Igor Calabria da Fonte

Uma Ferramenta para Personalização de Vocabulário de Prancha de Comunicação Alternativa

Brasil

2016

Igor Calabria da Fonte

Uma Ferramenta para Personalização de Vocabulário de Prancha de Comunicação Alternativa

Universidade Federal de Pernambuco - UFPE

Centro de Informática

Orientador: Robson Fidalgo

Brasil

2016

Este trabalho é dedicado a todos os educadores que passaram pela minha vida, educadores que fazem parte do pilar mais importante da sociedade.

Agradecimentos

Os agradecimentos principais vão para ao orientador Robson Fidalgo, que, com muita paciência, guiou o desenvolvimento deste trabalho. Edson Alves e Thiago Pinheiro também contribuíram de forma significativa com *insights* e opiniões.

Por fim, agradeço ao Centro de Informática por ter possibilitado a criação deste trabalho.

Resumo

Pranchas de comunicação são uma forma de tecnologia assistiva que têm como objetivo auxiliar portadores de necessidades especiais a se comunicar. Essas pranchas de comunicação possuem vocabulários que precisam ser criados e modificados. A criação de uma ferramenta para manipular esses vocabulários agrega problemas como completude de funcionalidades e a facilidade de uso. Este trabalho implementa uma ferramenta que provê a customização de uma maneira intuitiva para o usuário: a interface é similar a um sistema de manipulação de arquivos encontrado nos principais sistemas operacionais. Partindo dessa familiaridade, o usuário não precisa ser introduzido a novos conceitos para que ele seja capaz de utilizar a ferramenta de criação e customização proposta.

Palavras-chave: Tecnologia Assistiva. Customização. Implementação.

Sumário

1 Introdução

Este capítulo descreve o contexto necessário para tornar clara as motivações e decisões tomadas durante o processo de desenvolvimento. É apresentada uma introdução sobre Tecnologia Assistiva e Comunicação Alternativa e Aumentativa. Essa introdução serve como base para a apresentação dos objetivos e das motivações principais deste trabalho.

1.1 Apresentação de contexto

A Tecnologia Assistiva (TA) é uma área de conhecimento e pesquisa que tem como um dos objetivos auxiliar as Pessoas com Necessidades Especiais (PNE) (DAMASCENO; FILHO, 2002). Além desse auxílio, a TA também tem o compromisso de potencializar a participação do PNE na sociedade. Ou seja, além de tentar reduzir as limitações, a TA amplifica o potencial do indivíduo.

Dentro da TA, encontra-se a área de comunicação alternativa e aumentativa (CAA) (MIRANDA; ICD, 2004). CAA é a utilização de meios alternativos para promover a habilidade de comunicação a pessoas com algum impedimento de fala. A comunicação pode ser provida por meio de auxílios externos, que potencializam a capacidade do usuário de se comunicar. Cartões de comunicação, vocalizadores, pranchas alfabéticas e pranchas de comunicação alternativa são exemplos desses auxílios.

As pranchas de comunicação alternativa (PCA) podem ser descritas como um conjunto de símbolos que representam um certo vocabulário (BERSCH, 2008). No contexto de PCA, o símbolo é uma imagem acompanhado de uma legenda. O símbolo junto com a legenda representam um elemento no vocabulário, e o usuário da prancha pode compor esses elementos para se comunicar. As pranchas existem em várias formas, desde recortes em cartolina até aplicativos sofisticados que dão suporte à vocalização de sons gravados.

1.2 Motivação

Como uma tecnologia assistiva, a PCA deve cumprir o papel de auxiliar o usuário. O cumprimento ou não desse papel depende, entre outros fatores, do vocabulário da prancha. Se a prancha possui um vocabulário incompleto ou mal categorizado, as necessidades do usuário não vão ser satisfeitas. Por exemplo, se uma criança portadora de transtorno do espectro autista (TEA) precisa pedir alimentos utilizando a prancha, e o vocabulário dessa prancha não possui nenhum, isso significa que a prancha falhou com seu objetivo.

A incompletude de um vocabulário é uma das falhas que prejudica o papel da prancha como ferramenta de auxílio à PNE. Por outro lado, existe o problema contrário. Um vocabulário com um número elevado de símbolos pode aumentar a carga cognitiva do usuário e até atrapalhar a comunicação. Fallon, Light e Paige (2001) apresentam técnicas para melhorar a seleção do vocabulário. Uma de suas conclusões é que a relevância dos símbolos escolhidos é fator decisivo na aceitação da PCA.

Existe também o problema de categorização e classificação dos elementos. O problema consiste no desafio de organizar esses símbolos de tal forma que o usuário da prancha se beneficie do posicionamento dos itens do vocabulário. Fallon, Light e Achenbach (2003) fizeram um estudo que avalia a importância dessa organização semântica, e suas conclusões indicam que a organização melhora o acesso a os itens da prancha. Baseado nessa melhora, nota-se que listar todos os símbolos sem nenhuma hierarquia faz com que a prancha possa não cumprir o papel de facilitadora da comunicação. Franco et al. (2015) propõem organizar o vocabulário de maneira a se encaixar nos modelos mentais do usuário. Essa abordagem é interessante, porque a categorização pode facilitar a busca de símbolos e até potencializar a eficácia da PCA.

A fim de mitigar o problema da completude e da categorização, propõe-se a funcionalidade de customização da PCA. Com a customização, o vocabulário da prancha pode ser criado de forma incremental de acordo com as necessidades específicas de cada usuário. Vale ressaltar que a customização é uma ferramenta para resolver os problemas apresentados. Ferramenta que permite aplicar modelos de categorização e de vocabulário de forma experimental para cada usuário. Espera-se que essa capacidade de customização aumente a qualidade do vocabulário e, consequentemente, aumente a eficácia da prancha.

A customização do vocabulário da prancha deve ser provida por uma GUI[1] que possui essa capacidade. Construir uma GUI efetiva tem seus desafios: a ferramenta tende a ser usada por pessoas leigas e certas funcionalidades podem não ser familiares a esses indivíduos. Por exemplo, como deve ser apresentada a hierarquia dos elementos do vocabulário? Como deve ser dada uma visão geral? como mostrar os elementos atuais? Franco et al. (2014) implementaram uma ferramenta CASE que responde essas perguntas, mas a sua ferramenta não foi bem aceita pelos usuários. Esse caso exemplifica que criar uma interface fácil de usar para customização da prancha não é uma tarefa trivial. Partindo dessa dificuldade, é proposta a criação de uma interface baseada na GUI de um sistema de arquivos. Sistemas de arquivos são ferramentas familiares aos potenciais usuários e, ao mesmo tempo, pode ser criado um paralelo entre pastas e arquivos com os itens do vocabulário.

A possibilidade de aumentar a eficácia também é motivação deste trabalho. Em um contexto no qual as pranchas são usadas por crianças portadoras de TEA, a flexibilidade

[1] Interface de usuário

do vocabulário usado é significativa. Segundo Ganz e Simpson (2004), portadores de TEA não têm necessidades estáticas, pois suas habilidades de comunicação evoluem. Permitir que o vocabulário evolua junto com a capacidade da criança é fundamental.

1.3 Objetivos

O objetivo geral deste trabalho é implementar uma interface para prover a funcionalidade de customização para um vocabulário de PCA. Segundo a estrutura base do vocabulário definida por Franco et al. (2015), a interface deve ser capaz de representar e customizar o vocabulário. Além dessas funcionalidades, a interface também deve ser fácil de utilizar. Como objetivos específicos, se encontram:

- Definir a arquitetura do software;
- Definir o projeto da interface e suas telas;
- Validar as telas com os usuários;
- Implementar um protótipo funcional;
- Testar funcionalidades do protótipo;
- Comparar funcionalidades implementadas com trabalhos similares; e
- Identificar melhorias e propôr trabalhos futuros baseado nos *feedbacks* colhidos.

1.4 Estrutura

Este trabalho está dividido em três capítulos. No Capítulo 2, é construído o contexto sobre TA e CAA necessário para entender algumas decisões tomadas durante a implementação. Este é detalhada no Capítulo 3, desde a listagem dos requisitos até o projeto e implementação da interface. No Capítulo 4, explicitam-se as principais contribuições e sugestões trabalhos futuros.

2 Comunicação Alternativa

2.1 Tecnologia Assistiva

Tecnologia Assistiva é um termo abrangente que envolve várias áreas de estudo, como acessibilidade, adaptação e reabilitação para pessoas com deficiência (MANZINI, 2005). Segundo Hogetop e Santarosa (2002), um ponto decisivo para classificar TA é a potencialização do indivíduo. Potencialização que significa aumentar a independência para realizar atividades que não seriam possíveis sem assistência externa. Essa independência é importante, porque faz parte da reabilitação do indivíduo. Um estudo sobre ferramentas de TA no contexto da reabilitação cognitiva foi feito por LoPresti, Mihailidis e Kirsch (2004). Uma das conclusões indica que, apesar da baixa adoção, a TA tem um papel significativo no meio reabilitativo.

Com o foco na reabilitação, o uso da TA para auxílio de PNE se torna ainda mais relevante. Um exemplo que evidencia esse ponto é o caso de crianças portadoras do Transtorno do Espectro Autista (TEA). Portadores de TEA possuem necessidades educacionais especiais, e uma questão central da intervenção educacional consiste no desenvolvimento de suas habilidades cognitivas. Segundo a Associação de Amigos dos Autistas[1], essas habilidades incluem: autonomia e independência, comunicação não verbal e habilidades acadêmicas. Como mostrado na seção 2.2, a TA pode auxiliar portadores de TEA em desenvolver essas capacidades.

Vale destacar que TA não se limitam a tecnologias de assistência cognitiva. Filho e Garcia (2012) afirmam que:

> Qualquer ferramenta, adaptação, dispositivo, equipamento ou sistema que favoreça a autonomia, atividade e participação da pessoa com deficiência ou idosa é efetivamente um produto de TA. [...] Existem os produtos denominados de Baixa Tecnologia(*low-tech*) e os produtos de Alta Tecnologia(*high-tech*). Essa diferença não significa atribuir uma maior ou menor funcionalidade ou eficiência a um ou a outro, mas, sim, caracterizar apenas a maior ou menor sofisticação dos componentes com os quais esses produtos são construídos e disponibilizados. (FILHO; GARCIA, 2012)

Em outras palavras, a TA representa um conjunto de técnicas e ferramentas que têm como objetivo auxiliar o deficiente ou o idoso. A fim de ilustrar essa variedade, é apresentada uma lista de exemplos de TA, tanto *high-tech* quanto *low-tech*:

- *high-tech*

[1] <http://www.ama.org.br/site/>

  - Dispositivos para Comunicação Alternativa e Aumentativa;
  - Cadeira de rodas elétrica;
  - Teclados alternativos; e
  - Leitores de telas para pessoas com deficiência visual.

