Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Prevenir: Aplicativo Educacional para Smartphones de Apoio à Prevenção de Acidentes Domésticos para Pessoas com Transtorno do Espectro Autista

Clara Senra Rabello

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Wilson Henrique Veneziano

Coorientadora
Prof.a M.Sc. Maraísa Helena Borges Estevão Pereira

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Prevenir: Aplicativo Educacional para Smartphones de Apoio à Prevenção de Acidentes Domésticos para Pessoas com Transtorno do Espectro Autista

Clara Senra Rabello

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Wilson Henrique Veneziano (Orientador)
CIC/UnB

Prof.a Dr.a Maristela Terto de Holanda
CIC/UnB

Prof. Dr. Jan Mendonca Correa
CIC/UnB

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 26 de maio de 2021

Dedicatória

Este trabalho é dedicado aos meus pais, que me ensinaram a buscar o conhecimento e sempre me incentivaram e aconselharam durante minha jornada.

Agradecimentos

Agradeço aos meus pais, às minhas irmãs e à Socorro, pelo constante apoio, amor, carinho e incentivo ao longo da minha trajetória.

Aos meus amigos, Eduardo, João e Matheus, pela amizade e pelos inúmeros momentos compartilhados desde o início da faculdade.

À Universidade de Brasília e aos ótimos professores que pude conhecer, que me despertaram curiosidade pela computação e tanto me ensinaram.

Ao Prof. Dr. Wilson Henrique Veneziano e à Profa. M.Sc. Maraísa Helena Borges Estevão Pereira, que me guiaram neste trabalho e tornaram possível a idealização e elaboração do projeto.

Resumo

O presente trabalho refere-se ao desenvolvimento de um *software*, o aplicativo Prevenir, que é uma ferramenta que propõe apoio para os pais, familiares e responsáveis por pessoas com Transtorno do Espectro Autista (TEA), à medida que orienta o própria pessoa com TEA a reconhecer situações perigosas em seu domicílio e a evitá-las, ajudando, assim, na conquista da autonomia. O aplicativo é gratuito e pode ser utilizado em Android e iOS. Tendo em vista seu público, pessoas com TEA, o aplicativo utiliza comandos diretos, navegação fácil e intuitiva, orientações simples, imagens e vídeos que ilustram e exemplificam situações perigosas em ambientes domésticos e atividades que podem ser personalizadas levando em conta a subjetividade e a necessidade de cada indivíduo. O aplicativo possui, ainda, a capacidade de integração com um *hardware*, O Cubinho [1], que possui diversos reforçadores customizáveis para as atividades, utilizando cores, luzes, movimento e imagens com o intuito de aumentar o interesse da pessoa com TEA no *software*.

Palavras-chave: autismo, aplicativo, prevenção de acidentes domésticos, android, ios, educação especial, transtorno do espectro autista

Abstract

This work describes the development of a free application for Android and iOS named Prevenir, a tool for the families and people responsible for individuals in the autism spectrum disorder (ASD). The app aims to help people with ASD recognize and avoid danger situations in the domestic environment. It uses simple and direct commands, intuitive navigation, objective and easy to follow instructions, images and videos illustrating dangerous situations and how to deal with them and activities that can be customized considering each individual's needs. The app also has the capability to connect to a piece of hardware named Cubinho, which uses sounds, light, images and movement to motivate and enhance the interest of the person with ASD in the app's activities.

Keywords: autism, application, prevention of domestic accidents, android, ios, special education, autism spectrum disorder

Sumário

Lista de Figuras

Capítulo 1

Introdução

A utilização da tecnologia como um importante recurso voltado para a prática educacional já está há muito tempo consolidada. Contudo, o uso da informática na educação informal e na construção do desenvolvimento de habilidades voltadas para a autonomia de pessoas com Transtorno do Espectro Autista (TEA) ainda é um desafio, uma vez que não se encontra com facilidade *softwares* destinados a esse fim.

Caminha et. al [4] ressaltam que a tecnologia pode ser bastante atrativa para pessoas com autismo, despertando o interesse e a atenção. Britto [5], em sua dissertação para pós-graduação, "GAIA: uma proposta de guia de recomendações de acessibilidade Web com foco em aspectos do autismo", afirma que as tecnologias podem ajudar as pessoas com TEA e também a todos aqueles que lidam com elas:

> "[...] [As tecnologias de apoio] têm sido úteis não somente para a pessoa com TEA, mas também para os pais, terapeutas e educadores, provendo suporte para:
>
> 1. Desenvolver as habilidades da pessoa com TEA;
> 2. Ajudar a organizar a rotina;
> 3. Auxiliar na alfabetização e desenvolvimento da linguagem;
> 4. Auxiliar a pessoa com TEA a se comunicar com família e amigos."

Segundo Fonseca e Shirmer [6], os aplicativos móveis possuem a capacidade de amparar o aluno durante o aprendizado, proporcionando experiências sensoriais importantes para pessoas com TEA.

Assim, com a intenção de contribuir com o processo de aprendizagem das pessoas com TEA, auxiliando os pais, familiares e profissionais quanto à identificação de objetos que podem oferecer perigo no ambiente doméstico; criou-se o aplicativo Prevenir, que tem como foco a prevenção de acidentes domésticos.

O aplicativo Prevenir é ainda mais relevante neste ano, com a pandemia do novo coronavírus e o isolamento social: com mais tempo em casa, é necessário redobrar a atenção

com acidentes domésticos. Por isso, considera-se que a iniciativa de desenvolver um aplicativo gratuito para dispositivos móveis, que ofereça orientação para que pessoas com TEA possam reconhecer possíveis situações de risco em seu lar, pode ajudar a reduzir os acidentes domésticos e, ao mesmo tempo, contribuir para a autonomia desses indivíduos.

