

TAGARELA BRAILLE – APP PARA AUXÍLIO NO APRENDIZADO AO BRAILLE

Leonardo Pereira Vieira, Prof. Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

leonardopereira@furb.br, daltonreis@furb.br

Resumo: Este artigo apresenta o desenvolvimento do aplicativo para auxiliar no aprendizado do braille inicialmente criado por Cazagrande (2016). Um dos objetivos propostos por esse artigo é tornar o aplicativo mais acessível através de áudio e interação por gestos na tela, possibilitando assim a utilização por pessoas cegas. Este aplicativo pretende também tornar mais fácil a inclusão de novos módulos para que possa ser criado mais ferramentas dentro do aplicativo e abranger outras áreas da educação especial. Para isso foi feito a migração da tecnologia utilizada inicialmente para o Flutter que utiliza bem o conceito de componentização para reaproveitar os componentes. Foi percebido a necessidade de otimizar as instruções do manual de uso, porém a utilização do aplicativo se mostrou eficiente em acessibilidade, pois cumpriu de forma prática o objetivo de auxiliar e estimular o aprendizado de pessoas com ausência parcial ou total da visão.

Palavras-chave: Acessibilidade. Educação Especial. Braille. Componentização. Flutter.

1 INTRODUÇÃO

Segundo pesquisa do Instituto Brasileiro de Geografia e Estatística (2017) de acordo com o censo de 2010, existe no Brasil quase 46 milhões de pessoas com deficiência, cerca de 24% da população, se declaram deficientes de uma das habilidades investigadas (enxergar, ouvir, caminhar ou subir degraus), ou possuir deficiência mental / intelectual (INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA, 2017). Dentre esses já passaram ou ainda irão passar pelo período escolar, como é previsto pelo artigo 2º da lei nº 7.853 de 1989, que garante esse direito:

Ao Poder Público e seus órgãos cabe assegurar às pessoas portadoras de deficiência o pleno exercício de seus direitos básicos, inclusive dos direitos à educação, à saúde, ao trabalho, ao lazer, à previdência social, ao amparo à infância e à maternidade, e de outros que, decorrentes da Constituição e das leis, propiciem seu bem-estar pessoal, social e econômico. (BRASIL, 1989, p.1).

Conforme Masini e Gaspareto (2007), esse tipo de inclusão de alunos com deficiência tem se tornado alvo de muita discussão por profissionais da educação. O tema é polêmico por isso tem causado grande impacto e discussão entre professores, diretores e todos os profissionais que atuam no meio escolar (MASINI; GASPARETO 2007, p.35). Uma dessas deficiências que os portadores muitas vezes sofrem é com a falta de preparo e/ou adaptação do ambiente escolar e dos profissionais de educação é a deficiência visual. Segundo a organização Marta Gil (2000), cegueira pode ser tanto adquirida como congênita. Quando o indivíduo nasce com o sentido da visão, guarda memórias visuais, facilitando na sua adaptação. Quem nasce sem a capacidade da visão, por outro lado, jamais pode formar uma memória visual, possuir lembrança visual (MARTA GIL, 2000, p.8).

Atualmente existem diversos softwares para o auxílio a pessoas com deficiência como o DOSVOX, BlindSquare, entre outros que são usados pelos próprios deficientes. Existem outros que auxiliam também pessoas com deficiência como o Liane TTS que ensina linguagem de sinais. Mas esses softwares são encontrados em locais diferentes, fazendo com que o usuário tenha que baixar de diferentes fornecedores, criando diferentes contas e o progresso gravado em locais diferentes. Uma alternativa para não se ter vários softwares e reunir estas funcionalidades em um único aplicativo, seria utilizando o conceito de componentização para facilitar a inclusão de novos módulos. O conceito de componentização pode ser usado para criar um aplicativo com componentes que possam ser reaproveitados e facilitando a inclusão de novos módulos, tornando assim um aplicativo mais completo.

Outra possível vantagem na utilização do conceito de componentização está relacionado ao desenvolvimento de *softwares* que está cada vez mais complexo e detalhado. Com isso são necessários tempos mais longos para o desenvolvimento e muitas vezes é ultrapassado o tempo estimado, resultando em atrasos e reclamações por parte dos clientes. Para resolver esse problema, alguns desenvolvedores utilizam o conceito de componentização, reutilizando componentes e padronização das regras de negócio, dando mais agilidade no desenvolvimento. O desenvolvimento baseado em componentes proporciona uma alta produtividade e flexibilidade. Com esta abordagem, componentes podem ser empregados em qualquer sistema que precise de uma ou mais funcionalidades providas por eles. (OLIVEIRA, 2012, p. 40).

Diante dos argumentos citados, este trabalho desenvolveu um aplicativo para o auxílio no ensino de pessoas com e sem deficiência visual, possibilitando uma melhor adaptação entre os deficientes visuais e os que convivem com eles. Acreditasse assim que esse projeto possa vir auxiliar a integração dos deficientes ao meio onde vivem, facilitando o aprendizado e adaptação, principalmente no meio educacional. Para isso será utilizado padronização de projetos utilizando conceito de componentização para facilitar a criação de novos módulos.

2 FUNDAMENTAÇÃO TEÓRICA

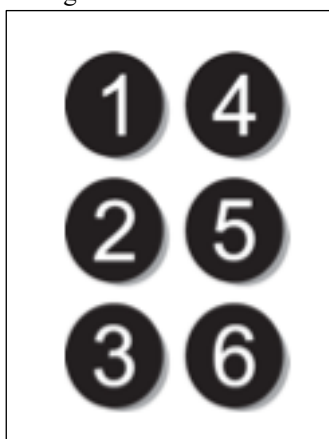
Nesse capítulo será apresentado a fundamentação teórica, onde o item 2.1 **Erro! Fonte de referência não encontrada.** apresentará um pouco sobre o braille, e no 2.2 mostrará um pouco sobre o conceito de componentização. O item 2.3 será apresentado o protótipo atual desenvolvido por Lucas Cazagrande em 2016 e no item 2.4 os trabalhos correlatos com o trabalho relatado nesse artigo.

2.1 BRAILLE

O braille é um sistema de escrita e leitura tátil para as pessoas cegas e foi criado por Louis Braille que desenvolveu um sistema de leitura para deficientes visuais (INSTITUTO BENJAMIN CONSTANT, 2018) e se tornou importante para portadores de deficiência visual. O sistema braille baseados em símbolos de alto-relevo resultantes da combinação de até seis pontos, no que se convencionou chamar de “cela braille”, dispostos em duas colunas de três pontos cada (COSTA, 2009). E através desses pontos são representados 63 símbolos (OTSUKA, 2010).

Segundo o ministério da educação (2018) os pontos são numerados de cima para baixo e da esquerda para a direita (Figura 1), os 63 sinais simples do Sistema Braille, são apresentados numa sequência denominada ordem braille, e são distribuídos por sete séries. Todas as letras do alfabeto são utilizadas apenas uma cela braille, mas existe alguns símbolos que se faz necessário a utilização de mais de uma cela braille como por exemplo para representação de números que são utilizados os pontos 3, 4, 5 e 6 seguidos dos mesmos pontos que representam as letras “a” a “j” e representam os números de um a zero. E para representar letras em maiúsculo é utilizado o símbolo precedido dos pontos 4 e 6, e quando as letras são todas em maiúsculas são utilizadas duas celas com os pontos 4 e 6 antes da palavra.

Figura 1 - Pontos braille



Fonte: Brasil (2018).

O braille foi introduzido no Brasil por José Álvares de Azevedo, que havia estudado em Paris e era cego. Em 1854 graças a ele foi fundada no Rio de Janeiro, a primeira escola para pessoas com deficiência visual da América Latina (OLIVEIRA, 2016). O Brasil utilizou o braille na sua forma original até a década de 1940, quando teve que alterar por conta da reforma Ortográfica da Língua Portuguesa, e ficou sem regra específica até que em 1999 foi criado a Comissão Brasileira do Braille. Em 2002 foi criado a Grafia braille para a Língua Portuguesa e passou a ser utilizado pelos territórios brasileiros e portugueses (INSTITUTO BENJAMIN CONSTANT, 2018).

Existem diversos instrumentos para a escrita braille como o reglete (Figura 2), que é acompanhado do punção. A escrita com o reglete é feita da direita para a esquerda, porque as palavras que são lidas no alto-relevo que é formado no papel (CIVIAN, 2017). O reglete ainda é usado apesar de que hoje em dia existem algumas tecnologias com máquina de escrever em braille, da impressora e de softwares leitores de tela (CIVIAN, 2017).

Figura 2 - Reglete



Fonte: Civian (2017).

Hoje em dia são utilizados *displays* braille que já vêm sendo utilizados em muitos países e resolvem as dificuldades de armazenamento permitindo que pessoas cegas possam ler com mais autonomia e em qualquer lugar. Contudo, o alto custo desses equipamentos ainda os torna inacessíveis para a maioria dos cidadãos (OLIVEIRA, 2016).

2.2 COMPONENTIZAÇÃO

Um conceito que foi usado nesse trabalho e que também é usado na indústria de jogos é a de reutilização de componentes, conceituado como a produção de componentes de software que são úteis para mais de um projeto (ROLLINGS; MORRIS, 1999). Segundo Oliveira (2012), componentização de software é uma abordagem arquitetural baseada na divisão de sistemas de software em unidades menores, denominadas componentes, permitindo a criação de um sistema como se houvesse vários sistemas menores, diminuindo sua complexidade, fazendo com que cada componente seja focado em um conjunto de funcionalidades semelhantes ou em uma função específica, nas quais estas funções podem ser reutilizadas através do acesso ao componente (OLIVEIRA, 2012). Existem vários benefícios com uma aplicação baseada em componentes como cita Oliveira (2012).