- *low-tech*
  - Bengalas;
  - Lupas;
  - Canetas com formatos especiais;
  - Copos que não viram; e
  - Fitas de velcro.

Para finalizar, é reforçada a relevância de se utilizar a TA como ferramenta potencializadora para crianças portadoras de TEA. Mirenda (2001) fez um estudo que compila várias pesquisas sobre o relacionamento do autismo com TA e CAA (seção 2.2). A maioria dos estudos analisados reportou impacto positivo na qualidade de vida dos portadores de TEA. Essa melhora na qualidade de vida explicita o valor de adotar uma TA no tratamento e acompanhamento desses indivíduos.

2.2 Comunicação Alternativa e Aumentativa (CAA)

Dentro do universo de TA, existe a CAA. Este é definida como uma forma de comunicação além da linguagem oral (GLENNEN; DECOSTE, 1997). Essas formas de comunicação são utilizadas para expressar sentimentos, ideias, necessidades, pensamentos, entre outros. Miranda e ICD (2004) afirmam que a CAA refere-se a qualquer meio de comunicação que substitua ou suplemente os meios habituais. O poder de substituir ou suplementar a comunicação de um indivíduo, faz com que CAA possa ser utilizada como uma ferramenta de tratamento e reabilitação. Por exemplo, um indivíduo que sofreu um trauma físico nas cordas vocais pode utilizar um sistema de síntese de texto para se comunicar. Nesse caso, o sistema de síntese é uma forma de CAA que substitui a fala, restaurando a capacidade do paciente de se comunicar verbalmente.

Essa capacidade de suplementação da CAA, vai além de simplesmente substituir a fala do indivíduo. Em se tratando de certos transtornos psicológicos, prover um meio de comunicação alternativo se prova uma ferramenta útil para os terapeutas e familiares. Schlosser e Wendt (2008) fizeram um revisão de trabalhos sobre a intervenção de CAA em portadores de TEA. Seus resultados indicam que a utilização de CAA não prejudica o desenvolvimento da fala e pode até resultar em melhoras no desenvolvimento.

Um dos tipos de CAA utilizado por portadores de TEA são as Pranchas de Comunicação Alternativa (PCA). As PCA agregam um conjunto de símbolos que, por sua vez, podem ser utilizados como meio de comunicação. De maneira análoga ao alfabeto, esses símbolos podem ser compostos para formar palavras e frases. Segundo Charlop-Christy et al. (2002), um sistema simbólico comumente utilizado em PCA é o *Picture Communication Symmbols* (PCS). O sistema possui figuras simples de fácil reconhecimento, que possibilitam o entendimento por todas as idades. As pranchas de comunicação também podem ser classificadas em *low-tech* e *high-tech*. Pranchas impressas são classificadas como *low-tech* enquanto dispositivos digitas são *high-tech*. Um exemplo de prancha *high-tech* é um aplicativo para *tablet*.

As PCA podem ser utilizadas como ferramenta para auxiliar no desenvolvimento de fala em portadores de TEA (AVILA, 2011). A prancha serve como um meio que auxilia a comunicação nos dois sentidos. Por exemplo, a PCA pode auxiliar tanto a comunicação do portador de TEA com seu terapeuta quanto a comunicação do terapeuta com o seu paciente. Além de potencializar a fala, a PCA também pode prover mais independência ao usuário que, segundo a Associação de Amigos dos Autistas, é central ao tratamento.

Concluindo, a CAA se mostra uma ferramenta capaz de melhorar a qualidade de vida de um grupo significativo de pessoas. Segundo Avila (2011):

> A CAA é importante como um recurso que, quando utilizado com estratégias e técnicas comunicativas, dá a oportunidade ao aluno com Necessidade Especial (PNE) para que se torne autossuficiente em suas situações de comunicação, com isso proporcionando oportunidades de interação com o outro, evitando sua exclusão social e seu isolamento.(AVILA, 2011)

3 Uma interface para personalização de vocabulários para PCA

Este capítulo é destinado à discussão sobre a especificação e implementação técnica da interface. A organização das seções segue o processo de desenvolvimento: do levantamento de requisitos, passando pela fase de projeto, até o detalhamento do software implementado. Por fim, é feita uma listagem de aplicativos similares e uma comparação com o resultado deste trabalho.

3.1 Requisitos

Nesta parte, a especificação do projeto é descrita na forma de requisitos. Descrições de certas características que o software deve possuir, seja uma funcionalidade ou alguma propriedade. Esses requisitos são divididos em *Funcionais* e *Não Funcionais*.

3.1.1 Requisitos funcionais

Os requisitos funcionais de um projeto definem como o software deve operar. Funções básicas são descritas, e, a partir da composição dessas funcionalidades primárias, o comportamento do software é derivado. Requisitos funcionais descrevem apenas funcionalidades. Características como *facilidade de uso*, desempenho, *responsividade*, *consistência* são Requisitos não Funcionais.

Os requisitos iniciais deste projeto foram apresentados pelo orientador. Ao longo da fase de prototipagem e desenvolvimento, novos requisitos surgiram e foram devidamente adicionados a esta lista:

- O software deve ser capaz de fazer requisições POST, PUT e GET para o servidor externo em formato JSON[1];
- O software deve ser capaz de receber o vocabulário em formato JSON do servidor externo;
- O software deve renderizar os ícones de cada elemento em um formato de imagem;
- O software deve ser capaz de mostrar um visão geral do vocabuário em uma *TreeView*[2];

[1] Formato *lightweight* para troca de dados.
[2] Estrutura em árvore similar ao que sistemas operacionais utilizam para mostrar hierarquia de pastas.

- O software deve mostrar um *Breadcrumb*[3];
- O software deve possuir formulários que possibilitem a criação, edição ou remoção de todos os elementos do vocabulário;
- O software deve submeter os formulários em formato JSON; e
- O software deve representar o vocabulário de maneira similar ao *tablet*.

3.1.2 Requisitos não Funcionais

Requisitos não funcionais descrevem certas características chave de um projeto. Como mencionado na introdução desta seção, Requisitos não Funcionais são mais subjetivos que Requisitos Funcionais. Por exemplo, validar que um software faz uma requisição correta para um servidor, pode ser mais fácil que validar se um software é *fácil de usar*. Enquanto verificar a facilidade de uso envolve usuários e uma interpretação não binária das métricas coletadas, verificar se uma requisição ao servidor é feita corretamente pode só envolver um teste unitário.

Os Requisitos não Funcionais deste projeto também foram propostos pelo orientador e coletados ao longo do processo de desenvolvimento:

- O software deve ser fácil de usar;
- O software deve ter aparência moderna;
- O software deve ter aparência similar a ferramentas que os usuários alvo utilizam; e
- O software deve ter uma interface limpa (menor carga cognitiva).

3.2 Projeto da interface

Nesta seção, é descrito o processo que levou à criação do *design* da interface. Nesta fase, não existe nenhum detalhe de implementação tecnológica, apenas considerações sobre os componentes visuais da interface e como o usuário deve interagir com esses componentes. A metodologia usada na construção da interface e os passos que levaram ao resultado final também são descritos.

3.2.1 Metodologia

Utilizou-se uma técnica de desenvolvimento ágil. Dentre outras características, se destaca o fato que utilizando esta técnica, os passos de desenvolvimento não são finais nem

[3] Estrutura que mostra a profundidade de navegação do vocabulário.

estáticos. Por exemplo, a fase *Projeto de Interface* não acaba depois que o desenvolvimento chega à fase de *Implementação de Interface*. A implementação pode fornecer *inputs* que são utilizados para projetar novos componentes. Em outras palavras, o desenvolvimento não ocorre de forma linear. A falta de linearidade é importante porque o *feedback* é colhido durante todo o processo. Caso seja detectada a necessidade de alguma mudança, não há motivo de recomeçar o ciclo todo de desenvolvimento em *Requisitos*. Basta retornar a fase relevante e retomar o projeto.

Outra vantagem do desenvolvimento ágil é o incentivo à prototipagem e validação com os usuários. *Protótipo* é um modelo simplificado que tem como objetivo validar alguma funcionalidade com usuário. Além de poder ser construído rapidamente, é mais vantajoso descartar um protótipo que não foi bem aceito do que descartar a funcionalidade já implementada no software.

A prototipagem foi utilizada como base do projeto da interface. Definiu-se um ciclo de *feedback*: cada versão foi validada com os usuários, e as melhoras foram adicionadas de maneira incremental até que a versão final da interface foi definida. A utilização dessa metodologia se provou eficaz no quesito tempo. Como foi mostrado, da primeira versão até a versão final houve várias mudanças. Mudanças que custariam mais tempo se tivessem sido feitas com o software funcionando e implementado.

As validação dos protótipos e do software implementado foi feita com a equipe Assistive: Robson Fidalgo (Doutor, trabalha com engenharia de software, banco de dados e TA), Edson Alves (Doutorando, trabalha com engenharia de software, banco de dados e TA), Natália Franco (Doutoranda, trabalha com engenharia de software e TA) e Tiago Lima (graduando de engenharia da computação, trabalha com engenharia de software, banco de dados, e TA). Para colher o *feedback* não foi utilizado nenhum questionário. O software ou protótipo é apresentado, e os comentários são colhidos à medida que os usuários encontram possíveis problemas.

