na figura 4.2.



Figura 4.2 – Layout para deficiência em desenvolvimento

4.3 Síndrome de Down

Na figura 4.3 é demonstrado como ficaria o design para pessoas com Síndrome de Down. Tem, portanto, a barra de navegação logo acima da foto representando um VSD. Nesta barra de navegação se encontram fotos menores que representam diferentes tipos de

categorias em um contexto social, se encaixando entretenimento, atividades com pessoas, alimentos, brinquedos, esportes e socialização. A categoria selecionada se encontra com uma marcação indicando estar ativa, enquanto o seu conteúdo se encontra nos VSDs.

Todos os VSDs deste padrão compartilham o mesmo contexto da foto selecionada na barra de navegação. É possível navegar pelas imagens através de *swipe* (deslizamento do dado na tela) ou apertando nas imagens seguintes ou anteriores.

Assim como no layout de deficiência em desenvolvimento, pontos de interesses podem parecer nas imagens com um texto no quadro inferior, podendo se referir ao que está acontecendo ou nomeando objetos.

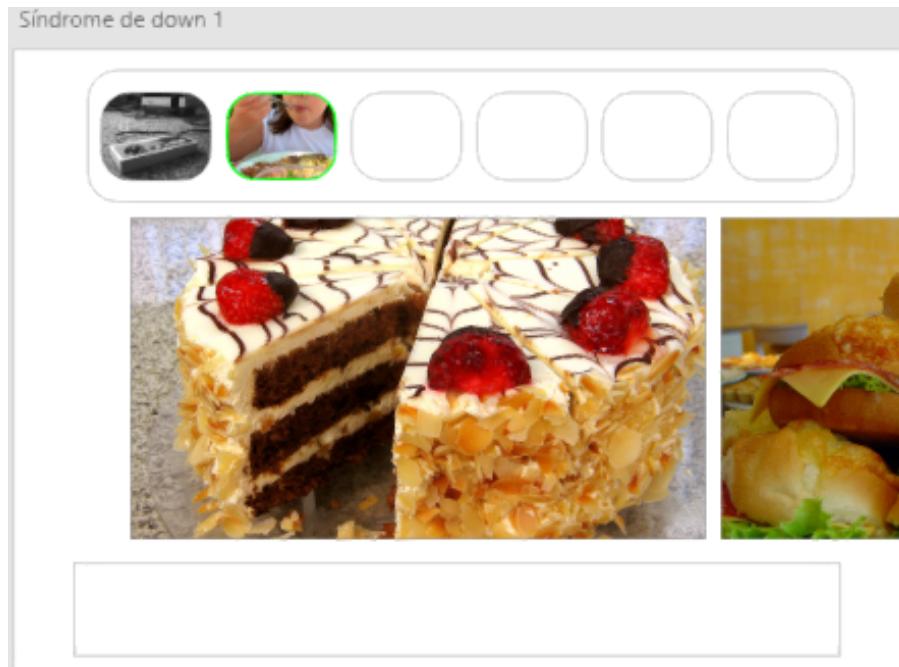


Figura 4.3 – Layout para crianças com Síndrome de Down

4.4 **Adultos**

Para o caso dos adultos, o *layout* deve ser de preferência modular, pois não possuem a necessidade de aprendizado. No entanto, também é mais difícil definir um layout padrão, variáveis como idade, escolaridade, nível de deficiência, devem ser levadas em consideração.

É necessário que haja uma forma de fácil acesso à aplicação, sem a necessidade de se montar um *layout*. Por causa deste motivo, na figura 4.4, vemos um exemplo sugerido por Light [LWT⁺19].

Com a barra de navegação localizadas no topo e na parte inferior da tela, é possível acessar imagens significativas ou selecionar categorias para representar novos contextos.

Ao apresentar uma imagem em destaque, este layout também trará textos pré-definidos de acesso rápido para que a pessoa possa se expressar. Devido a complexidade e o número de situações diferentes que podem ser identificadas em uma foto, será possível editar os textos presentes.



Figura 4.4 – Layout para Adultos com deficiência adquirida

4.4.1 Edição de Textos

Frases são importantes para a comunicação entre indivíduos. Por isso algumas frases com contexto genérico serão adicionadas no modelo padrão para adultos. No entanto, essas frases podem não atender as várias necessidades do usuário. Para habilitar uma pequena edição nelas, telas de edição estarão presentes entre as configurações.



Figura 4.5 – Alternativas a serem editadas.

Ao clicar em editar, o usuário será apresentado a uma tela semelhante a figura 4.5 com as opções já inseridas e a possibilidade de selecionar uma e editá-la, assim como cancelar a edição fechando a mesma. Seguindo um formato básico de inserção de texto, podemos ver na figura 4.6 uma caixa de texto e suas opções de interação.



Figura 4.6 – Seleção de caixa de texto para atualização de frases.

Ao selecionar a caixa de texto, se abrirá um teclado integrado do sistema (iOS). O usuário poderá digitar o que acredita ser pertinente e confirmar ou cancelar. Independente do texto inserido, o programa será capaz de ler o que foi escrito pelo usuário através de *text-to-speech* (T2S).

4.4.2 Teclado

Não só será possível a edição dos textos pré-definidos, como haverá a possibilidade de digitar textos através do teclado integrado do *tablet*, como na figura 4.7.

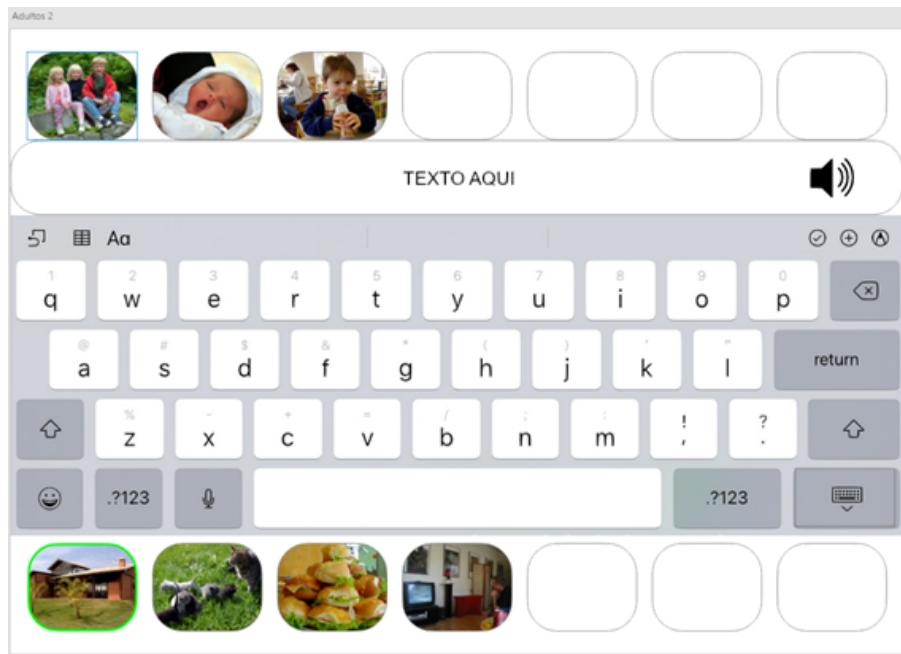


Figura 4.7 – Layout para Adultos com deficiência adquirida

A funcionalidade de T2S estará presente nesta tela. Para retornar a tela anterior é só interagir com qualquer VSD da tela ou fechar o teclado.