O desenvolvimento de aplicações baseado em componentes nos traz uma série de benefícios, dentre os quais podemos destacar:

- Produtividade: Pode-se economizar tempo de desenvolvimento, dependendo do portfólio de componentes já prontos;
- Robustez: Maior qualidade no produto final que utiliza componentes, pois os mesmos já foram largamente testados em um projeto dedicado à construção dos mesmos;
- Padrão de desenvolvimento: Equipe orientada a desenvolvimento nos moldes da componentização (OLIVEIRA, 2012, p. 40).

2.3 PROJETO ATUAL

O projeto atual foi desenvolvido por Cazagrande (2016), e tinha como objetivo desenvolver um módulo de jogos para o tagarela que é um aplicativo de comunicação alternativa, para auxiliar no aprendizado do sistema braille por pessoas normovisuais. Para desenvolver o projeto foi utilizado Brackets em conjunto com o PhoneGap e utilizado as linguagens HyperText Markup Language 5 (HTML5), Cascading Style Sheets (CSS) e JavaScript para o desenvolvimento de três opções, sendo acessada a partir da tela principal como mostra a Figura 3.

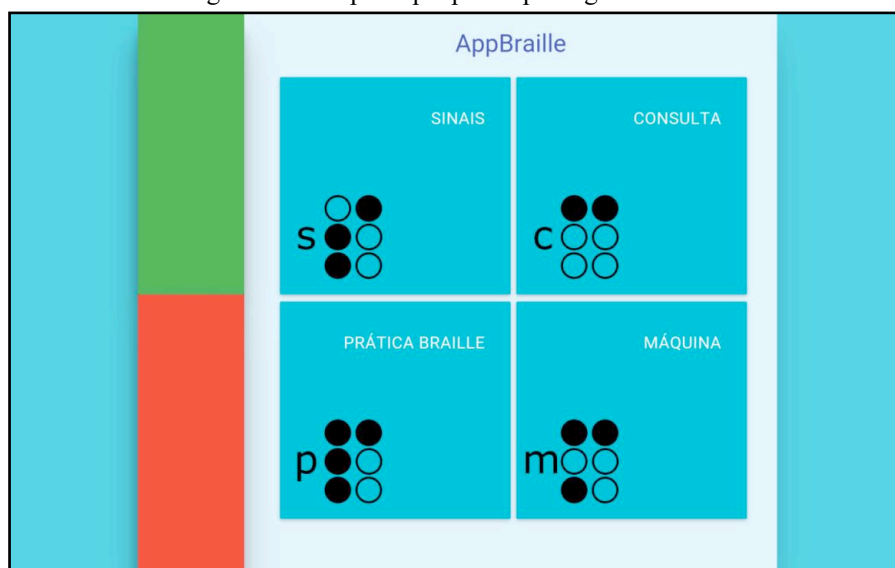
A primeira opção *Sinais* foi criada para demonstrar as sete séries de sinais braille, que representam todas as letras e alguns sinais escritos. Para facilitar o entendimento da disposição das séries, quando o usuário clica em alguma das celas é direcionado para um exemplo com a letra ou sinal selecionado.

A opção *consulta* representa a transição dos pontos braille. Nessa tela tem duas celas dispostas uma ao lado da outra já que um único símbolo escrito pode ser representado por mais de uma cela braille, como por exemplos os números.

Na terceira opção *Prática Braille* são apresentadas imagens com a descrição que remete ao significado da imagem, uma cela braille ao lado e abaixo do nome da imagem, e uma descrição com o uso dos sinais braille. Alguns dos

sinais são substituídos por um sinal de interrogação em tinta, assim o usuário tem que informar qual o sinal braille é referente ao sinal que está faltando.

Figura 3 - Tela principal protótipo Tagarela Braille



Fonte: Cazagranda (2016).

A quarta opção *Máquina* não foi implementada ficando como sugestão para extensões, que seria um simulador de máquina de escrever em braille. Um dos principais pontos fortes desse trabalho foi a divisão dos módulos que deixa bem intuitivo para o usuário e os exemplos com imagens, representando melhor e ajudando a fixar o exemplo que se quer passar. O ponto negativo foi a falta de acessibilidade, através de áudio, gestos na tela ou movimentos do dispositivo, para tornar o aplicativo mais acessível por quem é cego ou tem baixa visão.

2.4 TRABALHOS CORRELATOS

Foram escolhidos três trabalhos correlatos com propostas semelhantes a este. No Quadro 1 descreve o aplicativo *Aprende Braille* desenvolvido por Ugedo (2016), com exercícios para praticar e aprender braille. No Quadro 2 relata o trabalho *AbcNum Braille* (AQUINO *et al.*, 2015), que tem como objetivo desenvolver uma aplicação para auxiliar na aprendizagem do alfabeto braille para pessoas com baixa visão. Por último, no Quadro 3 o *LêBraille* (FAÇANHA *et al.*, 2012) um aplicativo que simula os instrumentos utilizados para o aprendizado em braille.

Quadro 1 - Trabalho correlato *Aprende Braille*

Referência	Ugedo (2016)
Objetivos	O aplicativo visa auxiliar no aprendizado de braille através de exercícios.
Principais funcionalidades	a) alfabeto – lista com o alfabeto e seus respectivos sinais em braille; b) ejercicios de letras – ejercicios para o usuário digitar a letra que é representada na célula; c) ejercicios de palabras – ejercicios para o usuário tem que digitar a palavra que é representado nas células braille; d) ejercicios de palabras – módulo de ejercicios para o usuário digitar o número informado na célula braille; e) ejercicios de frases – Assim como o módulo exercício de palavras, os ejercicios com frases para o usuário informar qual frase representa os símbolos em braille.
Ferramentas de desenvolvimento	Não foi encontrada a ferramenta usada para o desenvolvimento.
Resultados e conclusões	O que também auxilia na acessibilidade é o <i>feedback</i> com áudio presente em algumas opções do aplicativo. O aplicativo é prático e intuitivo, com nota de 4,6 na loja de aplicativo e mais de dez mil downloads na <i>play story</i> .

Fonte: elaborado pelo autor.

Quadro 2 - Trabalho correlato *AbcNumBraille*

Referência	Aquino <i>et al.</i> (2015)
Objetivos	Auxiliar deficientes visuais no processo de alfabetização, possibilitando a prática de atividades de alfabetização.
Principais funcionalidades	a) feedback sonoro; b) feedback tátil;

	c) exercício de vogais; d) exercício de consoantes; e) exercícios de números.
Ferramentas de desenvolvimento	Não foi informado as ferramentas usadas no desenvolvimento.
Resultados e conclusões	Os resultados não foram divulgados, mas pode-se concluir que a proposta auxilia o aprendizado do braille por pessoas com baixa visão porque possui retorno tátil e sonoro.

Fonte: elaborado pelo autor.

Quadro 3 – Trabalho correlato LêBraille

Referência	Façaanha <i>et al.</i> (2012)
Objetivos	O principal objetivo de disponibilizar uma ferramenta de auxílio na alfabetização no sistema braille.
Principais funcionalidades	a) o aplicativo dispõe de uma tela com 6 teclas que representam uma cela braille; b) interação por gestos na tela; c) feedback da interação por áudio.
Ferramentas de desenvolvimento	Linguagem e programação Java.
Resultados e conclusões	O aplicativo se mostrou acessível principalmente para pessoas que possuem um grau maior de deficiência visual, como mostrado nos resultados obtidos após a experiência com deficientes visuais (FAÇANHA <i>et al.</i> , 2012). Os pontos mais marcantes foram a navegabilidade e o conteúdo, como relatados pelos que participaram da pesquisa apesar de alguns não demonstrarem segurança no uso do sistema (FAÇANHA <i>et al.</i> , 2012).

Fonte: elaborado pelo autor.

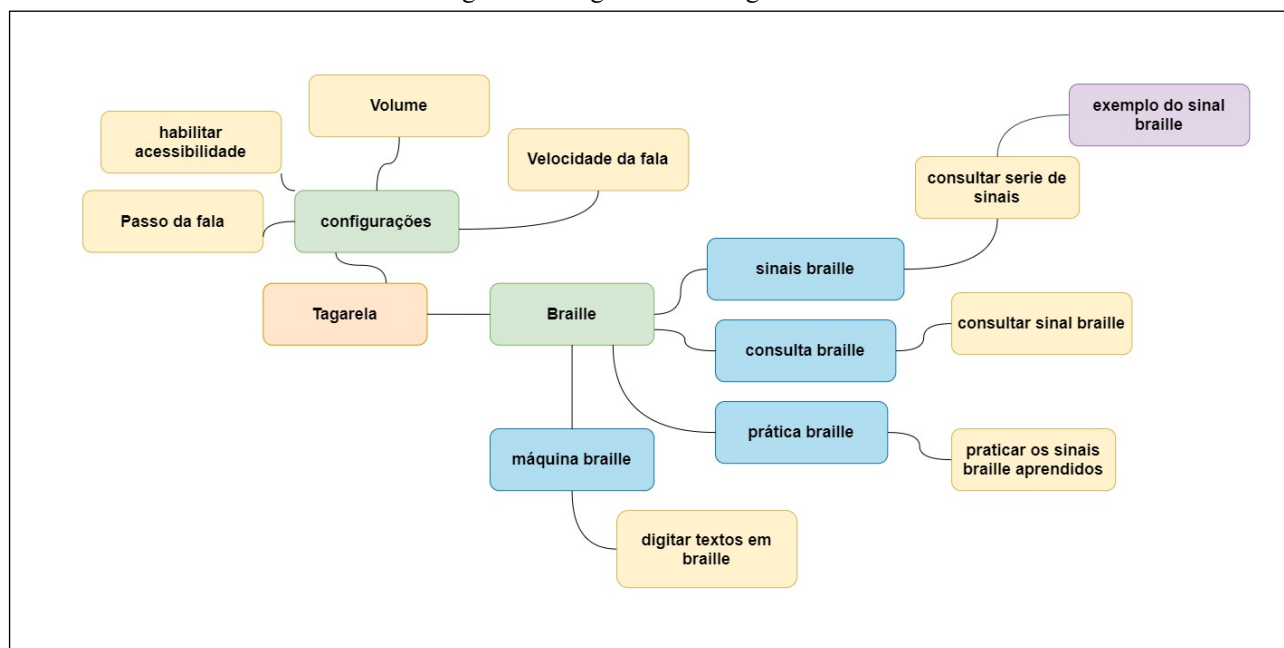
3 DESCRIÇÃO

Neste capítulo será descrito detalhes sobre a implementação do aplicativo, na seção 3.1 serão descritos os aspectos mais importantes e uma visão geral do aplicativo. Na seção 3.2 tem-se um detalhamento sobre a implementação do aplicativo e ferramentas utilizadas.