3.2.2 Protótipos

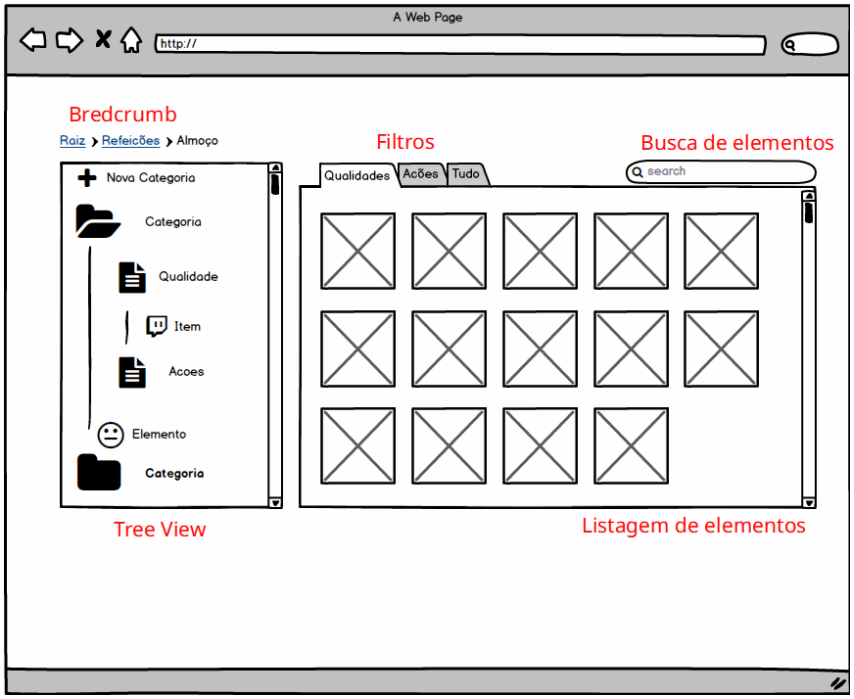
Durante este processo foram utilizados protótipos de baixa fidelidade. Os protótipos foram criados utilizando uma ferramenta especializada[4]. O uso dessa ferramenta possibilitou a criação eficiente e também proveu flexibilidade para realizar as mudanças requisitadas pelos usuários. Os protótipos apresentados não são interativos, durante a validação é explicado para o usuário como cada elemento se comporta.

O primeiro protótipo (Figura 1, Figura 2 e Figura 3), foi baseado nos requisitos do projeto e nas necessidades conhecidas pelo usuário. O objetivo deste protótipo é englobar todas as funcionalidades do projeto, não necessariamente de forma ideal. Essa abordagem

[4] Balsamiq <https://balsamiq.com>

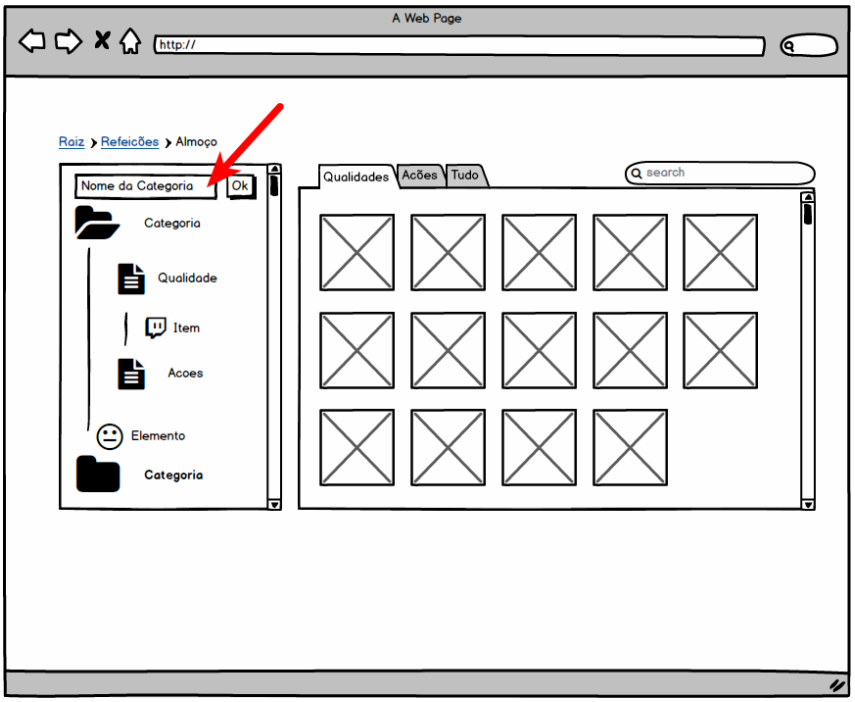
foi utilizada para colher o *feedback* do usuário o mais rápido possível.

Figura 1 – Protótipo 1, visão geral



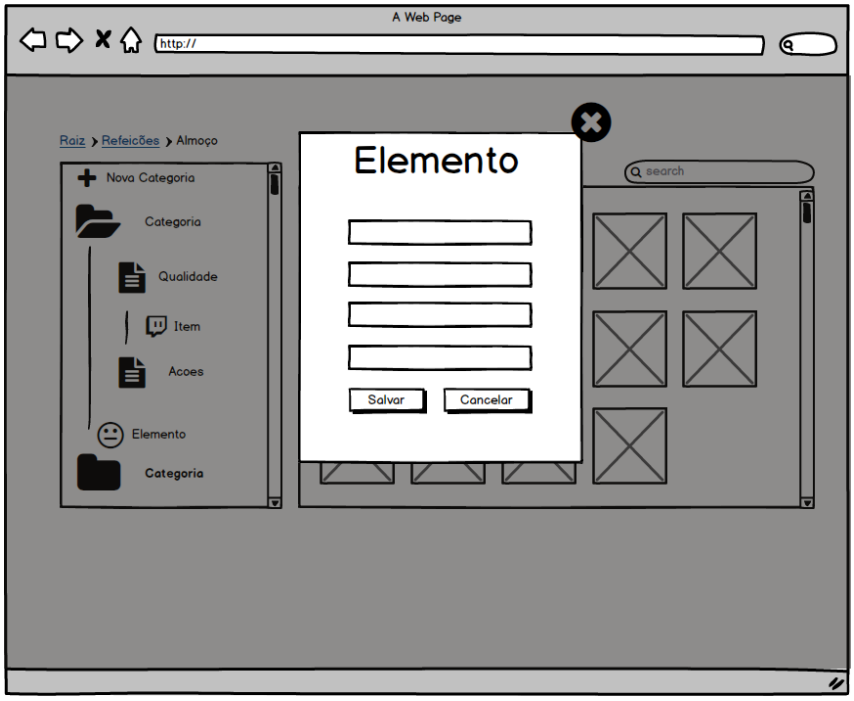
Fonte: o Autor (2016)

Figura 2 – Protótipo 1, adicionando nova categoria



Fonte: o Autor (2016)

Figura 3 – Protótipo 1, adicionando novo elemento



Fonte: o Autor (2016)

A interface possui componentes individuais que preenchem requisitos específicos do projeto.

- Lado esquerdo da Figura 1: *Tree View* Contento uma visão geral do vocabulário;
- Topo do lado esquerdo da Figura 1: *Breadcrumb* representa a raiz atual da navegação no vocabulário;
- Componente ao lado direto da Figura 1: listagem de elementos na categoria atual;
- Canto direito superior da Figura 1: busca de elementos. A busca é refletida na listagem atual; e
- Abas sob a listagem de elementos da Figura 1: filtram a listagem para mostrar elementos, ações ou qualidades.

Cada componente tem um propósito bem definido. A funcionalidade de navegação eficiente é provida por meio da *Tree View*. O conteúdo da categoria é representado por meio da listagem de elementos. A filtragem é fornecida pelas abas sob a listagem, e o *Breadcrumb* oferece uma âncora a mais para o usuário. Essa propriedade de âncora acontece porque é mais um local que mostra a categoria atual.

Apesar de cada componente ter sido implementado com base nas necessidades do usuário, não há garantia de que a racionalização utilizada para posicionar cada elemento

seja a mesma esperada por eles. A partir dessa premissa, o primeiro Protótipo foi sujeito a uma sessão de feedback, e as seguintes observações foram feitas[5]:

- *Tree View* confusa. Além de categorias, são mostrados elementos, qualidades e ações;
- Comportamento da *Tree-View* não está explícito e foi questionado que interações são necessárias para se expandir ou visitar uma categoria;
- O comportamento de adicionar nova categoria não está consistente com o de adicionar novo elemento; e
- A presença do campo de busca sob a listagem pode não ser necessária, dado que vocabulários tendem a ter poucos itens.

Baseado na lista de *feedbacks*, uma lista de melhorias foi compilada a ser introduzida no Protótipo 2. O critério decisivo para introduzir uma mudança ao protótipo é que essa mudança não deve remover ou adicionar um requisito funcional ao projeto. Esse critério existe a fim de fechar o escopo do projeto. Escopo que engloba somente a interface de customização.

O Protótipo 2, apresentado na Figura 4, não endereçou todos os questionamentos do usuário. O argumento por trás desta decisão é reduzir o ruído entre uma versão e outra. *Ruído* sendo a quantidade de mudanças feitas entre a versão passada e a versão atual. Essa estratégia permite validar as mudanças com maior precisão, já que a maioria da interface continua familiar ao usuário. Além da precisão, existe a vantagem de colher uma opinião sobre componentes já apresentados na versão passada. Apresentar esses componentes mais de um vez, permite validar duplamente se a opinião sobre o elemento se mantém.

A melhoria escolhida para o Protótipo 2, foi a *Tree View*. A partir do *feedback* dado sobre a primeira versão do protótipo, identificou-se que o maior o problema da interface era a ambiguidade desse elemento. Com o objetivo de tornar a funcionalidade da *Tree View* mais explícita, o elemento foi simplificado em duas mudanças principais:

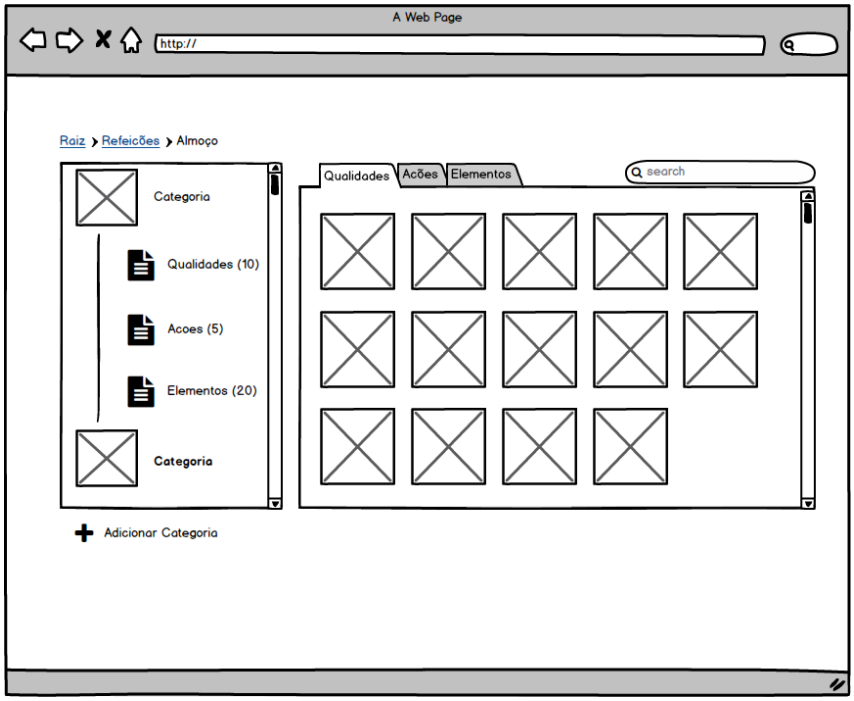
- Os filhos de uma categoria na *Tree-View* são filtros, que funcionam de forma igual às abas sob a listagem de itens; e
- O ícone que representa uma categoria não é mais uma pasta genérica, é a imagem do vocabulário associada àquela categoria.