5. MODULARIZAÇÃO DO DESIGN

A modularização do design é um dos maiores desafios para este trabalho. É necessário trabalhar com a liberdade de edição no ambiente, cobrindo o máximo possível de necessidades, que entre elas são o uso de VSDs, textos, formas de transição entre contextos, barra de navegação, entre várias outras mecânicas que podem fazer a diferença para uma variedade de usuários.

5.1 *Canvas*

Ao optar pela criação de um novo design, o usuário será apresentado com um *canvas* em branco como na figura 5.1, sem objetos posicionados nele. Esses objetos que podem ser posicionados no *canvas*, chamaremos de peças, pois se assemelham bastante a peças de um jogo de quebra-cabeça ou tetris. O *canvas* apresentará apenas áreas livres em forma de *grid* para a adição de peças, e facilitando para o usuário o encaixe destas peças, haverá uma limitação de distância entre elas.



Figura 5.1 – *Canvas*

Para a adição de peças, o usuário deve selecionar a engrenagem localizada no canto inferior direito da tela. Ela abrirá um *pop-up* com alternativas de edição como inserir um VSD, uma coleção de VSDs, textos e barra de navegação. Estes serão os principais tipos de peças, cada uma com suas peculiaridades.

5.2 Peças

Inserir VSD, coleção de VSDs, barra de navegação e texto, serão as principais categorias de peças editáveis. Cada uma terá sua sequência de características selecionadas pelo usuário até o momento de serem adicionadas ao *canvas*.

5.2.1 VSD

Uma das mais básicas peças, o VSD possuirá como características o seu tamanho, sua imagem e a possibilidade de escrever textos em cima ou em baixo da imagem. Neste caso também será disponibilizado a inserção de uma caixa de texto para identificar *hotspots* do VSD. Esta opção poderá ser usada apenas em peças grandes.



Figura 5.2 – Criação de VSD.

Esta peça, ao ter suas características selecionadas, se encaixará no *canvas* conforme o seu tamanho. O usuário terá a liberdade de movê-la pelo *canvas* ao segura-lá com o dedo. Caso seja necessário a edição desta peça, um ícone de engrenagem estará localizado no canto superior direito da mesma. Com um toque, será possível abrir sua edição. Junto a essa peça, caso tenha sido selecionado a adição de *hotspots*, aparecerá uma segunda peça em formato de caixa de texto.

5.2.2 *Hotspots*

Não exatamente uma peça, os *hotspots* são pontos de interesse dentro dos VSDs. Sua função e propósito é ter a possibilidade de indicar com um toque na tela objetos, pessoas, atividades e pontos em destaque que ajudariam na comunicação e interpretação do usuário.

Para a criação dos *hotspots* será necessário um VSD de, no mínimo, tamanho 8x8 (medida dos *grids*). Ao selecionar a adição de um *hotspot*, abrirá uma caixa de texto para identificar o ponto de interesse, expandindo o VSD. Para marcar a localização, é necessário somente selecionar uma área da imagem através do toque e movimento do dedo. Confirmado o processo, o usuário pode adicionar mais pontos ou simplesmente finalizar o VSD.

5.2.3 Coleção de VSDs

A coleção de VSDs não passa de uma série de VSDs adicionados pelo usuário, existindo algumas diferenças características dessa peça. A mesma somente poderá ter tamanhos grandes e textos não poderão ser inseridos em cima ou em baixo. Também será possível selecionar a forma de transição entre as fotos, como possibilitando deslizar para direita, esquerda, baixo, cima ou clicar nelas. Devido à necessidade de uma transição, não será possível colocar peças na direção selecionada.

A inserção de *hotspots* será opcional como na peça anterior. Caso selecionada uma segunda peça será carregada junto a coleção de VSDs. Todos os *hotspots* terão a opção de uso de T2S.

As coleções de VSDs também poderão ser associadas a uma barra de navegação, sendo possível a transição entre categorias de imagens, sem a necessidade de alterar de tela.

5.2.4 Bloco de Texto

Outra peça integral do sistema, a caixa de texto, terá seu tamanho conforme o seu conteúdo e o tamanho da fonte, sendo possível deixá-la maior. Esta peça permitirá a escrita de textos com tamanho de fonte selecionável, cor do texto e fundo. Caso o texto transbordar, o seu tamanho se adequará ao tamanho da peça.

Os *hotspots* presentes nos VSDs e as caixas de texto adicionáveis possuirão as mesmas características desta peça.

T2S estará disponível para o usuário, com um ícone ao lado do texto adicionado, caso esta opção seja selecionada.

5.2.5 Barra de Navegação

A barra de navegação é uma das peças mais importantes do sistema. Com ela, é possível navegar para novas telas ou alterar coleções de VSDs. Será possível criar barras de diversos tamanhos ao serem selecionadas e posicionadas. Será possível habilitar a associação com VSDs já criados ou a serem criados, ou a criação de um novo *canvas*, que terá aquela célula da barra de navegação como acesso.



Figura 5.3 – Barra de Navegação Horizontal.

6. DEFINIÇÕES DO AMBIENTE DE DESENVOLVIMENTO

6.1 Ambiente do Usuário

A aglomeração de objetos nas telas pode demonstrar-se um desafio de adaptação para seus usuários. Portanto foi definido que a aplicação seja desenvolvida para dispositivos com *display* grande, descartando *smartphones*. O uso de uma aplicação para computador não seria prática, impediria o usuário de levar a aplicação com ele para diferentes lugares, caso necessitasse se comunicar com outras pessoas. Já o *tablet* possui um tamanho consideravelmente grande e é prático carregá-lo para diferentes lugares.

O uso do iPad foi constatado por Light [LWT⁺19] como sendo um dos principais dispositivos utilizados em pesquisas e preferidos por seus usuários.