3.1 VISÃO GERAL DO APLICATIVO

O aplicativo dispõe de uma tela inicial com a opção *Braille* que dispõe das funcionalidades principais para auxiliar no aprendizado do braille (Figura 4). Atualmente só se tem o módulo *Braille*, mas outros módulos serão desenvolvidos, e o módulo *Braille* servirá como modelo para os próximos módulos. Ao iniciar o aplicativo um áudio informa que, se o usuário deseja acessar o módulo braille deve-se clicar duas vezes na tela. Ao entrar no módulo tem-se a tela inicial do *Tagarella Braille* que será explicada melhor a seguir. Para um melhor entendimento, a especificação do desenvolvimento deste trabalho será dividida em duas partes, a acessibilidade e as funcionalidades do módulo braille.

Figura 4 - Diagrama de navegabilidade



Fonte: elaborado pelo autor.

3.1.1 Modo acessibilidade

A acessibilidade pode ser configurado no menu lateral do aplicativo na opção configurações. Por padrão, essa configuração é ativada para que pessoas com baixa visão ou cegas possam utilizar o aplicativo sem ajuda externa. No menu configurações existem alguns ajustes que podem ser feitos, são eles:

- a) **acessibilidade:** habilita a função de acessibilidade para uso por pessoas com baixa visão ou cegas;
- b) **volume:** essa funcionalidade, assim como as demais configurações, só é usada quando o modo acessibilidade está ativo. Essa configuração serve para controlar o volume da voz do modo acessibilidade;
- c) **velocidade:** essa configuração é a velocidade com que o áudio é falado;
- d) **passo:** é a velocidade do passo das palavras que é falada pelo aplicativo.

A função de acessibilidade foi desenvolvida pensando em usuário que tem pouca visão ou são cegos, pois o feedback de áudio auxilia e possibilita que ele possa utilizar o aplicativo através de gestos na tela. Com essa opção ativa o usuário tem total controle no módulo Braille, com exceção das configurações. Para que o usuário não clique em uma opção por engano todos os botões da tela são desabilitados quando a acessibilidade estiver ativa e a interação com o aplicativo só funcionará através gestos na tela. Sempre que o usuário entrar na tela e o modo acessibilidade estiver ativo é dada as instruções por áudio para que ele possa utilizar as funcionalidades dela. Os gestos detectados pelo aplicativo seguem sempre um padrão e são acompanhados de um áudio da ação tomada. Como pode ser visto no Quadro 4 que lista os gestos e ações disponíveis no aplicativo.

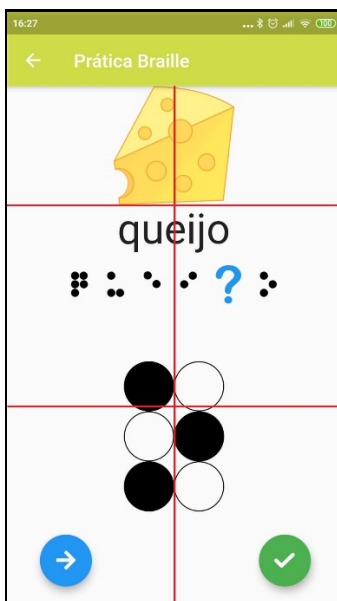
Quadro 4 - Gestos e ações

Gestos	Ações
Manter pressionado	Repete a instrução dada ao entrar na tela.
Deslizar para direita	Navega entre as opções de botões ou opções da tela, sempre seguido de dois cliques na tela para confirma a opção selecionada.
Deslizar para esquerda	Assim como o deslizar para a direita este gesto navega entre as opções, sendo que no sentido contrário e assim navegando para a opção anterior, caso tenha mais de uma opção.
Um toque na tela	Depende da tela, conforme explicado a seguir.
Dois toques na tela	Quando navegado pelas opções disponíveis e ao clicar duas vezes executa a ação.

Fonte: elaborado pelo autor.

Em algumas telas como a da opção prática é necessário informar qual ponto braille deve estar marcado. O que pode ser feito pressionando uma das regiões da tela, como mostra na Figura 5.

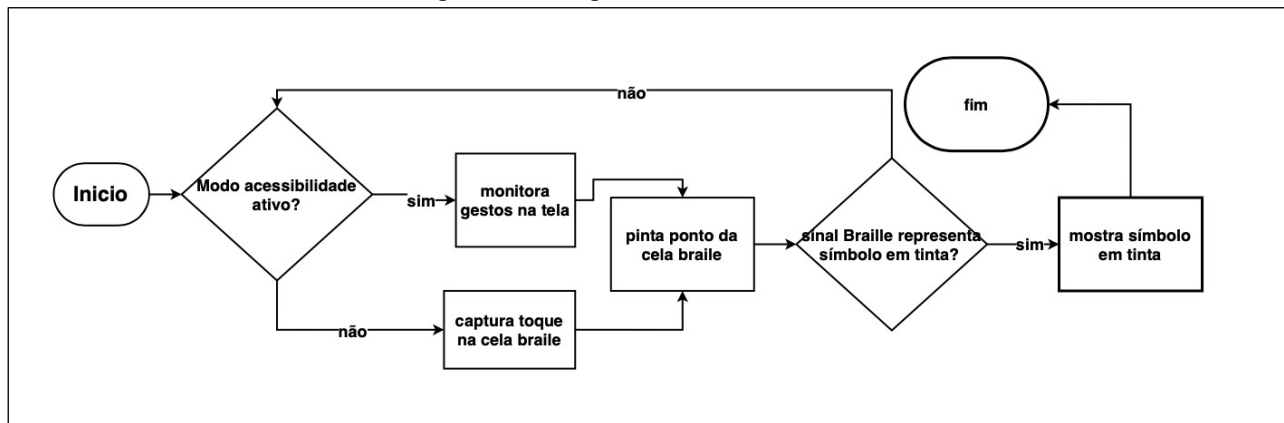
Figura 5 - áreas da célula módulo consulta



Fonte: elaborado pelo autor.

Sempre que o usuário clica em alguma das áreas divididas na tela, o ponto braille é marcado de acordo com o local da tela clicado como é representado pelo fluxograma na Figura 6. Sempre com o retorno de áudio para que o usuário saiba que sinal está sendo digitado. Nessa tela ainda o usuário pode navegar entre as duas opções: os botões pular palavra e confirmar. Para navegar entre essas opções ele tem que deslizar para direita ou esquerda e confirmar com dois cliques na tela. Assim como a tela de prática a de consulta também contém celas braille. A diferença é que ao marcar o ponto a representação do caractere em tinta é mostrado na tela. A Figura 6 demonstra o fluxograma da consulta de sinais braille.

Figura 6 - Fluxograma consulta sinal braille

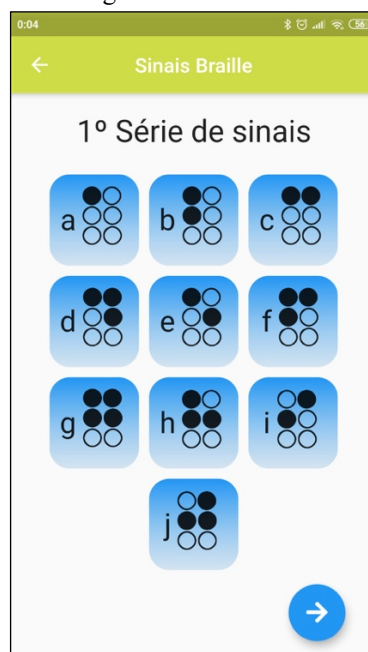


Fonte: elaborado pelo autor.

3.1.2 Funcionalidades do aplicativo

O aplicativo tem quatro funcionalidades, são elas: sinais, consulta, prática e máquina Braille. A opção sinais é composto por sete séries de sinais braille, na qual cada sinal representa um símbolo escrito (Figura 7). Cada símbolo tem um exemplo em que o usuário pode consultar. Os exemplos têm uma figura e um texto escrito em braille, e em letra de tinta. Para navegar entre as séries de sinais é utilizado os botões na parte inferior da tela para navegar.

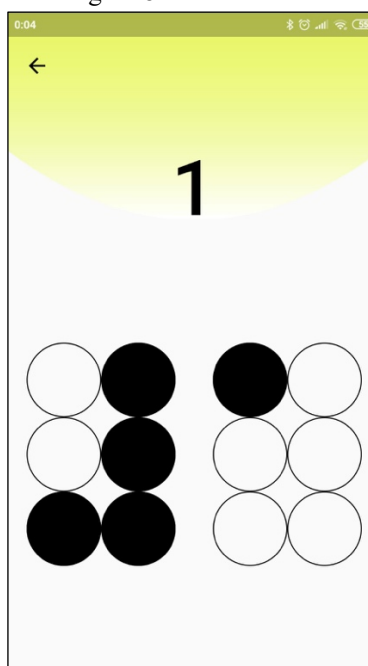
Figura 7 - Tela sinais



Fonte: elaborado pelo autor

Na opção *consulta* possui duas celas braille em que o usuário pode consultar os sinais digitando o símbolo em braille na tela (Figura 8). Na parte superior da tela será informada qual letra ou símbolo as células estão representando. Nem todos os sinais braille podem ser consultados, pois alguns poucos símbolos necessitam de mais de duas celas braille.