O resto da funcionalidades descritas para o Protótipo 1 não foram alteradas.

Sob validação, o Protótipo 2 foi bem recebido. Notou-se que os usuários ficaram menos confusos em relação à *Tree View*. A adição da imagem como ícone da categoria foi

[5] Usuários fizeram outros comentários, mas só está listado o que foi considerado relevante.

Figura 4 – Segundo Protótipo



Fonte: o Autor (2016)

bem aceita, mas a remoção dos filhos da categoria foi questionada inicialmente. A remoção foi explicada aos usuários após a validação, e atingiu-se o consenso de que a mudança foi positiva.

Baseado na reação positiva dos usuários, o ciclo de *feedback* foi encerrado. Em outras palavras, o protótipo 2 foi a base referencial para implementação da interface. O processo de implementação é descrito na próxima seção.

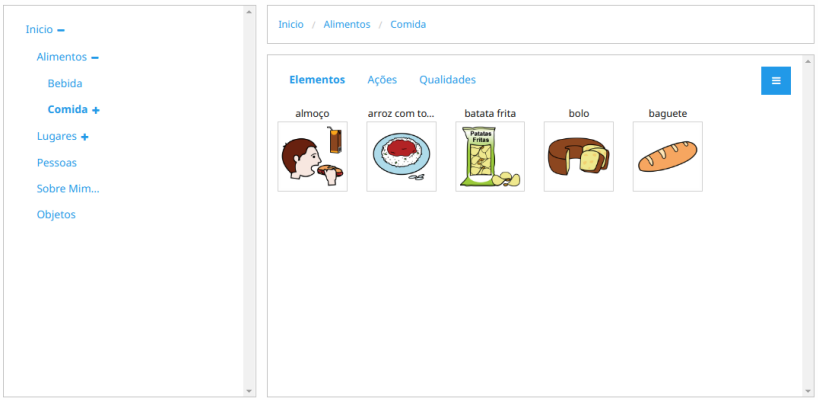
3.3 Implementação da interface

Esta seção descreve o processo de implementação da interface gráfica, desde as decisões sobre os paradigmas utilizados até a modelagem do software que foi implementado. O objetivo é detalhar a última parte do processo de desenvolvimento que resultou no produto funcional(Figura 5).

3.3.1 Arquitetura

A arquitetura de um software descreve a estrutura e as disciplinas utilizadas na construção do projeto. Essa descrição tem um caráter de alto nível e não entra em detalhes de implementação. A arquitetura se distancia desses detalhes com o objetivo de prover o entendimento sobre o funcionamento do software de maneira agnóstica para as tecnologias utilizadas. Distanciar-se dos detalhes também é importante porque reduz a quantidade

Figura 5 – Interface Implementada - Visão geral



Fonte: o Autor (2016)

de informação sem sacrificar o objetivo, que é prover uma visão geral sobre a estrutura do software. A fim de prover essa visão geral, esta seção descreve os pontos arquiteturais mais importantes. O paradigma de programação utilizado, as estruturas de dados e um diagrama de estados são exemplos desses pontos arquiteturais.

O paradigma adotado para este projeto é funcional, mais especificamente foi utilizada uma abordagem chamada de *Functional Reactive Programming* (FRP). *FRP* é um paradigma de programação declarativo que interpreta variáveis de uma maneira diferente da programação imperativa tradicional. A mutabilidade dos valores é vista como um cidadão de primeira classe, e o programa é construído em cima desse conceito. Em outras palavras, essas variáveis são interpretadas como sinais. Esse paradigma é particularmente útil em GUI, porque o estado da interface depende de valores dinâmicos e das interações com o usuário. Cliques, a posição do mouse, novas respostas do servidor e campos de busca são exemplos de interações do usuário. FRP provê as abstrações necessárias para lidar com esse tipo de eventos de uma maneira sensata. Czaplick (2012) descreve a implementação de uma linguagem para a web baseada nos conceitos da FRP.

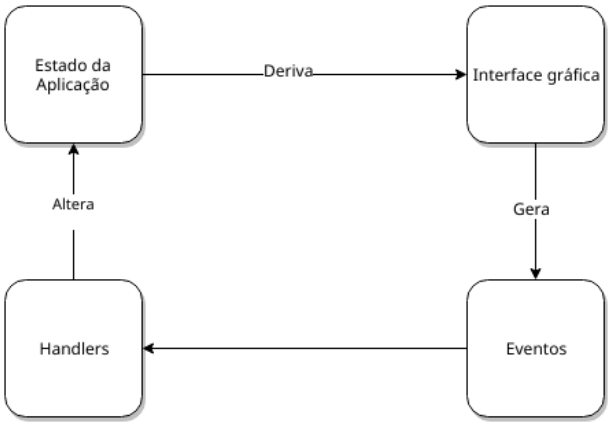
Baseado no paradigma FRP, o funcionamento do software é descrito a partir de um estado inicial e eventos que transformam esse estado (Figura 6). Ou seja, a interface renderizada para o usuário é um reflexão do estado do programa. Se o estado muda, a interface é automaticamente atualizada. Baseado nesses conceitos, descrever a arquitetura deste projeto se resume a detalhar como o estado do programa é guardado, como esse estado sofre mutações e como esse estado é renderizado na forma da interface para o usuário.

Dado que o paradigma adotado é funcional, o estado do programa não é representado por um objeto complexo mas, sim, por uma estrutura de dados. A estrutura utilizada é uma *Key-Value Store*, nas quais chaves são associadas a valores. Todo o estado do programa é

armazenado nesse banco de dados simplificado: Categorias, elementos, qualidades, ícones, *forms*, entre outros. Detalhar toda a estrutura não faz parte do escopo deste trabalho, mas um exemplo simplificado é fornecido na subseção 3.3.2.

A mutação do estado do programa se dá por meio do lançamento de eventos e é ilustrada na Figura 6. Eventos são lançados e posteriormente capturados por *handlers* que têm a responsabilidade de mudar o estado do programa. É importante destacar que a única entidade capaz de mudar o estado do programa são os *handlers*. Essa filosofia é importante, porque fornece previsibilidade para o comportamento do software. A previsibilidade por sua vez, facilita o desenvolvimento e pode reduzir o número de *bugs* devido à centralização dos efeitos colateiras. Destacar todos os eventos lançados e seus *handlers* também não faz parte do escopo deste trabalho, mas um exemplo é fornecido na subseção 3.3.2.

Figura 6 – Diagrama de eventos e *handlers*

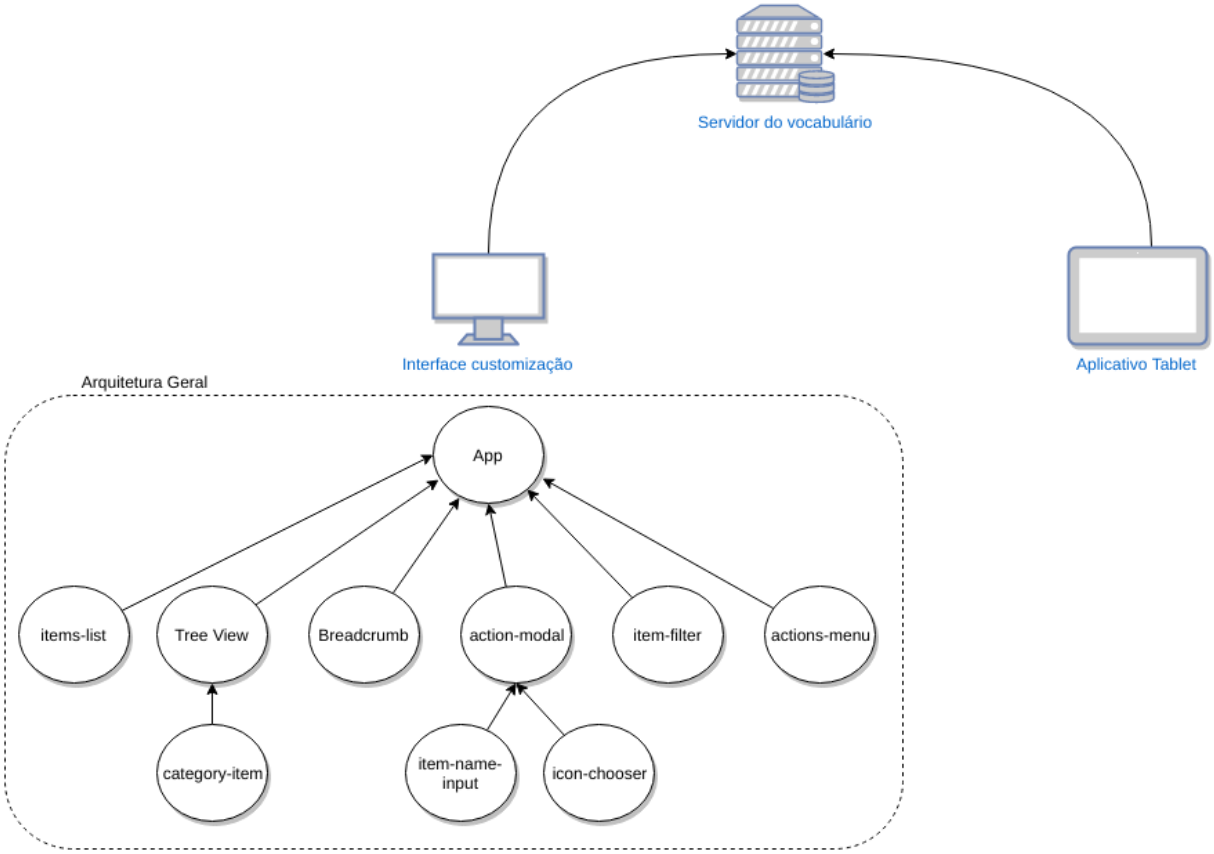


Fonte: o Autor (2016)

Outra parte da arquitetura é descrever como o estado do programa é transformado na interface gráfica. A abordagem utilizada consistiu em dividir a interface em componentes atômicos e representar esses componentes por meio de funções. Essas funções descrevem a estrutura do componente em termos de certas variáveis presentes no estado do programa. O conceito central dessas funções, é a sua conexão com valor associado. À medida que os valores associados sofrem mutações, as funções são executadas com os novos valores. Ao serem executadas com os novos valores, a estrutura representada também muda, e assim, a interface é atualizada.