6.2 Flutter

O *framework* desenvolvido pela Google, Flutter, foi o escolhido para o desenvolvimento desta aplicação multimodal sugerida. O Flutter tem como vantagem ser uma plataforma *open source*, disponibilizando para a comunidade uma grande variedade de funcionalidades pré definidas e um grande poder de customização, até mesmo facilitando o acesso a pacotes criados pela comunidade através do site <https://pub.dev>. Onde é possível analisar o código, fazer *download* de maneira simples através de um simples comando de terminal e até mesmo avaliar esses pacotes.

Este *framework* ainda possibilita o desenvolvimento multiplataforma, web, mobile e *desktop* de uma maneira nativa gerando código diretamente para Android ou iOS. [Dev20]

O Flutter procura trabalhar com o conceito de *Widgets*, que foi inspirado no *framework* React segundo a documentação [Tea20a]. A interface de usuário seria portanto populada por *Widgets* facilitando sua construção e configuração na tela de um árvore como na figura 6.1.

Para o desenvolvimento de uma potente interface de usuário o Flutter necessita também de uma linguagem capaz de suprir suas necessidades, para isso a linguagem Dart foi utilizada para sua construção.

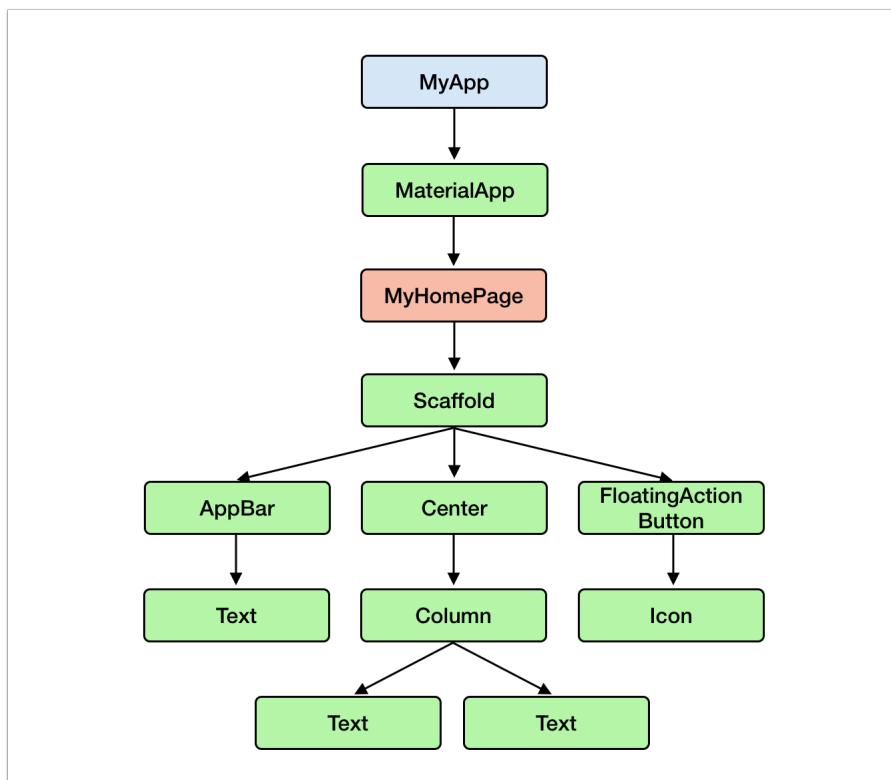


Figura 6.1 – Exemplo de árvore de Widgets retirada de um artigo por Andrea Bizzotto [Biz20]

6.3 Linguagem Dart

Desenvolvida pela Google, Dart foi projetada por Lars Bak e Kasper Lund [aut20], com a intenção de facilitar o desenvolvimento de interfaces web [eGB20].

Em conjunto com Flutter, Dart disponibiliza ferramentas que ajudam o desenvolvedor a ter uma experiência mais imediata, a interface atualiza na tela com uma simples alteração de código, sem necessidade nenhuma de recompila-lo [Biz20].

6.4 Ambiente de Desenvolvimento

O ambiente de desenvolvimento foi definido conforme as facilidades presentes para programar na linguagem Dart e no *framework* Flutter. Foram utilizados o Android Studio e o Visual Studio Code (VS CODE). O primeiro foi utilizado em um computador com um processador Core i7 8700, 16GB de memória RAM, placa de vídeo GeForce RTX 2070 Super, SSD NVME de 128GD com Windows 10, SSD Kingstom 240GBs e dois HDs de um terabyte. Já o VS Code foi utilizado em um Macbook Pro 2014/2 devido ter um impacto menor na performance de uma máquina já com hardware antigo e ainda assim possuído todas as principais utilidades do Android Studio.

Mesmo sendo possível programar para o sistema iOS em uma máquina com Windows 10, não existem opções de emuladores fora da plataforma da Apple, então para compilar o programa é necessário o XCode, um ambiente de desenvolvimento exclusivo para os sistemas MacOS, este fornece a possibilidade de emular qualquer *hardware* da empresa e até mesmo a possibilidade de compilar o programa diretamente em um dispositivo.

Os dispositivos emulados em que o projeto foi testado foram o Nexus 9 rodando a API 29 e o iPAD Pro de terceira geração, outros dispositivos como iPhone 11 e *smartphones* e *tablets* com sistema operacional Android também foram testados, mas tiverem pouca influência em decisões de *layout* como os dois primeiros.

7. DESENVOLVIMENTO

7.1 Ferramentas

O Flutter disponibiliza uma grande biblioteca de ferramentas, para o projeto foram definidas algumas que ajudaram a solucionar problemas, definir ideias e enriquecer as funcionalidades do projeto.

7.1.1 Pub.dev

A ferramenta *pub.dev* disponibilizada pelo Flutter é uma aplicação Web que disponibiliza e organiza várias bibliotecas criadas por outros programadores e até mesmo integrantes do time Flutter para facilitar e agilizar o desenvolvimento de aplicações mobiles.

Este projeto estudou múltiplas bibliotecas que pudessem ser úteis a aplicação, entre elas muitas que facilitavam a criação de interface de usuário com múltiplos *Widgets* na tela, além de funcionalidades simples. As que melhor se encaixaram no projeto foram a *expanded_grid*, uma biblioteca que cria uma interface de usuário *unscrollable*, disponibilizando os *Widgets* em qualquer posição na tela. E a *flutter_tts* disponibilizou usar texto para fala em qualquer parte do projeto, tanto em textos predefinidos quanto em textos escritos pelo usuário.