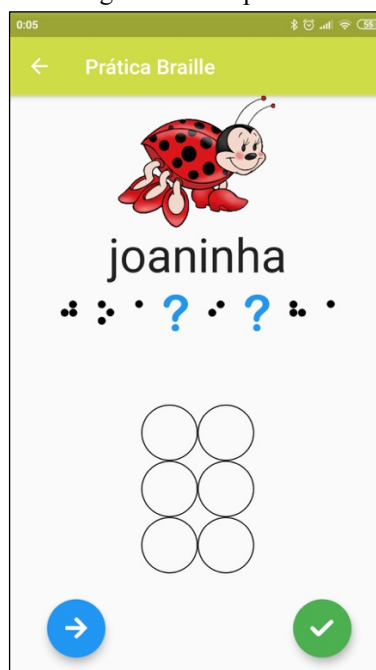
Figura 8 - Tela consulta



Fonte: elaborado pelo autor

Outra opção disponível é a *prática* que possibilita a prática do que foi aprendido (Figura 9). Nessa opção é exibido uma imagem com o nome embaixo e uma série de símbolos abaixo com um sinal braille faltando, para que o usuário possa digitar qual símbolo que está faltando.

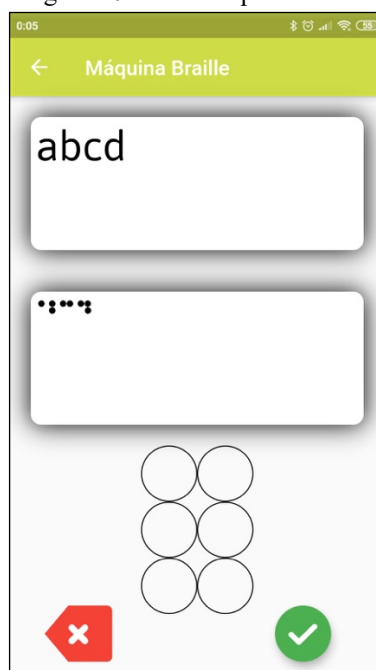
Figura 9 - Tela prática



Fonte: elaborado pelo autor

Na última opção, a Máquina Braille é composta por duas caixas de textos (Figura 10), uma cela braille e dois botões (confirmar e deletar). O usuário digita na cela braille os símbolos que deseja usar e nas caixas de texto acima é informado o texto em braille e em símbolo na representação de tinta. O usuário pode até mesmo digitar símbolos compostos, que utilizam mais de uma cela braille, que mesmo assim será reconhecido. No botão confirmar o usuário insere o símbolo que deseja digitar e o botão vermelho com “x” exclui o texto digitado.

Figura 10 - Tela máquina Braille



Fonte: elaborado pelo autor.

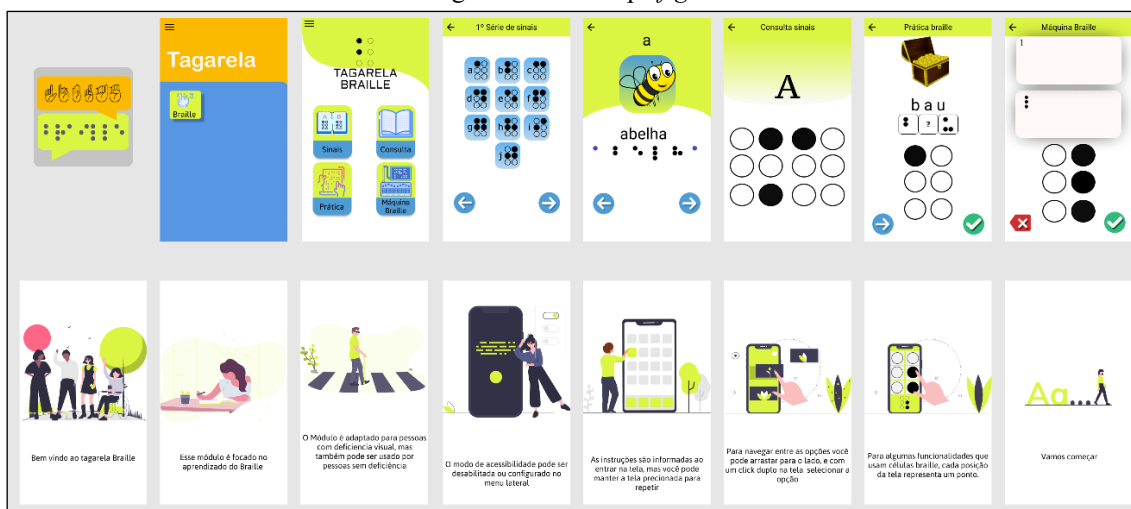
3.2 IMPLEMENTAÇÃO

Nas seções 3.2.1 e 3.2.2 se pretende apresentar os detalhes do desenvolvimento e especificações mais relevantes do aplicativo e as ferramentas utilizadas.

3.2.1 Ferramentas utilizadas

O projeto foi desenvolvido em Flutter, um *framework* para desenvolvimento de aplicativos para desktop, Android, iOS e WEB lançada em 2017 pela Google e utiliza a linguagem DART. Para a criação de um aplicativo é necessário utilizar um gerenciador de estado, que serve para atualizar a tela quando a variável tem seu valor alterado. Existem diversos gerenciadores de estado como o Bloc, MobX, GetX, setState, entre outros. Mas para esse projeto foi utilizado o MobX. O Flutter não tem uma estrutura de projetos padrão assim como outros *frameworks*. Mas nesse projeto foi utilizado uma estrutura de projeto do `flutter_modular` que serve também para a injeção de dependência, controlar as rotas e modularização. A estrutura do projeto criada pode ser vista no APÊNDICE A. Para a criação dos sinais braille foi utilizado o criador de fontes Calligraphr que é um criador de fontes on-line, onde todos os caracteres são desenhados e convertido em fonte de texto pela ferramenta como pode ser visto no APÊNDICE B. Antes de iniciar o desenvolvimento do aplicativo foi criado um protótipo usando o figma, que é uma ferramenta web para prototipação de sites e aplicativos móveis. As telas do protótipo estão representadas na Figura 11, as quais refletiram exatamente como ficaram as telas na aplicação final.

Figura 11 - Protótipo *figma*

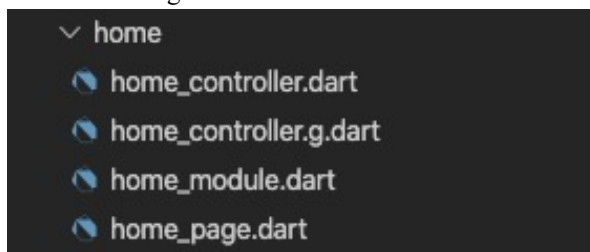


Fonte: elaborado pelo autor.

3.2.2 Desenvolvimento

O aplicativo inicia no módulo tagarela que é o módulo principal, e é a *home page* do aplicativo. Todos os módulos são compostos por quatro arquivos como mostra a Figura 12. Os arquivos `home_page.dart`, `home_controller.dart`, e `home_module.dart` são padrões de uma estrutura Model View Controller (MVC). E o quarto arquivo é o `home_controller.g.dart`, que é um arquivo utilizado pelo gerenciador de estado MobX para controlar a mudança de estado.

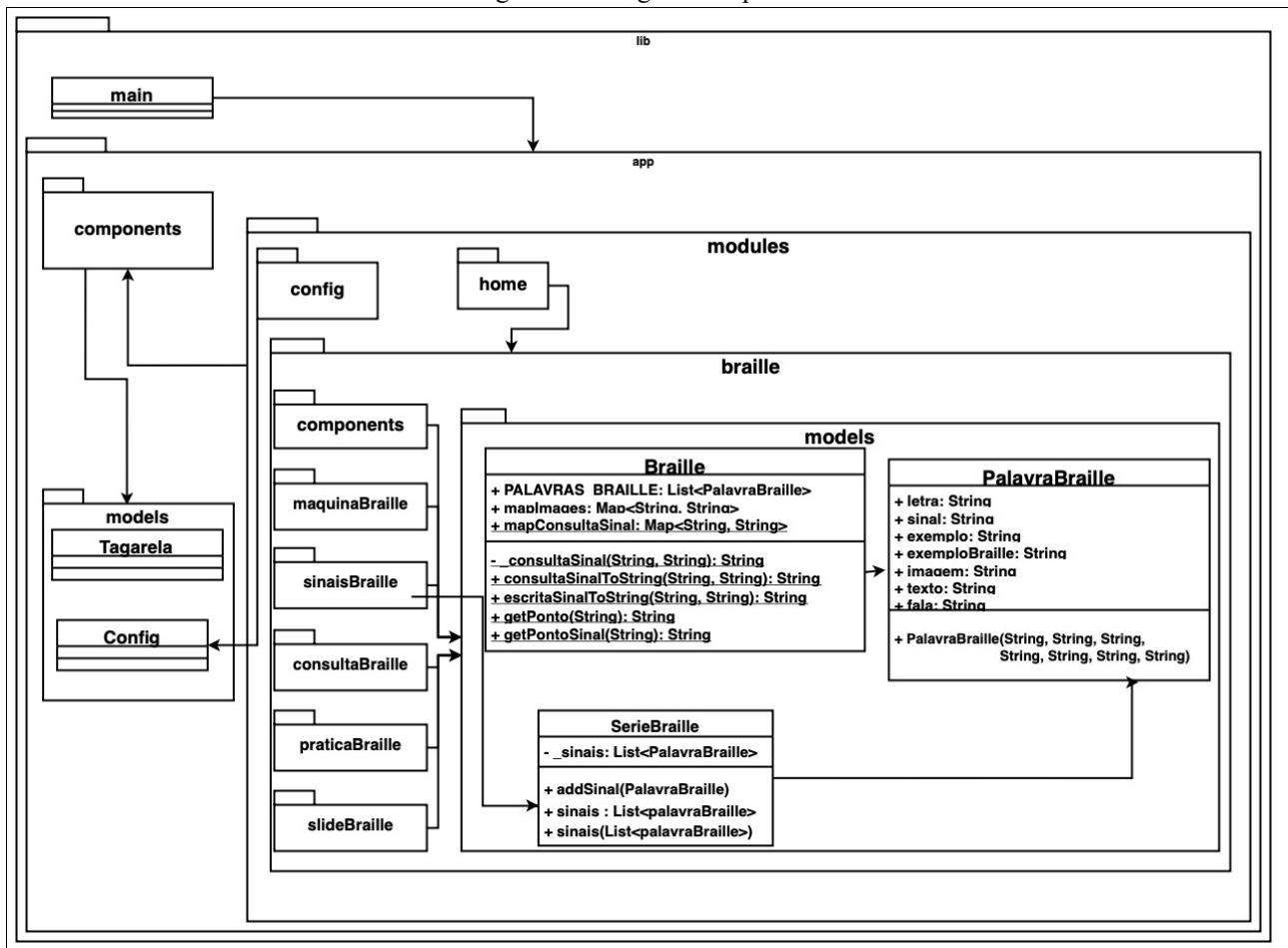
Figura 12 - Estrutura módulo.