Na seção 3.1 são mencionados alguns requisitos relacionados à comunicação com servidores. De uma maneira geral, esses requisitos fornecem a funcionalidade de sincronização com a aplicação cliente que roda em um *tablet*. Sincronização que automaticamente torna disponível as mudanças feitas no vocabulário para o aplicativo. Na Figura 7 é ilustrado esse comportamento: o servidor guarda e fornece o modelo do vocabulário que é consumido por ambas aplicações. Esse servidor também é notificado das mudanças realizadas no vocabulário para que possam ser persistidas e assim fornecidas à aplicação cliente. Vale

Figura 7 – Arquitetura



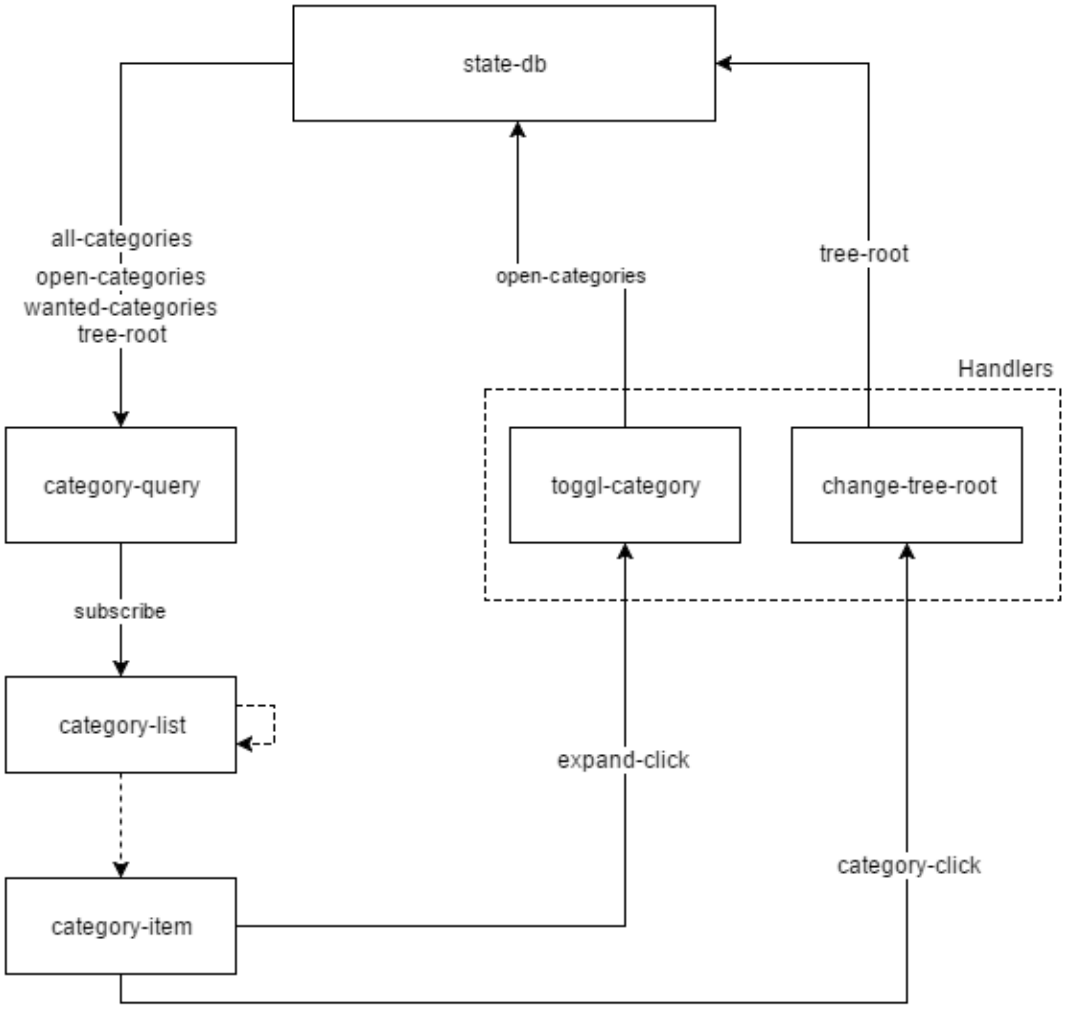
Fonte: o Autor (2016)

destacar que não há comunicação direta entre o software implementado e o aplicativo cliente.

3.3.2 Detalhamento de um componente

Esta subseção é dedicada a ilustrar uma fatia da arquitetura geral do projeto. Isolou-se um componente da interface, e a suas funcionalidades são dissecadas em termos do estado que as representa e os eventos lançados por esse componente.

Figura 8 – *Tree View* - Diagrama de estados.



Fonte: o Autor (2016)

O componente escolhido é a *Tree View* (Figura 18). A escolha se dá pelo fato de que esse componente depende de mais de uma variável e lança mais que um evento. A *Tree View* fornece uma visão geral do vocabulário e também provê um meio de navegação. O diagrama de estados desse componente é fornecido na Figura 8.

Na Figura 8, *category-list* é a *Tree View*. Ela "escuta" a *category-query* que retorna todas as informações necessárias para renderizar a *Tree View*. A *category-query* usa *all-categories*, *open-categories*, *wanted-categories* e *tree-root* para construir a estrutura de dados que a *Tree View* utiliza. A *category-list* tem uma linha pontilhada para ela mesmo porque a estrutura é recursiva, quando uma categoria é aberta na *Tree View*, a mesma

função é chamada com outra raiz. *Category-item* é o item que mostra o nome da categoria junto à opção de abrir (“+”) ou fechar ela (“-”). Esse elemento pode gerar dois eventos, *expand-click* e *category-click*. *expand-click* acontece quando o usuário clica em “+” ou “-”, e *category-click* acontece quando usuário clica sobre o nome da categoria. Esses eventos são capturados pelos *handlers* que, por sua vez, mudam o *state-db*. Por exemplo, se o usuário clica no "+"para abrir uma categoria, essa categoria é adicionada à lista de *open-categories*. Como o *open-categories* mudou, a *category-query* é atualizada com os dados novos, e a *Tree View* é renderizada novamente com essa categoria expandida.

Os outros componentes deste projeto seguem o mesmo principio descrito nesta subseção. A partir do estado armazenado são derivadas as informações necessárias e o componente é renderizado na interface gráfica. Esses componentes também lançam eventos que alteram o estado do programa fazendo com que essas mudanças sejam refletidas na interface.

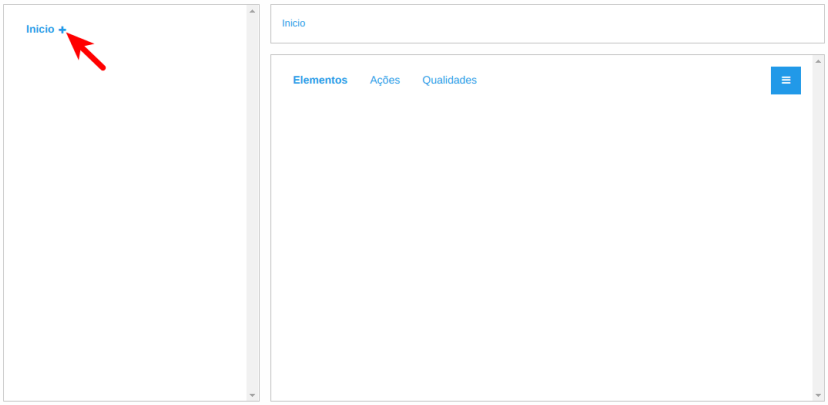
3.4 Manual de utilização

O objetivo desta seção é expor as funcionalidades implementadas, na forma de um manual de uso. Cada subseção detalha uma função ou uma característica e o como ela deve ser utilizada. Funcionalidades similares não serão repetidas. Por exemplo, a criação e edição dos elementos utilizam os mesmos componentes de interface, seria redundante repetir todas as instruções para esse caso.

3.4.1 Navegando pelo vocabulário

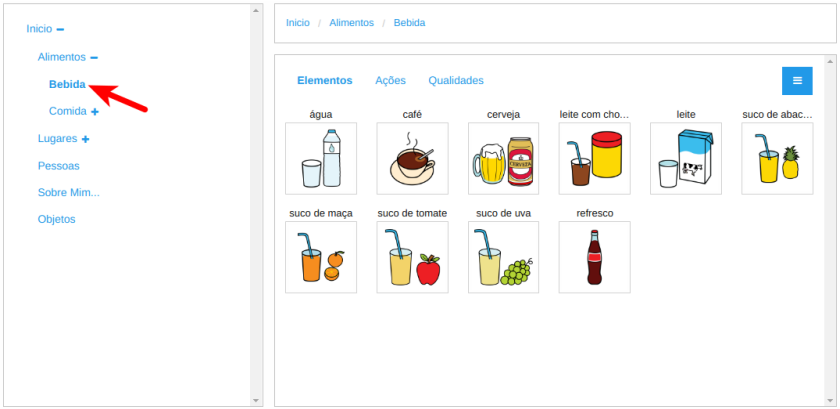
Navegar pelo vocabulário significa ter uma visão geral dos seus componentes junto a suas hierarquias. O software possui dois componentes que provêm funcionalidades de navegação: O *Breadcrumb* e a *Tree View*. Ao abrir um vocabulário pela primeira vez, todos os itens da *Tree View* estarão fechados e o *Breadcrumb* mostrará *Inicio*. Para visualizar que categorias estão presentes neste vocabulário, o usuário deve clicar sobre o “+” ao lado de *inicio* na *Tree View* (Figura 9). Se o usuário desejar, ele pode abrir mais categorias para ter uma noção da hierarquia geral do vocabulário. É importante destacar que para visualizar que componentes estão presentes em uma dada categoria, é preciso clicar sobre o nome da categoria na *Tree View*, essa ação é chamada de *visitar uma categoria* (Figura 10). Ao visitar uma categoria, o *Breadcrumb* mostrará o caminho atual do inicio até a essa categoria e a listagem de elementos mostrará os elementos pertencentes a essa categoria. A funcionalidade de visitar uma categoria também é fornecida pelo *Breadcrumb* e tem o mesmo efeito.