Vale ressaltar que o aplicativo é customizável às necessidades individuais dos usuários, sendo possível inserir imagens familiares dos objetos de casa e imagens de objetos que forneçam algum estímulo positivo à pessoa com TEA. Isso pode ser feito utilizando a galeria ou câmera fotográfica do aparelho.

1.1 Problema

Enquanto existe uma oferta grande e variada de aplicativos destinados aos mais diversos públicos, constatou-se uma carência de aplicativos em língua portuguesa voltados ao desenvolvimento de habilidades requeridas na vida diária de pessoas com TEA, em especial *softwares* que trabalhem a prevenção de acidentes domésticos utilizando situações cotidianas e sociais facilmente identificáveis pelas pessoas com TEA.

1.2 Justificativa

A cartilha Prevenção para Acidentes Domésticos e Guia Rápido de Primeiros Socorros, do Ministério da Mulher, da Família e dos Direitos Humanos [7], informa que, segundo dados do Sistema de Informações sobre Mortalidade (SIM) do Ministério da Saúde, em 2015 foram registradas 2.441 mortes de crianças de 0 a 14 anos, no Brasil, devido a acidentes domésticos. Perceber e evitar situações perigosas no ambiente doméstico já é desafiador para pessoas neurotípicas, e pode ser ainda mais complexo para pessoas com TEA.

Assim, o aplicativo Prevenir surgiu para contribuir com a formação e a conquista de habilidades e comportamentos capazes de auxiliar a pessoa com TEA a reconhecer e a afastar situações de risco, ajudando, portanto, na prevenção de acidentes domésticos. Tendo em vista que existem, no mercado, poucos aplicativos voltados para o desenvolvimento da autonomia de pessoas com TEA, o *software* se torna ainda mais relevante.

1.3 Objetivo Geral

Desenvolver um *software* gratuito para Android e iOS voltado para a prevenção de acidentes domésticos de pessoas do espectro autista e destinado a contribuir com o processo de aprendizagem destas pessoas e a auxiliar pais, familiares e responsáveis por elas a

reconhecer situações perigosas e a evitá-las, contribuindo, assim, para a autonomia das pessoas com TEA.

1.4 Objetivos Específicos

Com o intuito de atingir o objetivo geral estabelecido, foram definidos os seguintes objetivos específicos:

- Analisar as possíveis causas de acidentes domésticos para pessoas com TEA;
- Estudar e analisar os métodos de aprendizado para pessoas com TEA;
- Desenvolver atividades instrucionais levando em conta as particularidades de aprendizado de pessoas com TEA;
- Desenvolver, testar e avaliar o aplicativo.

1.5 Procedimento Adotado

O presente trabalho foi desenvolvido a partir de vasta pesquisa bibliográfica que incluiu a consulta a livros, artigos relacionados aos pertinentes temas e o banco de dados da Scientific Eletronic Library Online – SciELO, visando aprofundar os conhecimentos relativos ao desenvolvimento de aplicativos, ao autismo e à prevenção de acidentes.

É importante ressaltar que todo o direcionamento acerca do Transtorno do Espectro Autista foi dado pelos professores especialistas em Educação Especial, que especificaram as funcionalidades que deveriam existir no aplicativo para contemplar esse público e se adequar às suas reais necessidades. Neste contexto, foram realizadas as seguintes etapas:

- Estudo sobre o Transtorno do Espectro Autista visando entender suas peculiaridades e os desafios enfrentados por pessoas com TEA na prática das atividades diárias;
- Estudo sobre o uso de tecnologia como ferramenta de apoio a pessoas com TEA;
- Estudo sobre acidentes domésticos e sobre medidas de prevenção para minimizar os riscos de acidentes mediante a adoção de comportamentos mais seguros;
- Levantamento das funcionalidades e características do aplicativo com professores especialistas na área educacional;
- Estudo sobre o desenvolvimento de aplicações para *smartphone*;
- Desenvolvimento do *software*;

- Criação de uma base de dados de objetos domésticos separando-os entre objetos seguros e não-seguros;
- Testes das funcionalidades do *software*.

1.6 Organização do Trabalho

A presente monografia está estruturada da seguinte maneira:

- Capítulo 2 - aborda informações relevantes a este trabalho no que diz respeito ao público-alvo do aplicativo: pessoas com Transtorno do Espectro Autista, com intuito de ambientar acerca do transtorno e seu contexto histórico, suas características principais, formas de abordagem do ensino, métodos de aprendizado, entre outros;
- Capítulo 3 - apresenta conceitos teóricos e características das tecnologias utilizadas no desenvolvimento do *software*;
- Capítulo 4 - expõe a visão geral do *software* desenvolvido, com todas as telas do aplicativo e explicações acerca de cada atividade;
- Capítulo 5 - aborda a conclusão do desenvolvimento do projeto, com sugestões de trabalhos futuros para o aperfeiçoamento do aplicativo.

Capítulo 2

O Espectro Autista

Este capítulo tratará sinteticamente do contexto histórico do Transtorno do Espectro Autista, bem como abordará sua definição, diagnóstico, características e o processo de aprendizagem de pessoas com TEA. Também será relatada a legislação brasileira direcionada a proteção dessas pessoas.

2.1 Contexto Histórico

O termo autismo foi utilizado pela primeira vez em 1911 pelo psiquiatra suíço Eugen Bleuler, para se referir à fuga da realidade para um mundo interior observada em seus pacientes com esquizofrenia [8]. A palavra surgiu do grego autós+ismo, sendo que "autos" se refere à voltar-se para si mesmo.