Fonte: elaborado pelo autor.

A divisão dos módulos e componentes ficou como representado na Figura 13, que também mostra as principais classes. Essa divisão foi feita para melhorar a organização dos próximos módulos a serem implementados. O aplicativo tem duas classes principais `PalavraBraille` e `Braille`. Onde `PalavraBraille` é a classe de estrutura das palavras usadas em todo o módulo braille. Já a classe `Braille` contém os métodos principais de conversão do sinal braille para a letra de tinta e vice versa, e todas as palavras braille usadas nas séries de sinais.

Figura 13 - Diagrama de pacotes



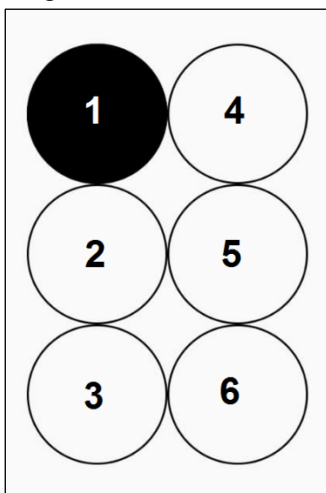
Fonte: elaborado pelo autor.

O atributo `PALAVRAS_BRaille` é uma lista de objetos `palavraBraille` e na tela sinais são divididas em sete series de sinais. Para isso na classe `SinaisBrailleController` é criado uma lista com sete `SerieBraille` e percorrida na página usando o componente `SinaisBrailleWidget` para mostrar os sinais. E ao clicar em um destes é exibido um exemplo para aquele sinal selecionado. Para os sinais braille foram criadas duas fontes específicas para esse projeto como mostra no APÊNDICE B. Uma fonte para as celas braille exibidos na tela de sinais e outra para os todos os outros sinais braille. Isso para que tenha uma melhor visualização dos sinais na tela de sinais. Como nem todos os sinais braille poderiam ter uma representação foi utilizado alguns caracteres especiais para representar certos sinais braille. Principalmente opção máquina Braille, em que o usuário pode digitar qualquer sinal e o texto em tinta é mostrado.

Os métodos `consultaSinalToString` e `escritaSinalToString` possuem a mesma funcionalidade, convertem sinal braille para um caractere, e para isso usam o método privado `consultaSinal`. Esse método recebe

por parâmetro dois sinais braille em binário. Por exemplo o símbolo “a” em braille seria o ponto 1 da cela braille marcado e todos os outros desmarcados como mostra a Figura 14.

Figura 14 - Pontos letra "a"



Fonte: elaborado pelo autor.

Para essa representação é convertido todos os pontos em binário 0 ou 1, quando 1 o ponto está marcado e 0 desmarcado, então a letra “a” seria 100000 onde os três primeiros dígitos seriam a primeira coluna da cela braille e os outros três seriam a segunda coluna. Como alguns sinais precisam de mais de uma cela, então é usado duas celas para converter em braille. A conversão é feita através do `map` `mapConsultaSinal` na qual a chave é o sinal braille em binário e o valor é o caractere como mostra o Quadro 5.

Quadro 5 - Map Pontos braille

```
426 static Map<String, String> mapConsultaSinal = {
427     '001111' + '100000': '1',
428     '001111' + '110000': '2',
429     '001111' + '100100': '3',
430     '001111' + '100110': '4',
431     '001111' + '100010': '5',
432     '001111' + '110100': '6',
433     '001111' + '110110': '7',
434     '001111' + '110010': '8',
435     '001111' + '010100': '9',
436     '001111' + '010110': '0',
437     '100000' + '000000': 'a',
438     '110000' + '000000': 'b',
```

Fonte: elaborado pelo autor.

A diferença entre os métodos `consultaSinalToString` e `escritaSinalToString` é que o `consultaSinalToString` trata sinais que não são escritos como, por exemplo o sinal que indica um número. Para isso são usadas constantes que em todo o aplicativo representam esse sinal. Pois para representar um número é utilizado uma cela com os pontos 3,4,5 e 6 marcados, e dependendo da próxima cela representar o número. Esse método é utilizado na tela de consulta em que deve mostrar o símbolo independente do sinal apresentado. Já o `escritaSinalToString` é usado na tela de prática e máquina Braille. Para utilizar esses métodos foi criado um componente `CelaBrailleWidget` que é composto por seis pontos formando uma cela braille. Esse componente retorna em binários quais pontos estão marcados, e assim podem ser usados pelos métodos de conversão descritos acima.

O componente principal para acessibilidade do aplicativo é o `GestureAccessibilityWidget`. Esse componente captura os gestos na tela e executa a ação dependendo dos gestos capturados. Para isso ele utiliza do componente `GestureTagarelaWidget` que através dos pontos x e y da posição dos gestos capturados na tela, retorna

um enum que representa a posição como pode ser visto no trecho de código no Quadro 6. Também faz uso do componente `GestureDetector`, que executa um método sempre que um toque na tela é detectado e retorna os pontos x e y.

Quadro 6 - Metodo getPosition

```
77 PositionTap getPosition(double x, y) {
78     Size size = MediaQuery.of(context).size;
79     if (x < size.width * .5 && y < size.height * .33) {
80         return PositionTap.leftTop;
81     } else if (x > size.width * .5 && y < size.height * .33) {
82         return PositionTap.rightTop;
83     } else if (x < size.width * .5 && y < size.height * .66) {
84         return PositionTap.leftCenter;
85     } else if (x > size.width * .5 && y < size.height * .66) {
86         return PositionTap.rightCenter;
87     } else if (x < size.width * .5) {
88         return PositionTap.leftButton;
89     } else
90         return PositionTap.rightButton;
91 }
92 }
93
```

Fonte: elaborado pelo autor.

Para reproduzir os textos de forma dinâmica é usado a classe `Tagarela` que utiliza de um pacote `flutter_tts` que ler e reproduz os textos nativamente. Alguns métodos foram criados para padronizar e reutilizar essa funcionalidade em todo o aplicativo. Por exemplo, através do método `speak` os textos são lidos e reproduzidos, e todos os controles de reprodução como controle do volume são feitos na própria classe.

A acessibilidade do aplicativo foi possível usando o componente `GestureAccessibilityWidget` e a classe `Tagarela`. As instruções iniciais e as ações a serem tomadas são passados por parâmetros no momento de criação do componente (Quadro 7), internamente o componente utiliza o método `speak` para reproduzir os textos passados por parâmetros ou os retornados ao executar o método `onTap` e assim retornando o texto a ser reproduzido, as ações tomadas ao deslizar na tela são passadas por parâmetros através de uma lista de objetos `OptionGesture`, que contem a ação a ser executada e o texto a ser reproduzido.

Quadro 7 - Uso componentes acessibilidade

```
31 GestureAccessibilityWidget([
32     active: Tagarela.config.accessible,
33     onTap: (position) {
34
35         String pontos1 = Braille.getPontoSinal(controller.sinal1);
36         String pontos2 = Braille.getPontoSinal(controller.sinal2);
37         return '${(pontos1 != ' ' ? 'Pontos primeira célula, ' + pontos1 : ' ')}';
38     },
39     options: [
40         OptionGesture(
41             action: () {
42                 cela = 1;
43             },
44             speak: 'Célula Braille 1, click duas vezes para confirmar'), // OptionGesture
45         OptionGesture(
46             action: () {
47                 cela = 2;
48             },
49             speak: 'Célula Braille 2, click duas vezes para confirmar') // OptionGesture
50     ],
51     primarySpeak:
52         'Consulta sinais braille, \n click na tela para marcar um ponto, \n pontos 1 e 4 na parte superior, \n ' +
53         ' pontos 2 e 5 no centro e pontos 3 e 6 na parte inferior. \n para navegar entre as células Braille arraste para o lado',

```

Fonte: elaborado pelo autor.