7.1.2 Flutter tts

T2S ou texto para fala é uma das principais funcionalidades da aplicação, encontramos ela tanto em frases pre-definidas quanto em frases escritas pelo usuário, essa funcionalidade é importante para que os usuários tenham mais uma forma de se comunicarem e serem compreendidos quando o necessário. Portanto para essa funcionalidade foi selecionada a biblioteca *flutter_tts* publicada pelo time *tundralabs.com* no *pub.dev* [Lut20] .

O *flutter_tts* facilita o acesso a interface de texto para fala tanto no sistema Android quanto no sistema iOS, através de uma simples chamada da função *play* ela identifica o sistema do aparelho e ativa a função de leitura do texto passado. Esta biblioteca também proporciona a possibilidade de alterar o volume, alterar o *pitch* da voz, pausar e parar o que está sendo lido, essas funcionalidades não foram implementadas no projeto pois foi definido um volume padrão e como a ideia é usar frases curtas pausar e parar não foram implemen-

tados, no entanto essas funções podem ser adicionadas ao sistema caso necessário no futuro.

Neste trabalho as implementações desta funcionalidade são bem básicas, sempre que for possível usá-la um botão com o ícone *play* encontra-se ao lado ou integrado a texto.

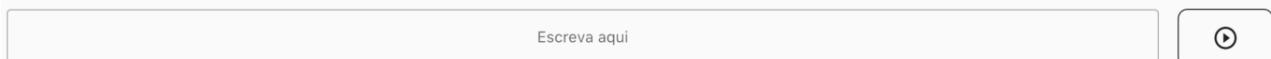


Figura 7.1 – Campo de escrita com T2S

Quando o usuário quiser escrever algo ele aciona a caixa de texto, escreve o que gostaria de falar e aperta no botão *play*, o que foi escrito será lido até o final, ao terminar o usuário pode novamente escrever algo e apertar novamente no botão *play*.

Outra situação que esta funcionalidade é usada, é nos VSDs que possuem uma lista predefinida de textos, muito comum nas interfaces de usuário para adultos, nela existem normalmente uma lista com textos ao lado e dentro de cada item da lista caso haja um ícone *play* o usuário poderá clicar no item e o item será lido para ele como na figura 7.2.

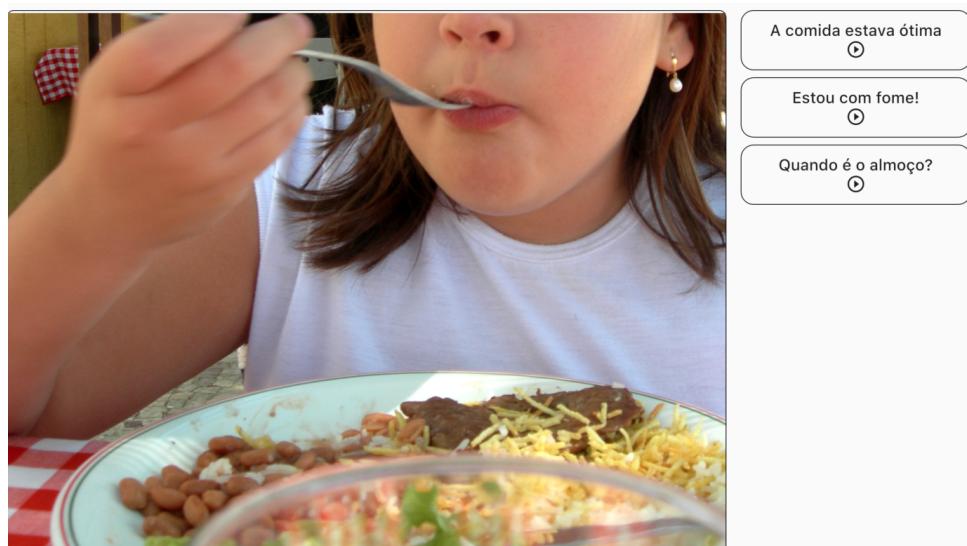


Figura 7.2 – VSD com lista com componentes com T2S

7.1.3 Expanded grid

Dá para se dizer que o *expanded_grid* é o coração da aplicação, de forma simples esta biblioteca tem como propósito criar uma *Grid View unscrollable*. Em flutter as *Grid Views* são por padrão *scrollable*, elas organizam os *Widgets* conforme a ordem que foram inseridos em uma lista, e sempre que ocorre um *overflow* o *scroll* padrão é ativado fazendo com que os *Widgets* apareçam fora da tela e de formar desorganizada. [fas20]

Com o *expanded_grid* é possível estimar um tamanho que torne viável a alocação de *Widgets* na tela conforme a necessidade do *layout*. Ao definir a quantidade de linhas e colunas o *expanded_grid* funciona como uma simples tabela onde é possível posicionar cada *Widget* passando sua posição na linha e coluna e dando um tamanho que representa a quantidade de linhas e colunas que este *Widget* deve ocupar.

Por exemplo, se definirmos uma tabela de dimensões 10 x 10 é possível posicionar um *Widget* na posição linha 1 e coluna 1 que ocupe 6 linhas e 6 colunas como na figura 7.3.

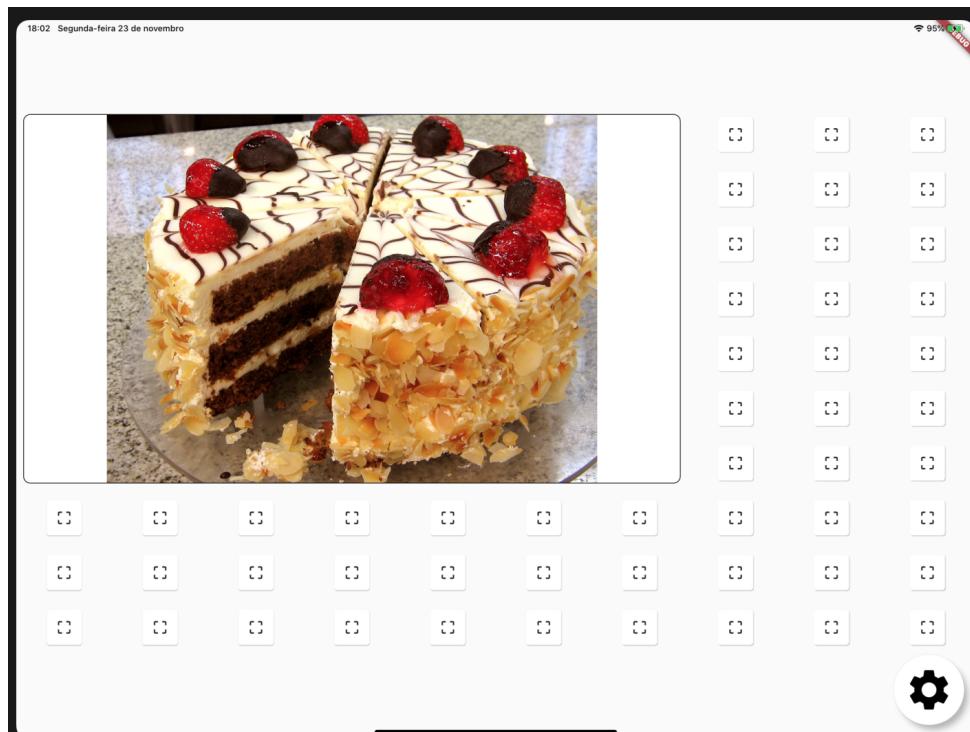


Figura 7.3 – Representação de uma grid.