Em 1943, o psiquiatra Leo Kanner publicou um artigo descrevendo o estudo de 11 crianças com um quadro caracterizado pelo desejo de rotina, dificuldade na interação social, estereotipias e ecolalia [9]. Em sua visão, a princípio, este quadro estava associado à esquizofrenia infantil. Posteriormente, ele atualizou o nome para "autismo infantil precoce", uma vez que os sinais eram visíveis já nos primeiros dois anos de vida.

Na mesma época, outro médico, Hans Asperger, trabalhava também com crianças com autismo e foi o primeiro a relatar a maior incidência do autismo em meninos [10]. Apesar do papel pioneiro, o artigo "A psicopatia autista na infância de Asperger" não recebeu a devida importância quando foi publicado durante a Segunda Guerra Mundial, e apenas na década de 1980 seu trabalho foi reconhecido.

Em 1978, o psiquiatra inglês Michael Rutter propôs uma nova definição do distúrbio, como um transtorno mental único, independente da esquizofrenia, com base em quatro critérios: atraso e desvio sociais; problemas de comunicação; comportamentos incomuns, tais como movimentos estereotipados e maneirismos; e início antes dos 30 meses de idade [11].

As inovações trazidas por Michael Rutter, em conjunto com a crescente produção de pesquisas científicas sobre o autismo, influenciaram na elaboração do DSM-3 (Manual Diagnóstico e Estatístico de Transtornos Mentais), em 1980. Pela primeira vez, o autismo foi reconhecido como uma condição única, fora do conceito de esquizofrenia, e colocado em uma nova classe denominada Transtornos Invasivos do Desenvolvimento (TID), em razão de várias áreas de funcionamento do cérebro serem afetadas pelo autismo e pelas condições a ele relacionadas [12].

Paralelamente, a psiquiatra Lorna Wing desenvolveu o conceito de autismo como um espectro e em 1981 cunhou o termo Síndrome de Asperger em referência a Hans Asperger. Wing reconheceu como características comuns ligadas ao autismo uma tríade de prejuízos - Tríade de Wing [13], que define:

- prejuízo na interação social recíproca;
- prejuízo na comunicação verbal e não-verbal;
- padrões estereotipados repetitivos e restritos de comportamento, interesses e atividades.

Em 2013, o Manual Diagnóstico e Estatístico de Transtornos Mentais - referência mundial para médicos, psicólogos e pesquisadores -, passou a abrigar todas as subcategorias do autismo em um único diagnóstico: Transtorno do Espectro Autista (TEA) [12].

2.2 Principais características e diagnósticos

A definição de autismo mais aceita atualmente é dada pela Academia Americana de Psiquiatria em seu Manual de Diagnóstico e Estatística de Doenças Mentais (DSM), atualmente em sua quinta versão. Segundo o manual, o autismo é definido como uma síndrome que afeta o neurodesenvolvimento e tem como características essenciais o "prejuízo persistente na comunicação social recíproca e na interação social (Critério A) e padrões restritos e repetitivos de comportamento, interesses ou atividades (Critério B)" [14].

Essas características estão presentes desde cedo e podem trazer prejuízo significativo no funcionamento social, profissional ou em outras áreas importantes da vida do indivíduo. Elas configuram o núcleo do transtorno, mas a gravidade de sua apresentação pode variar [15]. Assim, considerando que existe uma diversidade na intensidade das manifestações clínicas, as novas classificações começaram a preferir a expressão Transtorno do Espectro Autista.

Pessoas diagnosticadas com TEA podem ter seu transtorno classificado entre 3 categorias: alta funcionalidade, média funcionalidade e baixa funcionalidade [16]. A primeira

categoria apresenta prejuízos leves à pessoa, que podem não impedi-la de estudar, trabalhar e se relacionar. A segunda está associada a um menor grau de independência para o portador, que possivelmente precisará de algum tipo de auxílio para desempenhar funções cotidianas. A terceira manifesta dificuldades graves na vida da pessoa com TEA, resultando, geralmente, na necessidade de um apoio especializado conforme a necessidade.

Por outro lado, o diagnóstico de TEA pode ser, também, acompanhado de habilidades socialmente tidas como "positivas", como grande capacidade de memorização, facilidade para aprender visualmente, foco a detalhes e concentração em áreas de interesse específicas do portador de TEA [16]. Todas estas características influenciam em como cada pessoa se relaciona, se expressa e se comporta.

O diagnóstico de autismo é clínico, com entrevistas e informações dadas pelos pais, aplicação de escalas, questionários e histórico do paciente [15]. Ainda não há exames laboratoriais específicos ou marcadores biológicos para constatar o autismo, tornando o diagnóstico mais complicado. Diagnosticar precocemente é fundamental: segundo a Sociedade Brasileira de Pediatria, "[...] a intervenção precoce está associada a ganhos significativos no funcionamento cognitivo e adaptativo da criança". Entretanto, apesar de as características do TEA estarem presentes precocemente, muitas crianças são diagnosticadas tardiamente.

As causas do autismo ainda são desconhecidas. Segundo Mello [17], atualmente acredita-se que o TEA esteja relacionado a anormalidades em alguma parte do cérebro ainda não definida, tendo, portanto, origens genéticas.