Figura 9 – Navegação - expandir uma categoria



Fonte: o Autor (2016)

Figura 10 – Navegação - visitar uma categoria

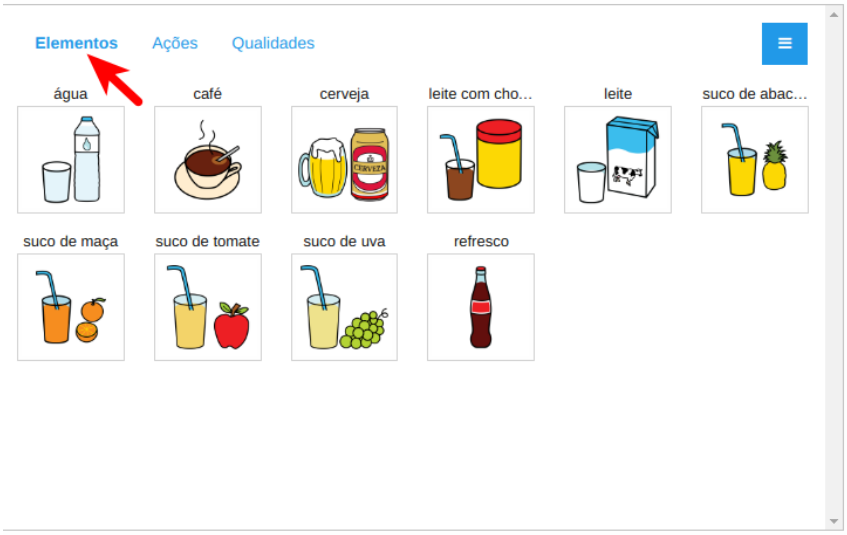


Fonte: o Autor (2016)

3.4.2 Filtragem de itens pertencentes a categorias

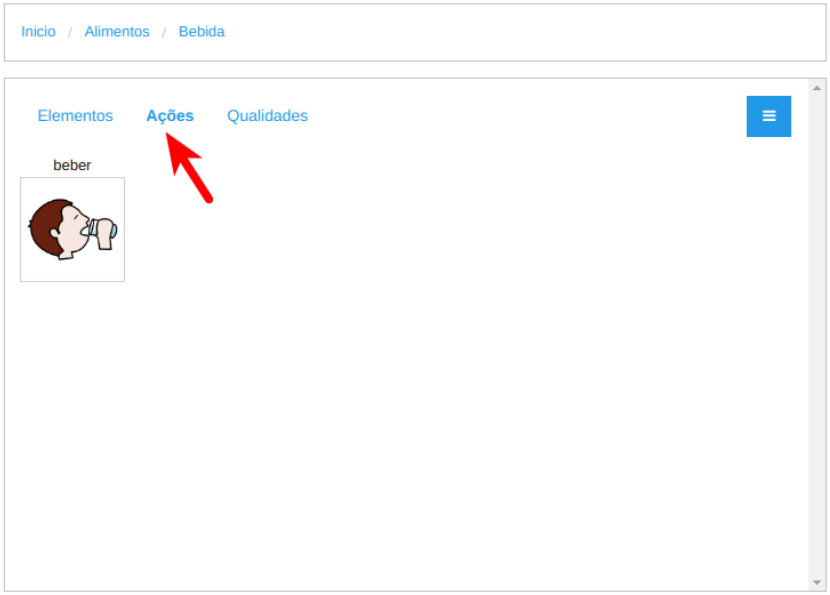
As categorias possuem três tipos de itens: elementos, ações e qualidades. Ao visitar uma categoria, o filtro *elementos* é ativo por padrão (Figura 11). Caso o usuário deseje visualizar outro tipo de item, é preciso clicar sobre o filtro desejado (Figura 12). É importante explicitar que os filtros são ativos para a categoria atual, já que os elementos mostrados pertencem a ela.

Figura 11 – Filtro padrão - Elementos



Fonte: o Autor (2016)

Figura 12 – Filtros - Ações

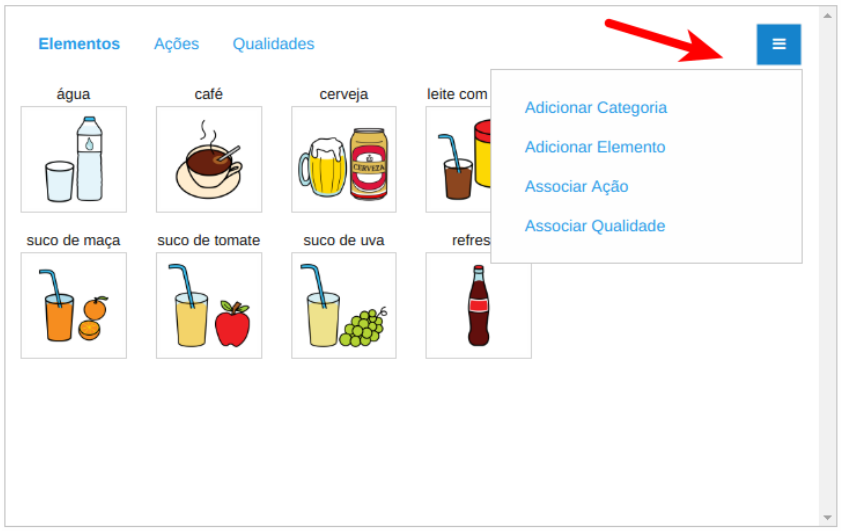


Fonte: o Autor (2016)

3.4.3 Criando novos itens

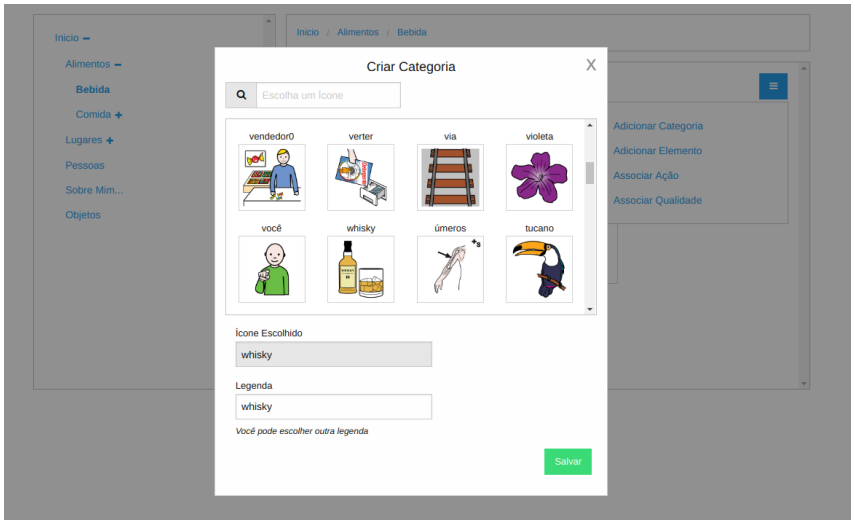
Para criar novos elementos, categorias, ações ou qualidades, é preciso escolher a opção desejada no menu (Figura 13). Escolhida a opção, o usuário precisa decidir que ícone e legenda serão associadas ao novo item (Figura 14). O item é criado sob contexto da categoria atual no vocabulário. Ou seja, se a categoria atual é *Bebida*, o item é automaticamente associado a essa categoria.

Figura 13 – Menu - Customização do vocabulário



Fonte: o Autor (2016)

Figura 14 – Criação de item - Formulário



Fonte: o Autor (2016)

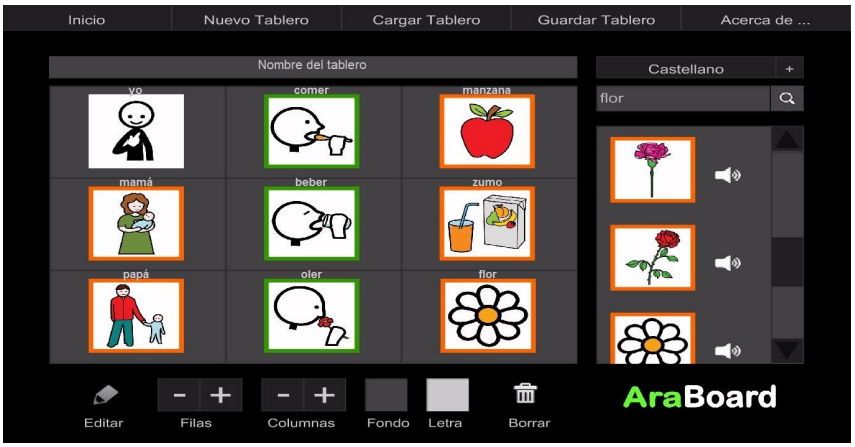
Todas as ações de criação e edição seguem esse mesmo processo e utilizam os mesmos componentes. As diferenças não são suficientes para justificar uma seção dedicada para a criação de cada tipo de item.

3.5 Trabalhos relacionados

O critério para escolher trabalhos relacionados foram softwares de CAA que utilizam recursos gráficos do ARASAAC[6] e também são aplicativos *mobile*. Nesta seção é feita uma compilação desses aplicativos e suas funcionalidades mais importantes. Na seção 3.6, é feita a comparação com as funcionalidades implementadas neste trabalho.

3.5.1 AraBoard

Figura 15 – AraBoard - tela de customização



Fonte: <https://sourceforge.net/projects/ara-board/>

O AraBoard[7] (Figura 15) é um software criado pela universidade Zaragoza que permite a customização e visualização de PCA. Suas funcionalidades incluem:

- Customizar o tamanho do *grid* da prancha de comunicação;
- Pesquisar ícones do ARASAAC;
- Adicionar legenda para um item;
- Adicionar qualquer imagem ou tirar uma foto para ser utilizada como item;
- Adicionar ou gravar um áudio para um item;
- Exportar vocabulário para um arquivo; e
- Importar vocabulário de um arquivo.

Figura 16 – LetMeTalk - visão geral

Fonte: <http://www.letmetalk.info/>

3.5.2 LetMeTalk

LetMeTalk[8] (Figura 16) é um aplicativo mobile que permite a edição do vocabulário na mesma interface de visualização. Suas funcionalidades incluem:

- Criação de categorias;
- Adicionar e editar legendas;
- Mudar a cor de fundo de um item;
- Pesquisar itens do ARASAAC;
- Mudar a posição dos itens;
- Compor itens em uma frase;
- Utilizar vozes;
- Utilizar fotos e imagens para representar um item; e
- Permitir o compartilhamento de vocabulários.