Para agilizar o desenvolvimento de novas opções e funcionalidades foi usado a estrutura de módulos, em que cada módulo tem seus módulos filhos e componentes padrões apenas para o módulo específico ou para o filho. Como é o caso do `CardLetraBrailleWidget` que foi desenvolvido apenas para o módulo `Braille`. Mas existem também componentes que podem ser usados em todo o aplicativo, como é o caso do `BoxTextWidget`, que é o componente de

caixa de texto existente na tela de máquina Braille. Isso foi feito para que os componentes possam ser reaproveitados e assim possibilitem uma padronização e agilizem o desenvolvimento de novos módulos para o aplicativo. Os componentes criados são sempre dinâmicos possibilitando assim o seu reuso, assim podendo ser utilizado.

4 RESULTADOS

Nessa seção serão apresentadas as considerações do autor, resultados dos testes realizados pelos usuários e um comparativo entre os correlatos e aplicativo desenvolvido.

4.1 CONSIDERAÇÕES DO AUTOR

O Aplicativo foi desenvolvido para dispositivos Android e iOS usando o *framework* Flutter, e foi postado nas lojas de aplicativos *app store* e *play store*. Ambas as lojas já aprovaram o aplicativo, e o mesmo está disponível para downloads. Inicialmente foi pensado em utilizar os mesmos métodos utilizados no projeto do Lucas Cazagrande em 2016(ver seção 2.3), como o de utilização de imagens para representar os sinais braille. Mas tendo em vista que a utilização de imagens aumentaria o tamanho do aplicativo, e o intuito é que sejam adicionados novos módulos, o tamanho do aplicativo pode aumentar muito. Então foi criado as próprias fontes de texto, pois com elas além de diminuir o tamanho do aplicativo, deixou mais fácil e rápido o seu desenvolvimento. Assim como as fontes de texto o áudio também foi pensado na possibilidade de usar áudios pré-gravados, mas seria muito mais difícil de fazer com que os textos dinâmicos tivessem o resultado esperado. Além de disso também foi pensado em usar API's externas para realizar a leitura dos textos. Mas assim o modo acessibilidade só funcionaria on-line, então a melhor solução foi usar o pacote `flutter_tts`, A desvantagem desse pacote é que a versão mínima do Android suportado é a 5.0, mas como de acordo com a própria Google developer (2020) 94,1% dos usuários usam a versão 5.0 ou superior. Isso não se tornou um grande problema.

Nos testes realizados todas as funções foram executadas com êxito, e os áudios ficaram claros, apesar de que em alguns dispositivos a voz inicial ficou rápida, mas isso pode ser corrigido nas configurações. As instruções de áudio nas páginas são dadas de forma que o usuário possa utilizar a tela facilmente, mas talvez algumas instruções não seriam necessárias serem repetidas, uma vez que o usuário já esteja habituado com o aplicativo.

Na opção de *sinais* foi percebido uma dificuldade para navegar nos botões de navegabilidade com o modo acessibilidade ativo, provavelmente porque a detecção dos gestos esteja capturando o gesto de deslizar na tela em vez do de clique, para corrigir isso poderia ser capturado mais gestos, como o deslizar pra cima, e assim associar o gesto aos botões, além de que as instruções ficariam mais claras. Com modo acessibilidade desabilitado as funções funcionam perfeitamente, e os botões de navegação são bem intuitivos, os exemplos ficaram claros e o texto visível.

Na tela de *consulta* a usabilidade está boa, as instruções estão bem claras, tanto com o modo acessibilidade habilitado como desabilitado, mesmo para sinais com mais de uma cela braille os símbolos têm representação, e são apresentados imediatamente, essa tela ficou simples e objetiva.

Na opção de *prática* a tela ficou um pouco sem objetivo, apesar de acertar as letras faltantes o nível de dificuldade não aumenta, e ao acertar a letra não é informado para o usuário que a letra está correta ou errada, simplesmente a palavra é trocada. Talvez uma animação ao acertar ou errar a letra deixaria a funcionalidade mais didática e intuitiva. Mesmo assim o objetivo de praticar o que foi aprendido nos outros módulos foi alcançado, pois apesar de não ter um nivelamento da dificuldade das palavras, ainda assim o usuário consegue praticar.

A tela de *máquina Braille* não existia no projeto inicial do Lucas Cazagrande (2016), e foi criado para nesse projeto. Nessa tela os componentes visuais estão bem intuitivos, e o modo acessibilidade as instruções estão claras, os textos são lidos de forma dinâmica, ou seja, as palavras são lidas e não soletradas. Os textos estão bem visíveis, ao incluir símbolos com mais de uma cela braille ainda assim os textos têm correspondência com o texto em representação em tinta.

A utilização do Flutter, em vez de IONIC inicialmente usado por Cazagrande em 2016, foi pensada para obter uma melhor performance, pois tendo em vista o crescimento do aplicativo e as possibilidades de implementação, o Flutter se mostra mais performático em relação ao IONIC, pois sua arquitetura utiliza de componentes nativos sem intermediários como o cordova utilizado pelo IONIC. Com a utilização de Flutter todas as telas tiveram os resultados esperados, a transição entre telas e animações fluem de forma leve e não foram encontrados travamentos nas transições. O gerenciador de estado MobX facilitou a implementação, mas acredito que o uso de outro gerenciador de estados como GetX teria deixado o aplicativo ainda mais rápido e fácil implementação, principalmente por conta das compilações necessárias que poderá ficar um pouco lenta com a adição de novos módulos. A componentização além de facilitar a implementação trouxe o benefício de fazer os componentes seguirem um padrão, assim como os botões que se pode observar essa padronização outros componentes *header* da página, também utilizaram a componentização de componentes. Um outro grande benefício alcançado com a componentização é a possibilidade de utilização dos componentes de acessibilidade por outros aplicativos, tornando assim mais fácil a utilização em outros projetos e aumentando o número de aplicativos acessíveis.

4.2 TESTE DE UTILIZAÇÃO

Para coleta de dados foi utilizado um formulário que pode ser visto no APÊNDICE C. Este formulário instruía os usuários a instalarem o aplicativo e executar funções, como abrir a opção de sinais e navegar até uma determinada letra. E, em seguida foi solicitado que o usuário digitasse um símbolo na tela de consulta, depois o usuário teria que entrar na tela de prática e informar qual palavra foi mostrada. Para os usuários sem deficiência visual foi encontrado grande dificuldade de executar as funções com o modo acessibilidade ativo, então todas as respostas foram com o modo acessibilidade desabilitados. Os testes foram realizados com 10 usuário na faixa etária entre 19 e 57 anos, na qual 90% desses conheciam, mas não sabiam ler braille e 10% nunca ouviu falar. Destes usuários 80% consideraram que o aplicativo tem boa usabilidade e 10% consideraram a usabilidade regular ou péssima, tanto para pessoas com deficiência como para pessoas sem deficiência. Foi perguntado também se eles consideravam que o aplicativo cumpriu com o objetivo de estimular o aprendizado do braille, e 50% dos usuários consideraram ótimo, 30% consideraram bom, 10% consideraram regular e 10% consideraram péssimo. Acreditasse que as notas baixas deveriam ao fato de que o modo acessibilidade é ativado por padrão e alguns usuários não conseguiram usar o aplicativo normalmente. Isso poderia ser solucionado com uma melhor instrução aos usuários, por exemplo um guia de primeiros passos, em que o usuário seria guiado para ter as primeiras impressões e conhecer as funcionalidades do aplicativo. O aprendizado do braille poderia ter tido melhores resultados se na tela de prática houvesse mais exercícios para praticar e aprender, e tornar a tela um pouco mais divertida e com mais objetivos para aumentar o estímulo dos usuários.

Não foi realizado testes com deficientes visuais, mas ao simular a ausência da visão, utilizando o aplicativo com os olhos cobertos, navegando pelas opções do aplicativo e executando ações básicas e complexas. Apesar que os testes não fossem precisos essa simulação serviu como guia para pessoas que perderam a visão recentemente, e os testes provaram que a utilização do aplicativo é eficaz mesmo por usuários sem visão, necessitando apenas um tempo para o usuário se habituar com as funcionalidades. Foi percebido que as instruções passadas ao entrara na tela deveriam ser mais bem explicadas para deixar mais claras as instruções, pois em alguns pontos do aplicativo as instruções foram um pouco confusas.

4.3 COMPARATIVO COM CORRELATOS

Em comparação com os trabalhos correlatos, o aplicativo se mostrou mais completo e ainda sendo multiplataforma, proporcionando o uso por um número maior de usuários. Como pode ser observado no Quadro 8.

Quadro 8 - Comparativo com os correlatos

	Aprende Braille (UGEDO, 2016)	AbcNum (AQUINO, W. <i>et al.</i> 2015)	LêBraille (FAÇANHA <i>et al.</i> 2012)	Tagarela Braille
Plataforma	Android	Android	Android	Android, iOS
Feedback com áudio	Sim	Sim	Sim	Sim
Captura de gestos	Não	Não	Sim	Sim
Ensino de escrita	Não	Sim	Sim	Sim
Ensino de Leitura	Sim	Sim	Não	Sim
Consultar sinais braille	Sim	Não	Não	Sim

Fonte: elaborado pelo autor.

5 CONCLUSÕES

Este trabalho apresentou um aplicativo que pode auxiliar no aprendizado do braille utilizando o conceito de componentização para incentivar a criação de novos módulos. Ao longo deste artigo foram descritas as funcionalidades, implementação e ferramentas utilizadas no desenvolvimento deste aplicativo.