Esse posicionamento é reforçado pela Sociedade Brasileira de Pediatria, que traz, no Manual sobre Transtorno do Espectro do Autismo [15], que o "TEA é causado por uma combinação de fatores genéticos e fatores ambientais". Complementa dizendo que:

> "Apesar de claramente importantes, os fatores genéticos não atuam sozinhos, sendo sua ação influenciada ou catalisada por fatores de risco ambiental, incluindo, entre outros, a idade avançada dos pais no momento da concepção, a negligência extrema dos cuidados da criança, a exposição a certas medicações durante o período pré-natal, o nascimento prematuro e baixo peso ao nascer."

2.3 Modalidades de Intervenção

Nem todas as pessoas com TEA conseguem viver de forma independente. Aquelas que possuem deficiências mais severas irão precisar de constante apoio e cuidados ao longo da vida [18]. Por essa razão, a Sociedade Brasileira de Pediatria reforça a importância da intervenção precoce com a adoção de uma gama de terapias para a pessoa com TEA, destinadas a incrementar a socialização e a comunicação, melhorar a qualidade de vida

com foco na autonomia, resguardar o funcionamento cognitivo, bem como dar suporte a família.

Conforme Carothers e Taylor [19], "quando se pensa em educar uma criança com autismo, o objetivo dessa educação, considerando-se as especificidades da mesma, é o de aumentar sua independência, de forma a proporcionar dignidade e qualidade de vida para a criança e seus familiares.".

Assim sendo, as modalidades de intervenção têm como escopo propiciar o desenvolvimento da pessoa com autismo com o máximo de autonomia possível, melhorando a sua qualidade de vida e a da sua família, uma vez que uma boa intervenção consegue reduzir comportamentos indesejados e minimizar os prejuízos nas áreas do desenvolvimento. Os três tipos mais usuais de intervenção [17] são o Tratamento e Educação para Crianças Autistas e com Distúrbios Correlatos da Comunicação (*Treatment and Education of Autistic and related Communication-handicapped Children* - TEACCH), Análise Aplicada ao Comportamento (*Applied Behavior Analysis* - ABA) e Sistema de Comunicação Através da Troca de Figuras (*Picture Exchange Communication System* - PECS), que serão brevemente explicados abaixo:

TEACCH: Usa como recurso um teste denominado PEP-R (Perfil Psicoeducacional Revisado). Essa avaliação considera, para montar um programa individualizado, os pontos fortes e as dificuldades das pessoas com TEA. O TEACCH tem como fundamento a organização do ambiente físico através de rotinas buscando desenvolver a independência da pessoa com TEA [17];

ABA: É uma intervenção que contribui para a ampliação da capacidade de comunicação, ajudando as pessoas com TEA a se relacionarem melhor com a sociedade e com o ambiente. Consiste em ensinar habilidades à criança através de etapas, fragmentando as tarefas e associando-as a instruções. Quando o indivíduo com TEA responde de forma adequada, obtém um reforçador positivo, um *feedback* imediato a fim de manter-se motivado e engajado para voltar a repetir a resposta. Na hipótese de resposta negativa não há o reforçador. O recurso utilizado como reforçador positivo deve ser escolhido tendo em vista o indivíduo – aquilo que ele gosta e que faz sentido para ele – e deve ser retirado assim que possível para não criar dependência, uma vez que o objetivo é a conquista da autonomia [20];

PECS: É um sistema de comunicação alternativa/aumentativa desenvolvido nos Estados Unidos em 1985 por Andy Bondy e Lori Frost. O PECS objetiva ajudar a criança a perceber que, através da comunicação, ela pode conseguir mais rapidamente aquilo que quer, acarretando na diminuição de problemas de conduta. É um sistema para

ajudar pessoas que não conseguem se fazer entender através da fala, ou que têm uma fala muito limitada [21].

2.4 Legislação Brasileira relacionada ao Autismo

As pessoas com TEA têm os mesmos direitos que todos os outros cidadãos do Brasil. Esses direitos são garantidos pela Constituição Federal de 1988 [22] e pelas demais leis nacionais. Assim, as crianças e adolescentes com autismo possuem seus direitos previstos no Estatuto da Criança e Adolescente (Lei 8.069/90) [23], e os maiores de 60 anos no Estatuto do Idoso (Lei 10.741/2003) [24].

A Lei Berenice Piana (Lei 12.764/12) [25] criou a Política Nacional de Proteção dos Direitos da Pessoa com Transtorno do Espectro Autista. Essa lei prevê o direito de pessoas com autismo a um diagnóstico precoce, tratamento, terapias e medicamento pelo Sistema Único de Saúde; o acesso à educação e à proteção social; ao trabalho e a serviços que propiciem a igualdade de oportunidades. Estipula, ainda, que a pessoa com TEA é considerada pessoa com deficiência, para todos os efeitos legais.

Isto permitiu que as pessoas com TEA fossem abrangidas nas leis específicas de pessoas com deficiência, como o Estatuto da Pessoa com Deficiência (13.146/15) [26], que define a pessoa com deficiência como "aquela que tem impedimento de longo prazo de natureza física, mental, intelectual ou sensorial", bem como nas normas internacionais assinadas pelo Brasil, como a Convenção das Nações Unidas sobre os Direitos das Pessoas com Deficiência (Decreto Federal 6.949/2000) [27].

Além disso, é interessante citar a alteração ocorrida na Lei de Diretrizes e Bases da Educação Nacional - LDB (Lei 9.394/1996) em 2018, que determinou como dever do Estado garantir que a educação especial na primeira infância (zero a seis anos) se estenda ao longo da vida para as pessoas com deficiência [28].