3.5.3 AAC Speech Communicator

Aplicativo mobile[9] que fornece um vocabulário pré-construído e categorizado para criação de frases. Suas funcionalidades incluem:

6 <http://arasaac.org/>
7 <http://giga.cps.unizar.es/affectivelab/araboard.html>
8 <http://www.letmetalk.info/>
9 <http://aacspeech.org/>

Figura 17 – AAC speech communicator - visão geral

Fonte: <https://play.google.com/store/apps/details?id=com.epfl.android.aac_speech&hl=pt_BR>

- Gerar frases em linguagem natural a partir da composição dos itens;
- Customização do gênero do usuário;
- Destacar itens recentemente utilizados;
- Classificação dos itens por cor; e
- Classificação de itens baseado em ações, nomes, descritivos, entre outros.

3.6 Análise comparativa

Para possibilitar a comparação com os trabalhos listados, foram identificadas as dimensões relevantes: características e funcionalidades que se destacaram como diferencial entre os aplicativos apresentados e a interface implementada neste trabalho. Partindo do princípio que foi implementada uma interface para customização do vocabulário, o foco da comparação são as funcionalidades relacionadas à criação e à manipulação do vocabulário.

As dimensões escolhidas foram as seguintes:

- Capacidade de adicionar legenda a um item;
- Capacidade de adicionar imagens a um item;

- Capacidade de criar categorias e adicionar itens a essas categorias;
- Categorização de sugestões para um item;
- Capacidade de alterar a cor de fundo de um item;
- Capacidade de gravar áudio para um item;
- Capacidade de utilizar áudio sintetizado para um item;
- Capacidade de customizar o *grid* do vocabulário; e
- Capacidade de compartilhar o vocabulário.

Essas dimensões foram adicionadas a um quadro (Quadro 1) comparativo onde é possível sintetizar o relacionamento entre os trabalhos apresentados.

Quadro 1 – Quadro comparativo

Funcionalidade	Este trabalho	AraBoard	LetMeTalk	AAC Speech Communicator
Adicionar legenda	Sim	Sim	Sim	Não
Adicionar imagem	Não	Sim	Sim	Não
Criar categorias	Sim	Não	Sim	Não
Categorizar sugestões	Sim	Não	Não	Não
Alterar cor de fundo	Não	Sim	Sim	Não
Gravar áudio para um item	Não	Sim	Sim	Não
Utilizar áudio sintetizado	Sim	Sim	Sim	Não
Customizar *grid*	Não	Sim	Não	Não
Compartilhar vocabulário	Sim	Sim	Sim	Não

Fonte: o Autor (2016)

Ao analisar o Quadro 1, é possível extrair algumas conclusões. Por exemplo, o AAC Speech Communicator está marcado como negativo para todas as dimensões. Isso acontece, porque esse software não possui nenhuma funcionalidade de customização. Outro ponto é a ausência do suporte para criação de categorias no AraBoard. Esse tal quesito é notável, porque a categorização é uma funcionalidade central neste trabalho e nos outros trabalhos comparados. Por fim, o software que se mostrou mais completo foi o LetMeTalk. Somente duas funcionalidades estão marcadas como negativas: a *customizar grid* e a *categorizar sugestões*.

Ao comparar este trabalho aos similares, nota-se que algumas funcionalidades não estão presentes. Adicionar imagens customizadas, alterar o tamanho do *grid* e a possibilidade de gravar áudio não foram implementadas. Baseado na prioridade e na dificuldade de implementação, foi uma decisão de projeto não incluir essas funcionalidades. O tempo economizado foi dedicado à implementação de funcionalidades mais críticas,

como a sincronização com o servidor e a listagem de elementos. Sob uma visão geral, a interface implementada se mostra tão completa quanto a dos similares.

4 Conclusão

Neste capítulo, é apresentando o resultado da implementação do software. O resultado inclui as funcionalidades implementadas e o *feedback* sobre o trabalho de uma forma geral. As principais funcionalidades são descritas juntamente com o seu valor para o usuário. Em outras palavras, as principais contribuições são listadas. Além da listagem, também é descrito que funcionalidades podem ser inclusas na forma de trabalhos futuros. Esses trabalhos são melhorias e mudanças ao software implementado que têm potencial para melhor servir os objetivos apresentados na seção 1.3.

4.1 Funcionalidades e principais contribuições

Como exposto na seção 1.3, um dos principais objetivos deste software é fornecer uma interface fácil de usar que tem a capacidade de customizar o vocabulário da PCA. Dessa maneira, as contribuições deste trabalho são apresentadas sob o contexto desses objetivos. Ou seja, é explicitada a contribuição da funcionalidade para a facilidade de uso e para a customização do vocabulário. As funcionalidades descritas são:

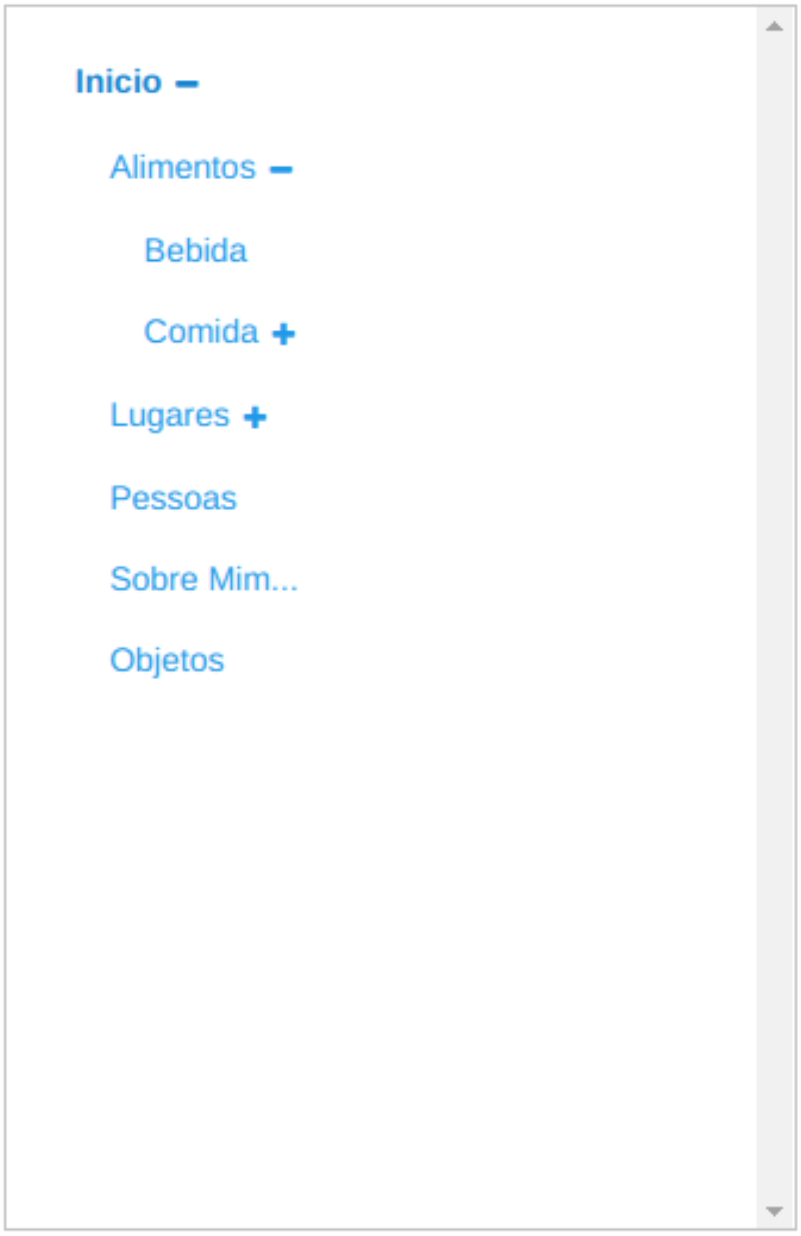
- *Tree View*;
- *Breacrumb*;
- Listagem de itens;
- Modal de criação e edição de elementos; e
- Listagem de ícones junto à busca.

A *Tree View* ilustrada na Figura 18 é um elemento central da navegação do vocabulário. Esse elemento contribui para a facilidade de uso, porque é algo familiar aos usuários. É um elemento utilizado na navegação de pastas e arquivos nos principais sistemas operacionais. Partindo dessa familiaridade, o usuário interage com vocabulário da mesma maneira que ele interage com o sistema de arquivos.

O *Breacrumb* (Figura 19) também é um elemento de navegação. Além da navegação, o objetivo do *Breacrumb* é deixar explícito em que categoria o usuário está navegando no momento. Explicitar essa posição é importante para que o usuário não se perca ao explorar o vocabulário.

A listagem dos itens (Figura 20) é o elemento principal para o usuário inspecionar o conteúdo do vocabulário. Nessa listagem, também é possível editar os itens. Assim,

Figura 18 – *Tree View*



Fonte: o Autor (2016)

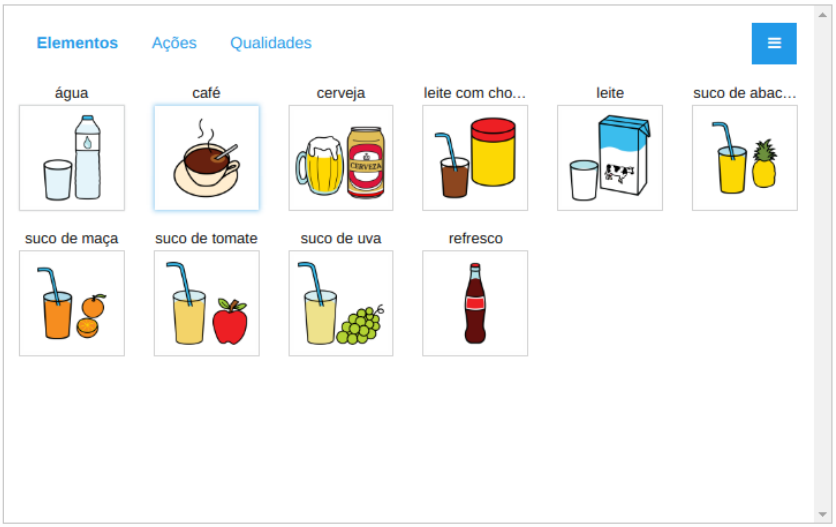
esse elemento é tanto uma ferramenta de navegação quanto de edição. Juntar essas funcionalidades se prova importante porque o usuário espera que isso aconteça. Novamente em analogia ao sistema de pastas de sistemas operacionais, é esperada uma ação ao clicar sobre um item.