A ideia inicial do aplicativo idealizada por Lucas Cazagrande (2016), era que o aplicativo auxiliasse aos professores, pais de deficientes visuais ou entusiastas do braille, tornando o ambiente em volta do deficiente visual mais acessível, mas o aplicativo descrito nesse artigo tornou possível a utilização por pessoas com pouca ou total perda de visão, isso sem perder a essência inicial, pois ainda possibilita que pessoas normovisuais possam utilizar o aplicativo. O aplicativo se mostrou eficiente apesar de ainda haver melhorias a serem feitas, como por exemplo mais instruções aos usuários e algumas melhorias em telas para tornar o aplicativo mais intuitivo. Muitas vezes algumas funcionalidades por si só não serão suficientes para o total aprendizado do braille, por isso a ferramenta busca auxiliar o aprendizado, servindo como mais uma ferramenta de inclusão. A grande maioria dos aplicativos não possuem uma acessibilidade, e esse sendo uma ferramenta voltada a aprendizado do braille não poderia deixar de ser inclusiva.

De acordo com os dados coletados o aplicativo cumpriu com o objetivo de auxiliar o aprendizado do braille, e com o modo acessibilidade permitiu o acesso ao aplicativo a usuários com deficiência visual. Todas as telas tiveram um

bom desempenho e suas funcionalidades ficaram claras para os usuários sem e com deficiência visual, o áudio com as instruções foram bem didáticas para passar as informações necessárias para os usuários. Em comparação aos correlatos o aplicativo se mostrou mais completo por ser o único que possui todas as funcionalidades como consulta, escrita e leitura do braille, e se mostrou mais completo nas funcionalidades de feedback com áudio e interação por gestos na tela, além de que foi desenvolvido para iOS e Android.

A estrutura apresentada facilita a criação de novos módulos e até mesmo a remoção de módulos existentes, caso o desenvolvedor queira criar os módulos individualizados e posteriormente adicioná-lo ao aplicativo, caso tenha sido criado com a mesma estrutura. Os componentes são dinâmicos permitindo que sejam utilizados em outros projetos ou em outros módulos.

Apesar de todas os objetivos alcançados pelo aplicativo pode-se perceber que além da inclusão de novos módulos, algumas funcionalidades poderiam tornar o aplicativo mais completo. E ficam como sugestões para trabalhos futuros:

- a) criar *login* para o aplicativo salvar o progresso dos usuários;
- b) incluir novos módulos para o aplicativo Tagarela;
- c) criar funcionalidade de conversão de texto do alfabeto escrito para o braille;
- d) migração para versão *web*;
- e) criação de novos exercícios para a prática Braille;
- f) tornar as configurações do aplicativo acessíveis;
- g) integração com IOT.

REFERÊNCIAS

- AQUINO, Wermeson *et al.* AbcNum Braille: Proposta de um Aplicativo para Auxiliar no Aprendizado do Alfabeto Braille para Pessoas com Baixa Visão. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO - SBIE, XXVI. 2015, **Anais...** Disponível em: <https://br-ic.org/pub/index.php/sbie/article/view/5372/3733>. Acesso em: 27 nov. 2020.
- BRASIL. Lei nº 7853, de 24 de outubro de 1989. Dispõe sobre o apoio às pessoas portadoras de deficiência, sua integração social, sobre a Coordenadoria Nacional para Integração da Pessoa Portadora de Deficiência - Corde, institui a tutela jurisdicional de interesses coletivos ou difusos dessas pessoas, disciplina a atuação do Ministério Público, define crimes, e dá outras providências. DOFC de 25/10/1989, p. 1920.
- BRASIL. Ministério da Educação. Secretaria de Educação Continuada, Alfabetização, Diversidade e Inclusão. **Grafia Braille para a Língua Portuguesa**. Brasília-DF, 2018, 3a edição. 95p.
- CAZAGRANDA, Lucas. **aprendendo braille**: o ensino do sistema braille com o uso do tagarela. 2016. 58 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- CIVIAN. **Você sabe o que é reglete?**, [2017?]. Disponível em: <https://www.civiam.com.br/blog/voce-sabe-o-que-e-reglete/>. Acesso em: 27 nov. 2020.
- COSTA, Renata. **Como funciona o sistema Braille?**, [2009]. Disponível em: <https://novaescola.org.br/conteudo/397/como-funciona-sistema-braille>. Acesso em: 27 nov. 2020.
- FAÇANHA, Agebson et al. Auxiliando o Processo de Ensino-Aprendizagem do Braille Através de Dispositivos Touch Screen. **Informática na educação**: teoria & prática. Porto Alegre, v.15, n.2, jul./dez. 2012.
- Google Developers. **Painel de distribuição** [2020]. Disponível em: <https://developer.android.com/about/versions/android-5.0.html>. Acesso em: 29 nov. 2020.
- Instituto Brasileiro de Geografia e Estatística (Org.). **Conheça o Brasil - População**: Pessoas com deficiência. [2017?]. Disponível em: <https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/20551-pessoas-com-deficiencia.html>. Acesso em: 28 nov. 2020.
- INSTITUTO BENJAMIN CONSTANT. **O Sistema Braille**. [2018]. Disponível em: http://www.abc.gov.br/index.php?option=com_content&view=article&id=675:o-sistema-braille&catid=121&Itemid=373. Acesso em: 27 nov. 2020.
- MARTA GIL(Org.). **Caderno da tv escola**: deficiência visual. Brasília: Secretaria de Educação, 2000. 80 p.
- MASINI, Elcie F. Salzano; GASPARETTO, Maria Elisabete Rodrigues Freire (Org.). **Visão subnormal**: Um enfoque educacional. São Paulo: Vetor, 2007.
- OLIVEIRA, Regina. **Braille nos dias de hoje**: objeto de vitrine ou ferramenta indispensável?. 2016. Disponível em: <https://www.fundacaodorina.org.br/blog/braille-nos-dias-de-hoje-objeto-de-vitrine-ou-ferramenta-indispensavel/>. Acesso em: 28 nov. 2020.
- OLIVEIRA, Renato Gonçalves de. Desenvolvimento baseado em componentes: **Revista Java Magazine**, Guarujá, v. 110, p. 40-45, 2012. Disponível em: https://arquivo.devmedia.com.br/Diagramacao/Revistas_PDF/JM/JAVA-MAGAZINE_110_NUJBNCW.pdf. Acesso em: 28 nov. 2020

OTSUKA, Daniela. **Braille** 2010. Disponível em: <https://www.infoescola.com/portugues/braille/>. Acesso em: 29 nov. 2020.

ROLLINGS, Andrew; MORRIS, Dave. **Game Architecture and Design**: A New Edition. Indiana: New Riders, 1999. p. 742

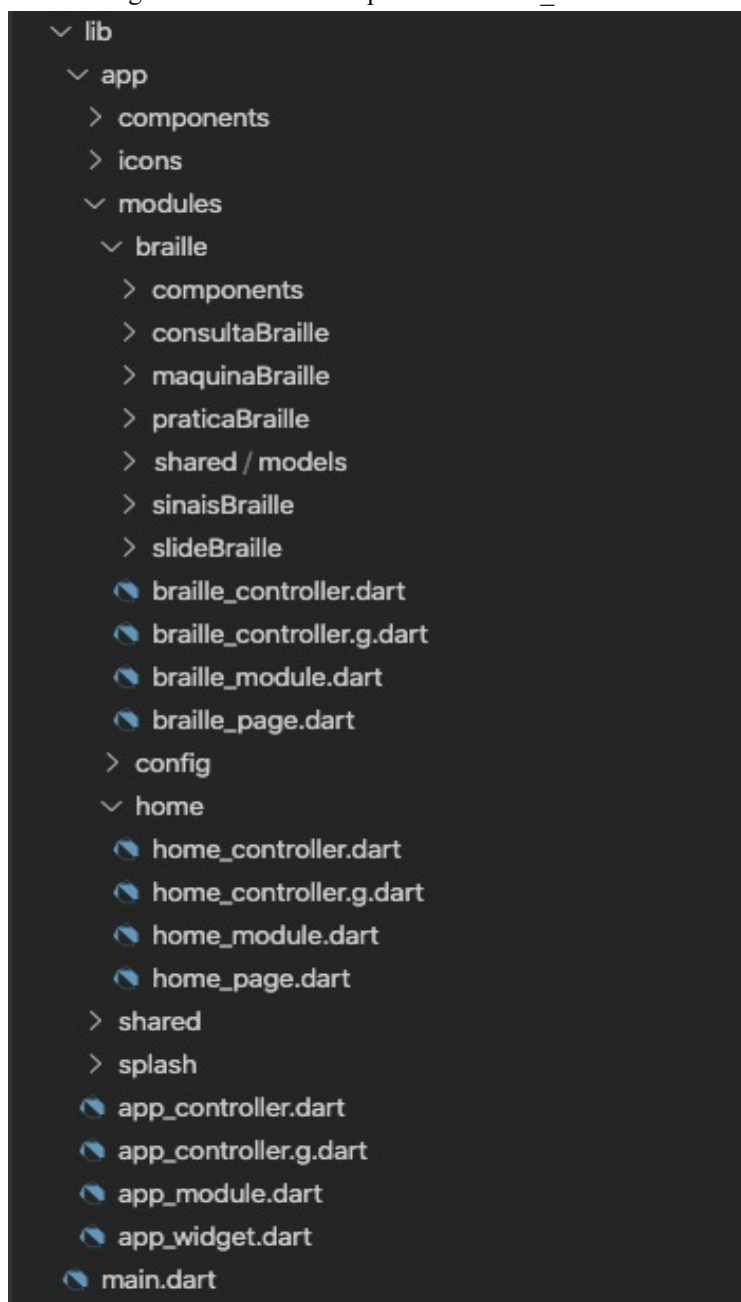
UGEDO, Mario. **Aprende Braille**. 2016. Disponível em:

<https://play.google.com/store/apps/details?id=com.comoelagua.android.braille>. Acesso em: 17 nov. 2020.

APÊNDICE A – ESTRUTURA DE PASTAS FLUTTER MODULAR

Na Figura 15 apresenta a estrutura de pastas utilizada pelo flutter_modular.

Figura 15 - Estrutura de pastas flutter_modular


































































Fonte: elaborado pelo autor.