Vale trazer, ainda, algumas leis que tratam de outras questões igualmente importantes:

- **Lei 8.899/1994:** concede passe livre à portadora de deficiência que comprove ser carente [29].
- **Lei 8.742/1993:** A Lei Orgânica da Assistência Social (LOAS) oferece o Benefício da Prestação Continuada [30].
- **Decreto 7.611/2011:** Dispõe sobre a educação especial e o atendimento educacional especializado [31].
- **Lei 7.853/1989:** Estipula o apoio às pessoas portadoras de deficiência, sua integração social, institui a tutela jurisdicional de interesses coletivos ou difusos dessas pessoas, disciplina a atuação do Ministério Público e define crimes [32].

- **Lei 10.048/2000:** Dá prioridade de atendimento às pessoas com deficiência e outras providências [33].

2.5 Prevenção de acidentes

A Portaria nº 737/GM, de 16 de maio de 2001, da Política Nacional de Redução da Morbimortalidade por Acidentes e Violências, do Ministério da Saúde, define acidente como "o evento não intencional e evitável, causador de lesões físicas e/ou emocionais no âmbito doméstico ou nos outros ambientes sociais, como o do trabalho, do trânsito, da escola, de esportes e o de lazer" [34].

Os acidentes domésticos são relevantes causas de internação hospitalar e de mortalidade entre crianças, adultos e pessoas idosas [35]. Esses acidentes constituem um desafio, uma vez que representam um problema multifatorial, com diferentes implicações e abordagens conforme a faixa etária da vítima.

O Plano da Rede Nacional Primeira Infância [36] traz as seguintes informações:

> "As lesões não intencionais (LNI), popularmente conhecidas como acidentes e que na área médica são chamadas de traumas, são uma das maiores vilãs na primeira infância no Brasil. Os acidentes são a primeira causa de morte na faixa etária de 0 a 14 anos no Brasil. [...] A violência doméstica é uma das mais frequentes e graves. Em 2017, segundo dados do Datasus, 6.143 crianças menores de 1 ano foram vítimas dessa violência, enquanto 4.092 foram por negligência e abandono; e 1.758, violência física. Entre as crianças de 1 a 4 anos, o número de casos relatados alcançou 12.728, sendo que 7.581 eram por negligência e abandono; 2.950, de violência física; 3.042, de violência sexual; e 1.784, psicológica/moral."

Os locais onde se encontram os maiores riscos são a casa e a escola, justamente por serem os mais frequentados por crianças e adolescentes. Segundo a Sociedade Brasileira de Pediatria [37], a maioria dos acidentes domésticos poderia ser evitada com medidas simples.

Com a pandemia do novo Coronavírus e o consequente isolamento social, houve necessidade de uma série de mudanças na rotina de todos, criando um desafio ainda maior para os responsáveis pelas pessoas com TEA. Apesar de ser comum assumir que dentro de casa fica-se em total segurança, inúmeras situações e objetos comuns do dia a dia podem apresentar riscos, como pontuado pelo Guia de Prevenção aos Acidentes Domésticos, do Ministério da Mulher, da Família e dos Direitos Humanos [7]. Estes casos são ainda mais recorrentes para indivíduos com autismo, que podem possuir déficits na capacidade de identificação de uma situação perigosa e precisar de instruções extensivas para dominar habilidades da vida diária.

2.6 Tecnologia Assistiva

O Comitê de Ajudas Técnicas - CAT, instituído pela Portaria 142, de 16 de novembro de 2006, conceitua Tecnologia Assistiva como uma área do conhecimento que engloba diversos recursos e metodologias para promover a funcionalidade, atividade e participação de pessoas com deficiência, visando sua autonomia, independência, qualidade de vida e inclusão social [38].

De acordo com a Subsecretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência, Tecnologia Assistiva diz respeito à pesquisa, à fabricação, ao uso de equipamentos, recursos ou estratégias para potencializar as habilidades funcionais das pessoas com deficiência; englobando profissionais de várias áreas do conhecimento [39].

Existem vários tipos de tecnologias assistivas classificadas de acordo com a função: tecnologias para auxiliar a locomoção, para facilitar o acesso à informação e à comunicação, para auxiliar o desempenho das atividades diárias, para permitir o esporte e o lazer, entre outras. Pode ser desde uma muleta, bengala ou lupa, até complexos sistemas de computadores.

A tecnologia móvel dos *smartphones* e *tablets*, por serem estes dispositivos portáteis, versáteis, acessíveis em tempo real e em qualquer lugar, está cada vez mais presente no dia a dia das pessoas. É capaz de exercer, sobre indivíduos com TEA, grande atratividade e despertar o interesse e atenção destes, conforma destacam Caminha et. al [4]. Isso torna a tecnologia móvel um recurso eficaz no atendimento das necessidades específicas das pessoas com transtornos, ou déficits, como no caso do Transtorno do Espectro Autista [40].

É nessa concepção de tecnologia assistiva que se encontram os fundamentos para a criação e o desenvolvimento do presente software de prevenção de acidentes domésticos destinados a pessoas com TEA.

2.7 Acessibilidade na *Web*

Acessibilidade na *web* diz respeito à prática que permite que todos os indivíduos, independente de suas capacidades motoras, visuais, auditivas, intelectuais, culturais ou sociais, sejam capazes de perceber, entender, navegar, interagir e contribuir para a *web* [41]. Esta definição é legitimada pela W3C Brasil (*World Wide Web Consortium* - Brasil), uma organização que cria e mantém os padrões para os sites na internet, em sua Cartilha de Acessibilidade na *Web*. A cartilha define, ainda, princípios que um site deve seguir para se mostrar acessível, dentre os quais podem ser citados:

- O conteúdo deve estar disponível de diferentes formas, como por exemplo no uso de textos alternativos como descrição de imagens;
- O site deve permitir que o usuário controle o tempo de consumo das informações;
- O conteúdo deve ser compreensível e possuir níveis de linguagens adequados para cada público;
- O site deve ser bem estruturado no âmbito técnico para que possa ser corretamente interpretado por tecnologias assistivas.