A modal (Figura 21) de edição e criação é o elemento central da customização do vocabulário. Utilizando esse elemento, é possível adicionar novos itens e categorias. Também é possível editar os itens já criados. A modal funciona como uma janela de diálogo que permite o usuário a realizar a ação desejada. Dentro da modal, encontra-se um elemento de busca de ícones (Figura 22). A busca é útil, dado que existem mais de

Figura 19 – *Breadcrumb*

Fonte: o Autor (2016)

Figura 20 – Exemplo da listagem de itens da categoria *Bebida*



Fonte: o Autor (2016)

7000 ícones disponíveis.

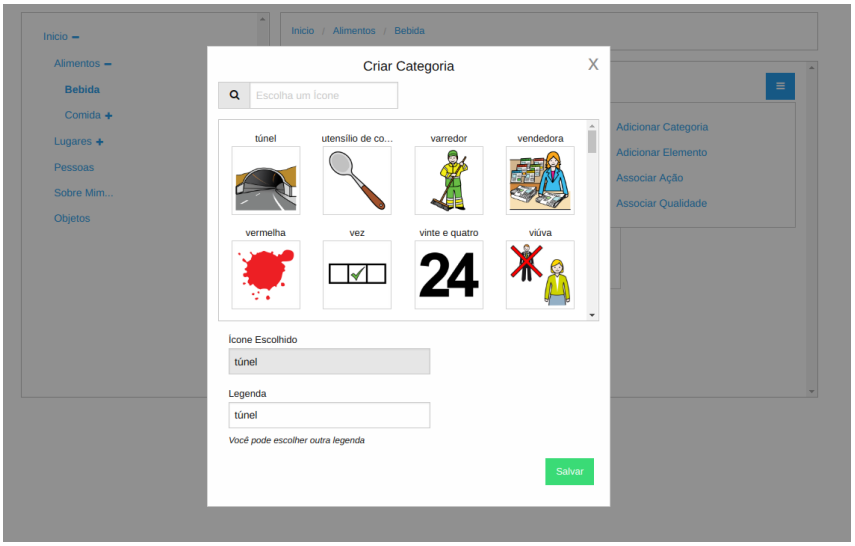
Uma das principais contribuições deste trabalho é a criação de uma interface que foi bem aceita pelos usuários. A fácil utilização foi conquistada a partir do paralelo entre um vocabulário de PCA e um sistema de arquivos. Esse paralelo faz com que não seja necessário introduzir o usuário a novos conceitos, basta um conhecimento técnico básico.

Outra contribuição importante é a sincronização com o servidor. Ao editar o vocabulário, as mudanças são automaticamente salvas. Dessa maneira, o novo vocabulário se torna disponível ao *tablet* assim que houver uma conexão com a Internet. Essa funcionalidade é considerada importante porque não está presente em nenhum dos trabalhos comparados na subseção 3.3.2.

4.2 Trabalhos futuros

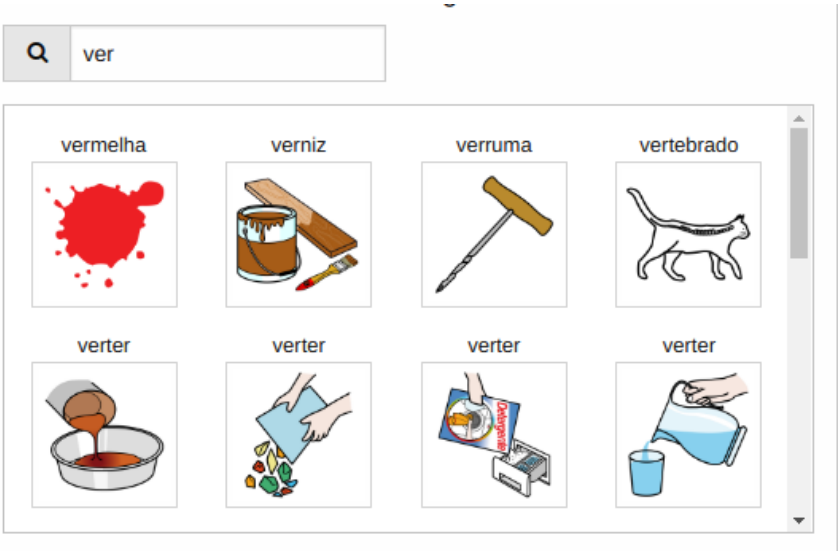
Baseado no *feedback*, a primeira melhoria proposta é adicionar uma funcionalidade que permita customizar o posicionamento dos elementos na prancha. Essa capacidade tem valor porque o usuário da prancha tende a se familiarizar com o posicionamento dos elementos. A medida que o vocabulário sofre alterações, é preciso posicionar os elementos de maneira a diminuir o ruído.

Figura 21 – Exemplo da modal de edição, criação de categoria



Fonte: o Autor (2016)

Figura 22 – Exemplo de busca de ícones, utilizando o termo *ver* na barra de busca



Fonte: o Autor (2016)

Outra funcionalidade proposta é a possibilidade de visualizar todo o vocabulário em um tipo de mapa mental. A utilidade dessa visualização é fornecer, ao usuário, uma visão geral do vocabulário da prancha. Dessa maneira, é possível extrair certas métricas de qualidade do vocabulário. Métricas que podem possibilitar sua melhoria, sob o ponto de vista da eficácia e aceitação do usuário da prancha.

Uma melhora não relacionada a funcionalidades é validar o software com profissionais de saúde e parentes de portadores de TEA. Como mencionado na subseção 3.2.1, o grupo de validação foi a equipe *Assistive*. Pessoas que possuem experiência na área de TA, mas que podem não representar as expectativas do público alvo deste trabalho. Realizar

validações com o grupo de usuários esperado para o software implementado é importante porque pode revelar novas necessidades e problemas.

Por fim, é sugerida a criação de uma rede social na qual os usuários possam compartilhar seus vocabulários criados. Nessa aplicação, a interface implementada neste trabalho seria apenas um elemento. Os outros componentes poderiam prover funcionalidades de compartilhamento, de categorização e de comunicação entres os usuários. Por exemplo, o pai de uma criança portadora de TEA poderia compartilhar o vocabulário criado com o terapeuta. Este, por sua vez, poderia fazer modificações e sugestões baseadas nas necessidades dos seus pacientes. A criação dessa rede social demanda um esforço que pode envolver vários trabalhos futuros, com cada um podendo abordar uma funcionalidade específica.

Referências

AVILA, B. G. Comunicação aumentativa e alternativa para o desenvolvimento da oralidade de pessoas com autismo. *Universidade Federal do Rio Grande do Sul. Faculdade de Educação. Programa de Pós-Graduação em Educação*, 2011. Citado na página 15.

BERSCH, R. Introdução à tecnologia assistiva. *Porto Alegre: CEDI*, 2008. Citado na página 9.

CHARLOP-CHRISTY, M. H. et al. Using the picture exchange communication system (pecs) with children with autism: Assessment of pecs acquisition, speech, social-communicative behavior, and problem behavior. *Journal of applied behavior analysis*, Wiley Online Library, v. 35, n. 3, p. 213–231, 2002. Citado na página 15.

CZAPLICK, E. Elm: Concurrent frp for functional guis. *HARVAD - School of Engineering and Applied Sciences*, 2012. Citado na página 24.

DAMASCENO, L. L.; FILHO, T. A. G. As novas tecnologias como tecnologia assistiva: utilizando os recursos de acessibilidade na educação especial. In: *CONGRESSO IBERO-AMERICANO DE INFORMÁTICA NA EDUCAÇÃO ESPECIAL (CIIEE)*. [S.l.: s.n.], 2002. v. 3. Citado na página 9.

FALLON, K.; LIGHT, J.; ACHENBACH, A. The semantic organization patterns of young children: Implications for augmentative and alternative communication. *Augmentative and Alternative Communication*, Taylor & Francis, v. 19, n. 2, p. 74–85, 2003. Citado na página 10.

FALLON, K. A.; LIGHT, J. C.; PAIGE, T. K. Enhancing vocabulary selection for preschoolers who require augmentative and alternative communication (aac). *American Journal of Speech-Language Pathology*, 2001. Citado na página 10.

FILHO, T. G.; GARCIA, J. C. Pesquisa nacional de tecnologia assistiva. *Instituto de Tecnologia Social - ITS BRASIL e Ministério da Ciência, Tecnologia e Inovação - MCTI/SECIS*, 2012. Citado na página 13.

FRANCO, N. M. et al. Modeling language and case tool for communication board customization. *2014 IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2014. Citado na página 10.

FRANCO, N. M. et al. A model-driven approach to customize the vocabulary of communication boards: Towards more humanization of health care. *MEDINFO 2015: eHealth-enabled Health*, 2015. Citado 2 vezes nas páginas 10 e 11.

GANZ, J. B.; SIMPSON, R. L. Effects on communicative requesting and speech development of the picture exchange communication system in children with characteristics of autism. *Journal of autism and developmental disorders*, Springer, v. 34, n. 4, p. 395–409, 2004. Citado na página 11.

GLENNEN, S.; DECOSTE, D. C. *The handbook of augmentative and alternative communication*. [S.l.]: Cengage Learning, 1997. Citado na página 14.

HOGETOP, L.; SANTAROSA, L. Tecnologias assistivas: Viabilizando a acessibilidade ao potencial individual. *PGIE-UFRGS*, 2002. Citado na página 13.

LOPRESTI, E. F.; MIHAILIDIS, A.; KIRSCH, N. Assistive technology for cognitive rehabilitation: State of the art. *Neuropsychological Rehabilitation: An International Journal*, 2004. Citado na página 13.

MANZINI, E. J. Tecnologia assistiva para educação: recursos pedagógicos adaptados. *Ensaios pedagógicos: construindo escolas inclusivas. Brasília: SEESP/MEC*, p. 82–86, 2005. Citado na página 13.

MIRANDA, C.; ICD, G. Contribuições da comunicação alternativa de baixa tecnologia em paralisia cerebral sem comunicação oral: relato de caso. *Rev CEFAC*, v. 6, n. 3, p. 247–52, 2004. Citado 2 vezes nas páginas 9 e 14.

MIRENDA, P. Autism, augmentative communication, and assistive technology what do we really know? *Focus on Autism and Other Developmental Disabilities*, SAGE Publications, v. 16, n. 3, p. 141–151, 2001. Citado na página 14.

SCHLOSSER, R. W.; WENDT, O. Effects of augmentative and alternative communication intervention on speech production in children with autism: A systematic review. *American Journal of Speech-Language Pathology*, ASHA, v. 17, n. 3, p. 212–230, 2008. Citado na página 14.