Caso ocorra um *overflow* a própria biblioteca impossibilita do *Widget* ser posicionado na tabela.

Esta biblioteca com alterações possibilitou a criação de *layouts* para usuários, além de posicionar o *Widget* como na figura 7.3, nesta mesma é possível adicionar outros *Widgets* que caibam nos espaços restantes, como *Cards*, Barra de navegação, listas entre outros.

7.2 Elementos

Os elementos são como peças a serem inseridas em um quebra-cabeça ou *layout* como no caso deste projeto, eles enriquecem a experiência do usuário através de suas

funcionalidades. Alterações foram feitas conforme o desenvolvimento do projeto, mas suas ideias se mantiveram até o final dele.

7.2.1 Canvas

Este elemento é composto por uma tabela que pode receber e posicionar diferentes tamanhos de *Widgets*. Ele se encontra tanto nas telas de edição como nos *layouts* padrões sugeridos. Sua implementação é bem simples, mas um pouco diferente para cada situação.

Na tela de edição o *canvas* é implementado um pouco menor para facilitar a visualização dos objetos a serem inseridos e é inicializado preenchido com ícones que auxiliam no posicionamento dos *Widgets* como na figura 7.4.

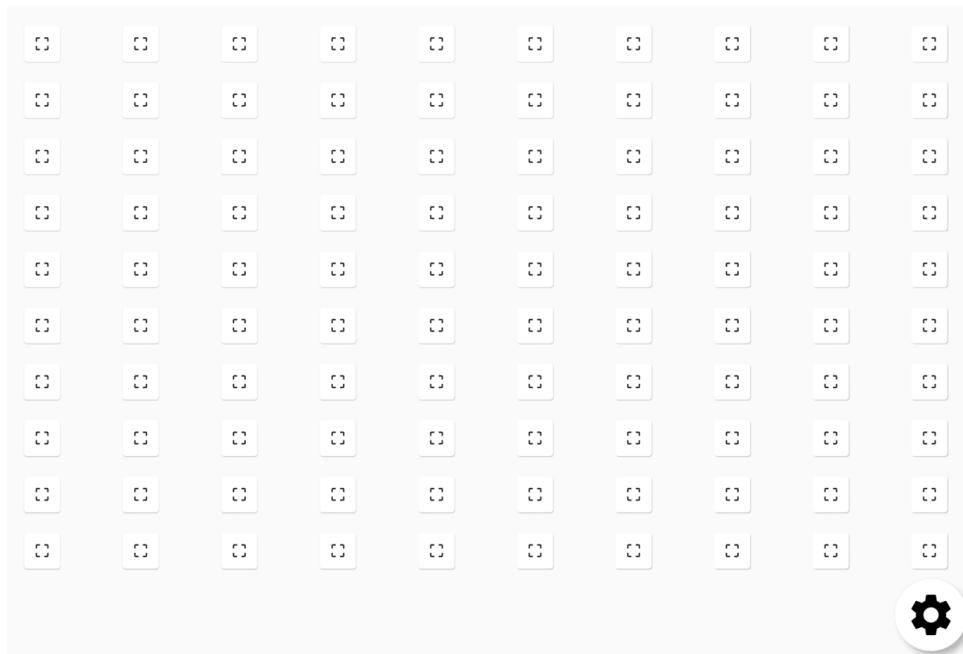


Figura 7.4 – Canvas sem *Widgets* posicionados.

Esses ícones são interativos, quando selecionado o *Widget* a ser adicionado, é só clicar em um deles que o mesmo será inserido na lista de componentes do *layout*.

Já nos *layouts* padrão, o *canvas* já apresenta os *Widgets* posicionados na tela, sem os ícones de referência. Esta versão também preenche a tela inteira pois não tem a necessidade de apresentar a ferramenta de edição no canto inferior direito como na figura 7.4.

7.2.2 Barra de navegação

A barra de navegação é responsável por carregar as informações de VSDs que devem aparecer na tela, ela ocupa uma linha e múltiplas colunas como na figura 7.5, dentro dela se encontram pequenas imagens que representam seu conteúdo a ser demonstrado. Cada barra de navegação pode carregar quantos VSDs quiser, que será ajustada conforme a sua quantidade de elementos.



Figura 7.5 – Em vermelho a barra de navegação.

Todas as informações presentes na barra de navegação são carregadas em um VSD presente no *layout* ou em uma tabela contendo *cards* significativos.

7.2.3 VSDs

Os VSDs são uma combinação de *Widgets*, sendo esses um contêiner obrigatório responsável por disponibilizar uma imagem na tela, uma lista opcional com textos com contextos relacionados a imagem do contêiner com a funcionalidade de T2S e *hotspots* também opcionais. Como tamanho os VSDs possuem o mínimo de ocupação de seis linhas e seis colunas, em alguns *layouts* padrão eles são apresentados como 6x6, 8x10 e 8x8.

Os VSDs precisam estar acompanhados de uma barra de navegação, ela é responsável por carregar as informações de todos os *Widgets*, portanto quando se seleciona

um contexto na barra, o contêiner do VSD é renderizado na tela com todas as informações que está carregando. Opcionalmente caso o VSD possua informações de *hotspots* uma caixa de texto deve estar presente no *layout* para informar a descrição do ponto selecionado pelo usuário. Caso o VSD também possua uma lista os textos estarão acompanhado de um ícone informativo de T2S.