Entretanto, conforme Britto [5], diretrizes genéricas de acessibilidade podem não atender pessoas com TEA em razão das especificidades próprias desses indivíduos, gerando ansiedade e estresse, além de impossibilitar o alcance de objetivos pedagógicos e terapêuticos por falta de uma interação adequada. Britto traz recomendações de acessibilidade para pessoas com TEA, que, em síntese, são as seguintes:

1. **Vocabulário Visual e Textual**
   (a) Cores – o contraste entre as cores de fundo e objetos de primeiro plano deve ser adequado para distinguir os itens e diferenciar conteúdos ou relacionar informações similares.
   (b) Textos - linguagem visual e textual simples, sem jargões, erros ortográficos, metáforas, abreviações e acrônimos; termos, expressões, nomes e símbolos familiares ao contexto dos usuários; textos curtos, sem complexidade e sem linguagem conotativa.
   (c) Legibilidade - estrutura que permita a legibilidade, utilizando elementos como subtítulos e listas e percebendo a quantidade de caracteres por linha e espaçamento entre linhas.
   (d) Compatibilidade com mundo real - ícones, imagens e nomenclatura de ações e menus devem manter coerência com o mundo real, com ações concretas e com atividades de vida cotidianas a fim de serem mais facilmente identificadas.
2. **Customização**
   (a) Customização visual - cores, tamanho de texto e fontes utilizadas em elementos da página adaptados de acordo com as necessidades.
   (b) Customização informacional - visualização de informações com imagens, som e texto de acordo com o gosto individual.
   (c) Interfaces flexíveis - opção de personalizar a quantidade e a disposição de elementos na tela e as funcionalidades.

(d) Modo de leitura – possibilidade de modo leitura ou impressão.

3. **Engajamento**

   (a) Elimine distrações – elementos que distraiam a atenção não devem ser usados e se forem deve haver opções para retirá-los.

   (b) Interface minimalista – simples e com poucos elementos, contendo apenas as funcionalidades e conteúdos para cada tarefa.

   (c) Organização visual – espaços em branco entre os elementos da página para separar conteúdos diferentes ou para chamar a atenção em determinado conteúdo.

   (d) Forneça instruções – clareza nas orientações sobre as tarefas.

4. **Representações redundantes**

   (a) Múltiplos formatos – além de conteúdo escrito, deve haver representações em imagem, áudio ou vídeo próximas do texto correspondente.

   (b) Equivalentes textuais - símbolos, pictogramas e ícones devem estar perto do equivalente textual para ajudar na compreensão do símbolo e no aumento do vocabulário.

   (c) Legendas - instruções e legendas em áudio devem ser colocadas em textos, mas esta não deve ser a única alternativa de representação do conteúdo.

5. **Multimídia**

   (a) Múltiplas mídias – devem ser colocadas informações em texto, vídeo, áudio e imagens.

   (b) Ampliação de imagens – imagens ampliadas permitem uma melhor visualização.

   (c) Evite sons perturbadores – sons tais como de explosivos, sirenes ou fogos de artifício podem ser incômodos.

6. **Visibilidade do estado do sistema**

   (a) Instruções de interação - as instruções devem ser adequadas para interação com os elementos da página; as mensagens devem ser claras sobre os erros e devem existir mecanismos para solucionar os erros.

   (b) Reverter ações - ações críticas devem poder ser revertidas, canceladas, desfeitas ou confirmadas.

(c) Número de tentativas – deve ser permitido até cinco tentativas em uma atividade antes de mostrar a resposta correta.

7. **Reconhecimento e Previsibilidade**

   (a) Consistência – colocar resultados similares e previsíveis para elementos e interações similares.

   (b) Local para clicar - ícones, botões e controles de formulário devem ser maiores com área para tocar e com aparência de clicáveis.

   (c) *Feedback* de interação - devem existir instruções e *feedback* imediato sobre uma restrição de interação com o sistema ou com algum elemento.

8. **Navegabilidade**

   (a) Navegação simples - todas as páginas devem possuir indicadores de localização, progresso e o uso de botões de navegação global (Sair, Voltar para página inicial, ajuda).

   (b) Evite redirecionamentos – o redirecionamento de páginas automático deve ser evitado, assim como o tempo de expiração para tarefas.

9. **Resposta às ações**

   (a) Confirmação de ações - deve haver *feedback* para confirmação de ações corretas ou com alerta sobre possíveis erros mediante o uso de áudio, texto e imagens, evitando ícones com emoções ou expressões faciais.

10. **Interação com tela sensível ao toque**

    (a) Sensibilidade adequada – como os *websites* e aplicações *web* estão cada vez mais sendo acessados através de dispositivos móveis com telas sensíveis ao toque, estes devem possuir a sensibilidade adequada para a navegação.

É possível observar, ainda, outras recomendações de acessibilidade para pessoas com TEA, mais especificamente crianças, obtidas no artigo "Recomendações para o Desenvolvimento de Softwares Voltados para Crianças com Transtorno do Espectro Autista" [42]. Dentre elas, é importante citar a apresentação de níveis de dificuldade passíveis de serem configuráveis pelos responsáveis da pessoa com TEA, a fim de respeitar o tempo de aprendizagem de cada um; e a simplicidade na sequências de passos durante a navegação dentro do software, com o intuito de não desconcentrar a pessoa com TEA e gerar desistência.