APÊNDICE B – ARQUIVO FONTES CALLIGRAPHR



Neste apêndice são apresentados alguns dos arquivos utilizados para criação das fontes dos sinais Braille. A Figura 16 mostra o arquivo que foi utilizado para criar a fonte sem borda utilizada em todo o aplicativo. Já a Figura 17 o arquivo utilizado para criação da fonte com borda utilizada na tela de *signal*.




























































Figura 16 - Arquivo de criação de fonte sem borda


<div>  calligraphr  </div>							
!	"	%	&	'	(	
							
)	+	,	-	.	/		
							
0	1	2	3	4	5	6	7
							
8	9	:	;	=	?	@	A
							
B	C	D	E	F	G	H	I
							
J	K	L	M	N	O	P	Q
							
R	S	T	U	V	W	X	Y
							
Z	a	b	c	d	e	f	g
							
<div>  Include all those four markers ← untrimmed on your photo or scan. www.calligraphr.com  </div>							

Fonte: Organizado pelo autor.

Figura 17 - Arquivo de criação de fonte com borda



calligraphr


!	"	%	&	'	(	
							
)	+	,	-	.	/		
							
0	1	2	3	4	5	6	7
							
8	9	:	;	=	?	@	A
							
B	C	D	E	F	G	H	I
							
J	K	L	M	N	O	P	Q
							
R	S	T	U	V	W	X	Y
							
Z	a	b	c	d	e	f	g
							



Include all those four markers
← untrimmed on your photo or scan.

www.calligraphr.com

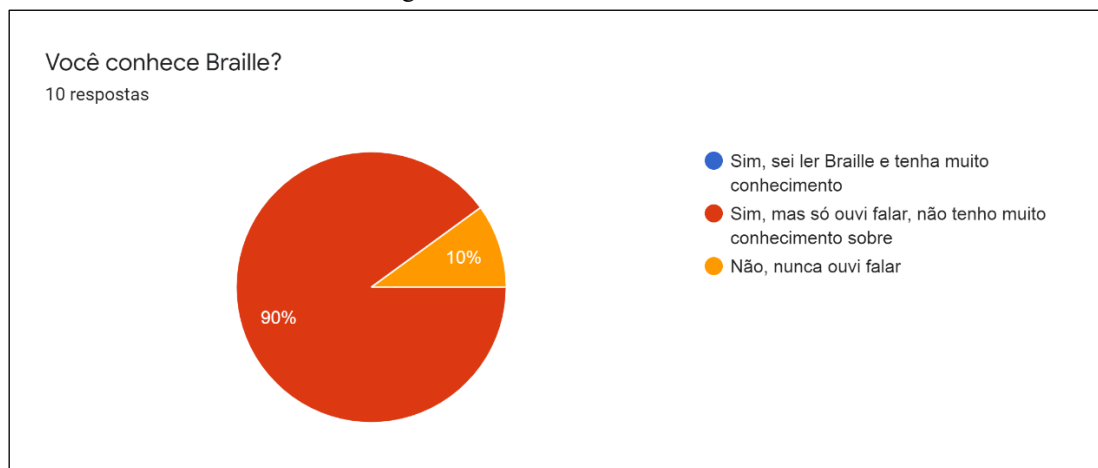


Fonte: Organizado pelo autor.

APÊNDICE C – QUESTIONÁRIO

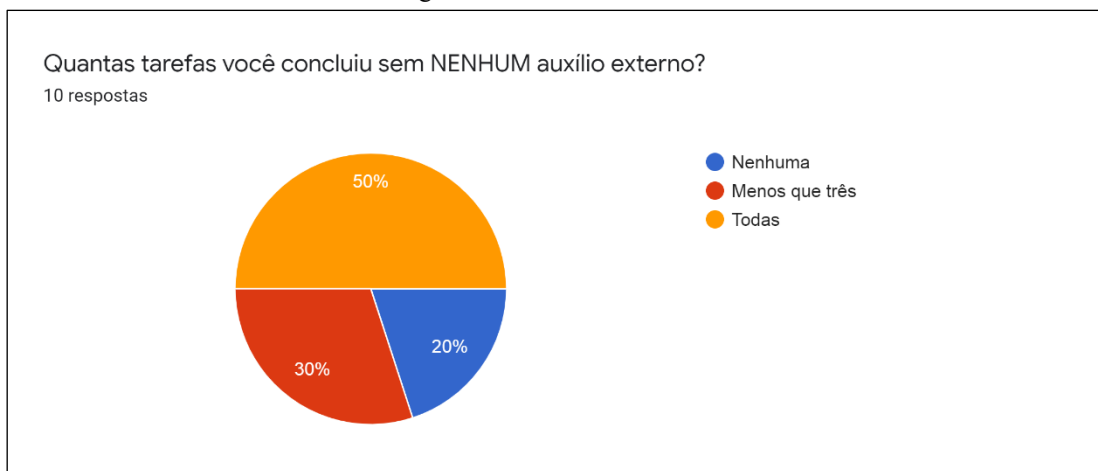
O questionário foi realizado com dez usuários que seguiram uma série funcionalidades do aplicativo, dentre esses 10% nunca tinha ouvido falar, e o restante só ouviu falar, mas não tinha conhecimento sobre (Figura 18). Na Figura 19 mostra que 50% conseguiu concluir todas as funcionalidades sem auxílio externo. Dos usuários que testaram o aplicativo 90% classifica a usabilidade do aplicativo como boa ou excelente. E 90% consideram que o aplicativo cumpriu com o objetivo de estimular o ensino do Braille.

Figura 18 - Conhecimento Braille



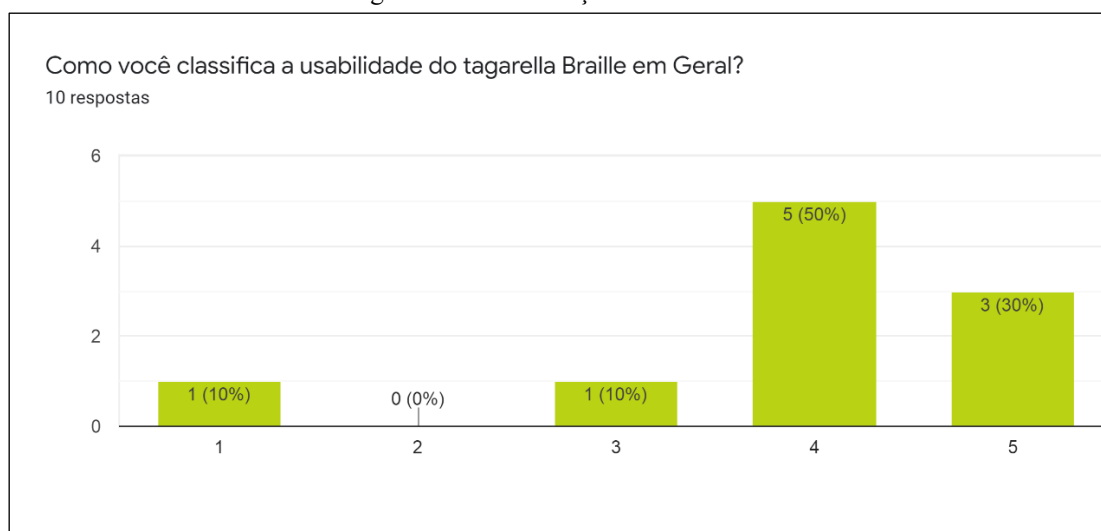
Fonte: elaborado pelo autor.

Figura 19 - Tarefas concluídas



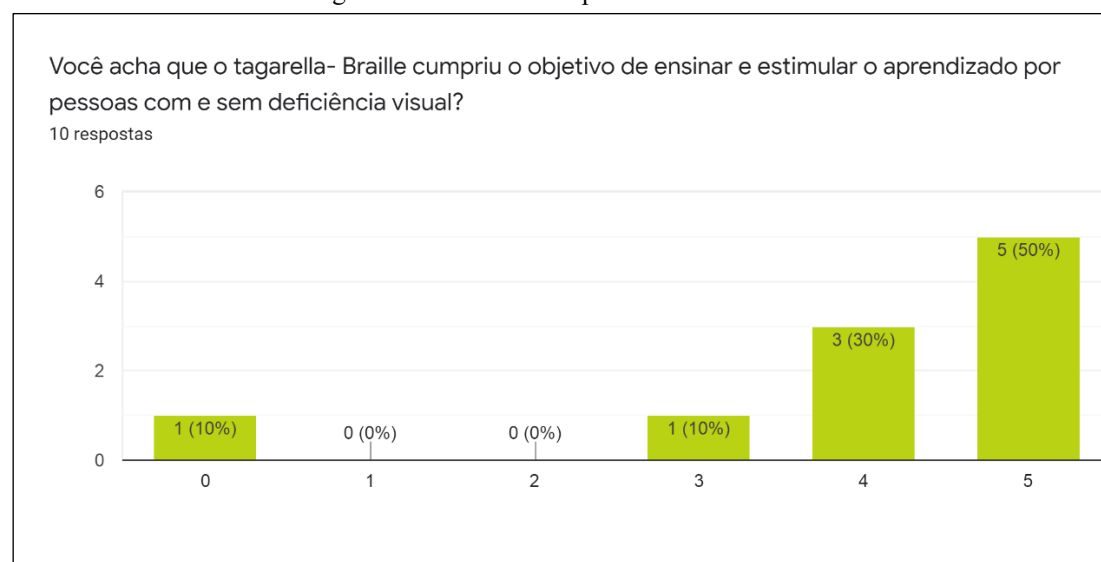
Fonte: elaborado pelo autor.

Figura 20 - Classificação de usabilidade



Fonte: elaborado pelo autor.

Figura 21 - Estímulo ao aprendizado do Braille



Fonte: elaborado pelo autor.