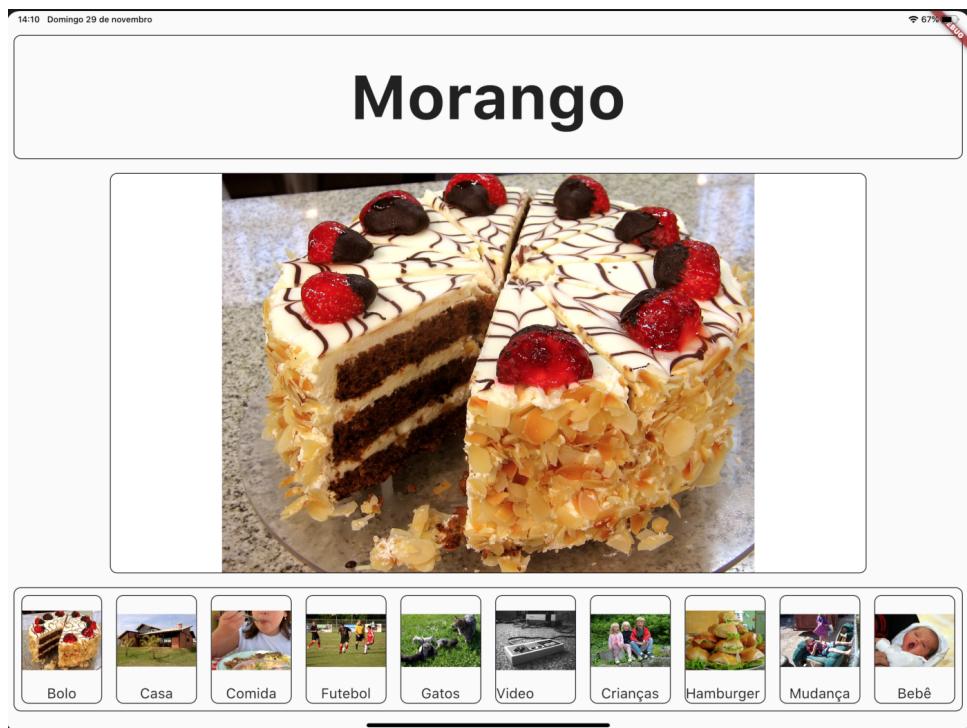


Figura 7.6 – VSD para crianças com deficiência ao selecionar um morango no bolo.

7.2.4 *Cards*

Os *cards* são elementos livres, eles podem fazer parte de uma tabela com contextos significativos ou aparecerem no *layout* individualmente com algum conteúdo predefinido. Tanto em tabela ou individualmente esses *cards* são compostos por uma imagem e sua descrição logo abaixo, de preferência uma única palavra. O seu tamanho também pode variar, como tamanho mínimo ele deve ser composto por uma coluna e duas linhas ou duas colunas e uma linha, a primeira composição é a mais presente na aplicação enquanto a segunda é apenas uma alternativa de tamanho. Outros tamanhos podem ser atribuídos a esses elementos, mas quando *clusterizados* é aconselhável que possuam a mesma altura e largura.

Os *cards* estão presentes na tela padrão para crianças com deficiência em *layout em grid*, esta tela segue um padrão semelhante a outras aplicações que possuem o mesmo propósito. Selecionando um contexto na barra de navegação uma série de *cards* apare-

cerão. Esses *cards* podem ser pressionados que apareceram em uma barra inferior, essa barra é responsável por ler o conteúdo presente nela quando se aperta em *play* como na figura 7.7.

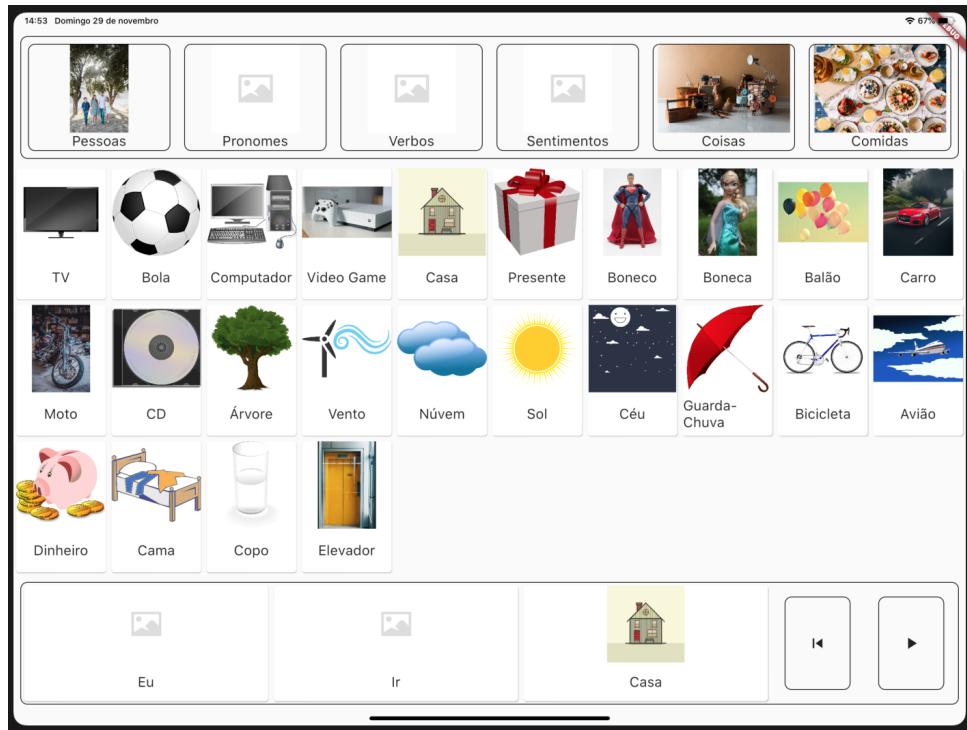


Figura 7.7 – Grid para crianças com deficiência.

7.3 Dificuldades

Desenvolver uma aplicação multimodal se apresentou desafiadora, a escolha do Flutter pareceu ser a mais interessante por suas versatilidades, mas o pouco conhecimento no *framework* e a falta de experiência com ele, dificultaram a tomada de decisões para dar continuidade ao projeto. A criação de uma *grid unscrollable* daria ao projeto uma estrutura padrão ao *layout* que poderia ser utilizada tanto na edição de *layouts* quanto nos padrões definidos por Light. No entanto, a procura por alternativas de criação de *grids unscrollables* foi frustrada pelo grande número de alternativas *scrollables* já implementadas no *framework* do Flutter, entre elas *GridView*, *CustomGridView* e *SliverGrid*, além delas ainda existem outras alternativas que foram desenvolvidas por membros da comunidade do Flutter, como *flutter_staggered_grid_view* e *reorderables* encontrados respectivamente em [Ras20] e [Han20].