Capítulo 3

Desenvolvimento do Software

Neste capítulo, serão desenvolvidos alguns conceitos teóricos importantes acerca das tecnologias utilizadas para a elaboração desse trabalho, incluindo as plataformas de desenvolvimento, linguagens de programação, interface de desenvolvimento e demais elementos de software relevantes.

3.1 Contexto Histórico

O avanço tecnológico, de maneira geral, interfere positivamente na inclusão digital, facilitando o acesso a aparelhos eletrônicos e à internet. Este acesso vem se tornando praticamente imprescindível nos dias de hoje, sendo considerado um direito humano pela Organização das Nações Unidas [43].

O aparelho responsável pelo início da popularização da internet foi o *desktop*, ou computador de mesa, que por um bom tempo foi a forma mais fácil de se conectar à rede. Com a introdução do *smartphone*, há pouco mais de dez anos, e com o enorme sucesso desse aparelho, o computador de mesa foi, aos poucos, perdendo lugar para o aparelho portátil, que se tornou uma tecnologia mais barata e de mais fácil acesso. Já faz alguns anos que o *smartphone* é o aparelho mais utilizado para acessar a internet no Brasil: segundo pesquisa do IBGE de 2019 [44], 98,6% das pessoas que acessam a internet o fazem por telefones móveis celulares, enquanto apenas 46,2% o fazem a partir de microcomputadores.

Este dado não é estranho, considerando que, ainda nesta pesquisa, foi constatado que 81% dos brasileiros possuíam um aparelho celular em 2019, e apenas 40,6% dos domicílios possuíam um *desktop*. Os brasileiros passam em média 3 horas e 45 minutos por dia utilizando o celular [45], e esse número só tende a aumentar nos próximos anos, com o barateamento das tecnologias.

Esse crescente uso de *smartphones* gera uma demanda por produtos e serviços acessíveis por aparelhos portáteis, que seguem padrões de *design* e usabilidade diferentes de sites e programas acessíveis no computador.

A necessidade de lançar conteúdo de fácil acesso por *smartphones* fez com que surgissem os aplicativos de celular, ou apps: programas de software desenvolvidos para *smartphones*. Seu desenvolvimento leva em conta o hardware e o software desses dispositivos, adaptando-se às necessidades de um aparelho portátil, como o tamanho reduzido da tela e a menor capacidade de processamento do que um computador de mesa. Hoje, estima-se que existam cerca de 5.2 milhões de aplicativos disponíveis para os dois principais sistemas operacionais de dispositivos móveis (Android e iOS) [46] e a tendência é que este número só aumente.

Com a alta do uso dos *smartphones* e a crescente formação de desenvolvedores, é fácil perceber que ferramentas que fazem parte ou auxiliam no desenvolvimento de aplicativos vêm ganhando cada vez mais espaço, surgindo uma grande gama de opções de linguagens de programação, interfaces de desenvolvimento (IDEs - Integrated Development Environment) e *frameworks* pensados primordialmente para o desenvolvimento móvel.

3.2 Sistemas Operacionais

O crescimento do mercado de *smartphones* fez com que surgissem diversos sistemas operacionais para estes dispositivos, como Android, iOS, KaiOS, Windows Phone, SymbianOS, BlackBerryOS, entre outros [47].

Inicialmente, quando os *smartphones* ainda eram novidade, existia uma grande variedade de sistemas operacionais dividindo o setor de dispositivos móveis. Em pesquisa de janeiro de 2012 [48], 23.21% dos celulares utilizavam Android, 24.04% utilizavam iOS, 31.89% utilizavam Symbian OS e 6.94% utilizavam BlackBerryOS.

Com o passar do tempo, a maioria desses sistemas foi perdendo força, e hoje em dia, apenas 2 deles detêm quase 100% do setor: uma pesquisa de janeiro de 2021 [48] mostrou que 71.93% dos *smartphones* rodam em Android, 27.47% rodam em iOS, 0.1% rodam em KaiOS e 0.5% rodam em outros sistemas.

Nesta seção, serão apresentados os sistemas operacionais mais utilizados atualmente em dispositivos móveis, Android e iOS, que foram analisados para o desenvolvimento do presente aplicativo.

3.2.1 Android

O Android é um sistema operacional de código aberto para dispositivos móveis, como celulares e tablets, lançado comercialmente pelo Google em setembro de 2008.

Arquitetura

O sistema operacional Android tem sua arquitetura construída sobre o núcleo do Linux padrão e pode ser separada entre as seguintes camadas [2] (Figura 3.1):

- **Aplicativos do sistema**: é a camada onde se encontra o grupo de aplicativos que vem incluído no sistema, como aplicativos de e-mail, SMS, calendários, navegadores de internet, entre outros.
- **Java API Framework**: esta camada contém os componentes necessários para criar aplicativos que utilizam recursos do sistema operacional, incluindo funcionalidades de gerência de notificações e compartilhamento de dados com a agenda e a lista de contatos do sistema. No Android, é interessante notar que aplicativos desenvolvidos por terceiros têm o mesmo acesso a esta camada que aplicativos do sistema.
- **Bibliotecas C/C++ nativas**: é a camada que contém bibliotecas nativas escritas em C e C++ que são necessárias para o funcionamento de alguns componentes e serviços do sistema.
- **Android Runtime (ART)**: é um ambiente de tempo de execução onde são executados os aplicativos e alguns serviços do sistema. Foi escrito para trabalhar com o formato DEX, um formato bytecode especial para Android que otimiza o uso da memória; e oferece ao desenvolvedor um ótimo suporte à ferramenta debug, com diagnósticos para exceções e relatórios de falha detalhados.
- **Camada de abstração de hardware (HAL)**: esta camada é responsável por expor as capacidades do hardware (como câmera, bluetooth, entre outros) para a API de alto nível, onde os aplicativos poderão acessá-las.
- **Kernel do Linux**: é a camada mais baixa na arquitetura do Android, onde se encontra uma versão do kernel do Linux padrão, aproveitando recursos de segurança deste, mas com algumas adições relacionadas a gerenciamento de memória, energia, entre outros.