As alternativas de terceiros, *flutter_staggered_grid_view* e *reorderables*, demonstraram funcionalidades que poderiam acrescentar bastante ao projeto na procura da modularidade, no entanto o problema de serem *scrollable* persistia, a primeira, *flutter_staggered_grid_view*,

tinha como principal funcionalidade apresentar os *Widgets*, de diferentes tamanhos, "encaixados" uns aos outros como na figura 7.8.

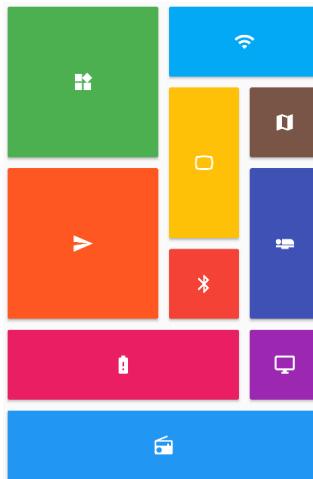


Figura 7.8 – Exemplo do flutter staggered grid view.

A disposição na tela seria perfeita para o projeto, no entanto se demonstrou impossível implementar de forma *unscrollable*, já que os *Widgets* precisavam ser inseridos em uma lista que determinava a posição com base nos espaços não ocupados.

Já o *reorderables* conseguia reorganizar *Widgets* através de *drag and drop*, no entanto esta funcionalidade ficava atrelada ao fato dos *Widgets* possuírem o mesmo tamanho, além dos mesmos problemas que a *flutter_staggered_grid_view* apresentou, como disposição dos *Widgets* na tela e não possuir a possibilidade de ser *unscrollable*.

Usar colunas e linhas, que são *Widgets* padrões disponibilizados pelo Flutter, conseguiriam criar uma *grid unscrollable*, no entanto também foram descartados pela falta de suporte em sua modularidade, os elementos do projeto ficariam presos a posições fixas sem a possibilidade de se "encaixarem" a outros como na figura 7.8, apresentando espaços em branco. O uso de uma tabela também foi estudado e experimentado, porém esbarrava nos mesmos problemas das colunas e linhas.

Tentando agilizar o projeto, se passou para o desenvolvimento dos elementos que iriam compor os *layouts*. Esses elementos herdam um *Widget* básico, que possuíam informações como "id", identidade do elemento, "mini", imagem que representa aquele elemento e "descricao", descrição do elemento. Com essas informações foi possível gerar todos os elementos presentes no projeto, através do "id" é possível dizer quais informações devem aparecer na tela, através do "mini" e da "descricao" é possível compreender o contexto que é carregado na barra de navegação. Portanto popular as telas com *layout* padrão se tornou simples pois era só alocar os elementos em suas devidas posições.

Essa abordagem, no entanto torna o projeto "fixo", impossibilitando futuras edições pelo usuário.

7.3.1 Tela de Edição

Infelizmente o que seria a principal ideia do projeto não pôde ser implementada de forma ideal, o atraso na definição das ferramentas a serem utilizadas, os problemas com os diferentes tipos de *Widgets* existentes na ferramenta e a falta de conhecimento no desenvolvimento em Flutter foram responsáveis por possibilitar apenas um protótipo onde se pode encaixar alguns *Widgets* na edição. A implementação dos menus para configurar as informações dos elementos não está presente, apenas é possível selecionar alguns deles e colocar na tela nas posições definidas na tabela.

7.4 Soluções

Mesmo com os *layouts* padrão já montados, foi encontrada uma solução que facilitaria montar a interface de usuário através de edição ou até mesmo remontar os já existentes. Foi então adicionado ao projeto a ferramenta *expanded_grid*. Ela dá a oportunidade de montar *layouts unscrollables* semelhantes ao da figura 7.8, passando alguns parâmetros responsáveis por definir sua posição na tela.

Diferente das outras alternativas de *GridView*, o *expanded_grid* não se utiliza das implementações de Grid predefinidas pelo Flutter, sua abordagem envolve mapear o posicionamento dos *Widgets* com base nos seus tamanhos e posições, desenhando-os em uma *Stack*.

Em Flutter uma *Stack* é um *Widget* que é responsável por empilhar todos os *Widgets* que fazem parte do seu escopo como na figura 7.9. No entanto, seus *Widgets* podem ser envoltos pelo *Widget Positioned*, que tem como função dar uma posição fixa ao *Widget* dentro da *Stack*. [Tea20b]



Figura 7.9 – Exemplo de *Widgets* em uma *Stack*

Definido como posicionar os *Widgets*, essa ferramenta monta uma *Grid* pegando a altura e largura do dispositivo e dividindo pela quantidade de linhas e colunas solicitadas pelo usuário. Com o tamanho da *Grid* definido, falta agora inserir os *Widgets*. Semelhante as alternativas experimentadas anteriormente, *flutter_staggered_grid_view* e *reorderables*, esta *Grid* recebe uma lista de *Widgets*, no entanto estes precisam carregar sua posição e seu tamanho, para que possam ser inseridos e visualizados na tela como na figura 7.10.

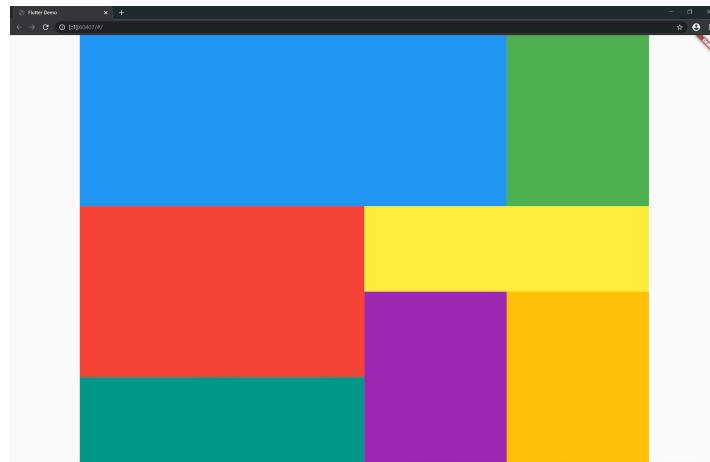


Figura 7.10 – Exemplo retirado da documentação do *expanded_grid*

Foi necessário portanto alterar a implementação de todos os elementos já criados no projeto, definir tamanhos, posições, alterar os *layouts* já existentes para ficarem condizentes com a ferramenta, causando muito retrabalho e tempo desperdiçado.

7.4.1 Refatorações

Durante o projeto foram feitas várias refatorações, resultado de uma falta de planejamento e conhecimento em Flutter para a implementação do projeto. Essas refatorações se mostraram eficientes conforme a descoberta de novidades e tomadas de decisões importantes, como retirar das *Widgets* responsabilidades que deveriam ser de modelos ou telas.

A refatoração seria a principal ferramenta para melhorar o projeto e até mesmo implementar a tão importante tela de edição. O primeiro passo seria criar modelos para cada *Widget*, ao invés deles herdarem um *Widget* básico, os mesmos teriam modelos de mesmo nome que herdariam um modelo básico, facilitando a implementação de funcionalidades e retirando das *Widgets* inúmeras informações desnecessárias.

8. CONCLUSÃO

Atingir o maior número de pessoas com necessidades complexas de comunicação é extremamente complicado. Aplicações existentes de AAC são desenvolvidas para grupos específicos de pessoas e muitas vezes essas aplicações estão cobertas por uma barreira monetária. O desenvolvimento da aplicação modular e de graça, tem como premissa dar a liberdade ao usuário em construir um ambiente do qual se sinta confortável e familiar.

Com o design que procura disponibilizar uma grande variedade de alternativas de **layout** através de elementos (VSDs, barra de navegação, T2S), sugeridos por Light [LWT⁺19], Drager [DLC⁺04], McKelvey [MDH⁺07] e O'Neill [OWL19] em estudos com pessoas com diferentes tipos de necessidades complexas de comunicação. Este trabalho procura disponibilizar uma ferramenta que possa ser usada em um ambiente acadêmico e no quotidiano de pessoas com CCN.

O desenvolvimento deste projeto demonstrou o quanto complicado é chegar a um produto de qualidade nesta área, talvez com melhor conhecimento nas ferramentas utilizadas seria possível ter apresentado algo mais consistente com as ideias sugeridas pelos pesquisados.

Ainda assim é possível chegar a bom resultados com melhor planejamento e mais tempo, como trabalhos futuros fazer uma refatoração completa no código, separar componentes e modelos, estudar um pouco mais o Flutter, não tomar decisões precipitadas e principalmente procurar avaliar e testar a aplicação justamente com as pessoas para as quais ela foi planejada, procurar ter um *feedback* para validar o projeto e talvez disponibilizar nas plataformas sugeridas.

REFERÊNCIAS BIBLIOGRÁFICAS

- [Ass19a] AssistiveWare. “Intuitive, user-focused products”. Capturado em: <https://www.assistiveware.com/products>, set 2019.
- [Ass19b] AssistiveWare. “Proloquo2go”. Capturado em: <https://apps.apple.com/us/app/proloquo2go/id308368164#?platform=ipad>, set 2019.
- [aut20] autores, M. “Dart (programming language)”. Capturado em: [https://en.wikipedia.org/wiki/Dart_\(programming_language\)](https://en.wikipedia.org/wiki/Dart_(programming_language)), nov 2020.
- [Biz20] Bizzotto, A. “What’s great about flutter?” Capturado em: <https://codewithandrea.com/videos/2019-12-16-whats-great-about-flutter/>, nov 2020.
- [CBH17] Carniel, A.; Berkenbrock, C.; Hounsell, M. “Um mapeamento sistemático sobre o uso da comunicação aumentativa alternativa apoiada por recursos tecnológicos”, *Revista Brasileira de Computação Aplicada*, vol. 9–2, jul 2017, pp. 84–98.
- [Dev20] DevMedia. “Guia de flutter”. Capturado em: <https://www.devmedia.com.br/guia/flutter/40713>, nov 2020.
- [DLC⁺04] Drager, K. D. R.; Light, J. C.; Carlson, R.; D’Silva, K.; Larsson, B.; Pitkin, L.; Stopper, G. “Learning of dynamic display aac technologies by typically developing 3-year-olds”, *Journal of Speech, Language, and Hearing Research*, vol. 47–5, 2004, pp. 1133–1148.
- [eGB20] e Gilad Bracha, L. B. “Dart”. Capturado em: http://gotocon.com/dl/goto-aarhus-2011/slides/GiladBracha_and_LarsBak_OpeningKeynoteDartANewProgrammingLanguageForStructuredWebProgramming.pdf, nov 2020.
- [fas20] fastriver_org. “expanded_grid”. Capturado em: https://github.com/organic-nailer/expanded_grid, nov 2020.
- [Fle19a] Flex, T. S. “Sono flex”. Capturado em: <https://apps.apple.com/us/app/sono-flex/id463697022#?platform=ipad>, set 2019.
- [Fle19b] Flex, T. S. “Tobi sono flex - the symbol vocabular app: flexible, structured, growth-oriented”. Capturado em: <http://tobiisonoflex.com/>, set 2019.
- [Han20] Hansheng. “reorderables”. Capturado em: <https://github.com/hanshengchiu/reorderables>, nov 2020.
- [Lut20] Lutton, D. “flutter_tts”. Capturado em: https://github.com/dlutton/flutter_tts, nov 2020.

- [LWT⁺19] Light, J.; Wilkinson, K. M.; Thiessen, A.; Beukelman, D. R.; Fager, S. K. “Designing effective aac displays for individuals with developmental or acquired disabilities: State of the science and future research directions”, *Augmentative and Alternative Communication*, vol. 35–1, jan 2019, pp. 42–55.
- [MDH⁺07] McKelvey, M.; Dietz, A.; Hux, K.; Weissling, K.; Beukelman, D. “Performance of a person with chronic aphasia using personal and contextual pictures in a visual scene display prototype”, *Journal of Medical Speech-Language Pathology*, vol. 15–3, set 2007, pp. 305–317.
- [nHEL19a] 2nd Half Enterprises LLC. “Mytalktools”. Capturado em: <http://www.mytalktools.com/dnn/2/Products.aspx>, set 2019.
- [nHEL19b] 2nd Half Enterprises LLC. “Mytalktools mobile”. Capturado em: <https://apps.apple.com/us/app/mytalk-mobile/id324286288#?platform=ipad>, set 2019.
- [OWL19] O’Neill, T.; Wilkinson, K. M.; Light, J. “Preliminary investigation of visual attention to complex aac visual scene displays in individuals with and without developmental disabilities”, *Augmentative and Alternative Communication*, vol. 35–3, 2019, pp. 240–250, pMID: 31305160.
- [Ras20] Rastel, R. “flutter_staggered_grid_view”. Capturado em: https://github.com/letsar/flutter_staggered_grid_view, nov 2020.
- [Tea20a] Team, F. “Introduction to widgets”. Capturado em: <https://flutter.dev/docs/development/ui/widgets-intro>, nov 2020.
- [Tea20b] Team, F. “Stack class”. Capturado em: <https://api.flutter.dev/flutter/widgets/Stack-class.html>, nov 2020.