Segundo Tanenbaum [49], desde seu lançamento, o Android cresceu para ser um dos sistemas operacionais de *smartphones* mais amplamente usados. Sua popularidade pode ser explicada, principalmente, por ser um sistema de código aberto, que permite uma melhor customização para cada dispositivo e um custo mais baixo para o produto final do *smartphone*. Atualmente, é utilizado não somente em dispositivos móveis, mas também em diversos dispositivos dedicados que precisam de uma interface gráfica de usuário, como smartwatches e painéis de automóveis.

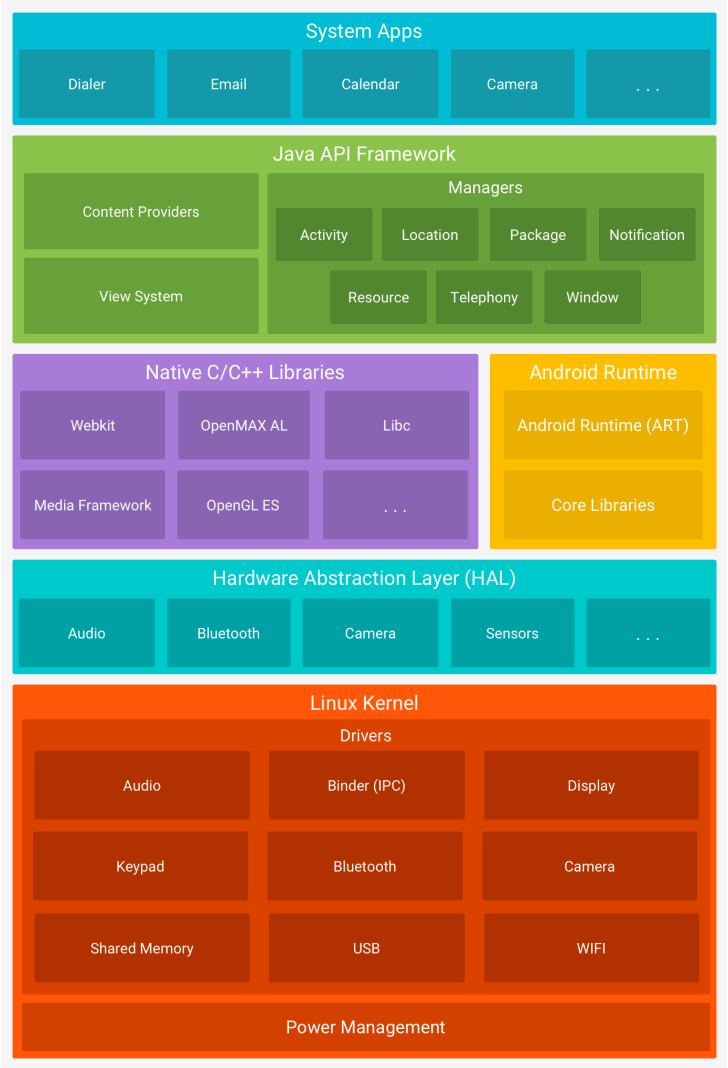


Figura 3.1: Camadas da Arquitetura Android [2].

Aplicativos

Aplicativos para Android são arquivos compactados numa extensão .apk (Android Package) e definem um contêiner com tudo que é necessário para formar aquela aplicação. São constituídos principalmente por [49]:

- um manifesto descrevendo o que é, o que faz e como executar a aplicação
- os recursos necessários para a execução da aplicação
- o código da aplicação
- informações de assinatura para identificar o autor da aplicação

A plataforma oficial de aplicativos Android é a Google Play Store, lançada em outubro de 2008 pelo Google (inicialmente, com o nome de Android Market). De acordo com pesquisa realizada pela AppBrain em maio de 2021, a plataforma conta atualmente com cerca de 2.9 milhões de aplicativos disponíveis [50].

Desde o lançamento da plataforma Android, Java predominou como linguagem de programação para o desenvolvimento de aplicativos Android [51]. Em 2017, entretanto, foi anunciado na conferência Google I/O daquele ano que o Google passaria a suportar oficialmente a linguagem Kotlin, lançada em 2011 pela JetBrains, para o desenvolvimento de seus aplicativos [52]. Já na conferência de 2019, Kotlin foi definida pelo Google como a melhor linguagem para se desenvolver um aplicativo Android. Depois disso, a nova linguagem se popularizou bastante, e atualmente mais de 50% de desenvolvedores Android a utilizam para desenvolver aplicativos [53].

3.2.2 iOS

O iOS é um sistema operacional desenvolvido pela Apple, lançado em junho de 2007, de código fechado – seu uso é exclusivo para os hardwares da própria Apple (iPhones, iPads e iPods). É um sistema baseado em UNIX e escrito em C, C++, Objective-C e Swift [54].

Arquitetura

A arquitetura do iOS pode ser dividida entre as seguintes camadas [3] (Figura 3.2):

- **Cocoa Touch**: é a camada que contém *frameworks* chave que definem a aparência de um aplicativo e oferecem infraestrutura básica e suporte para tecnologias como multitarefa, entrada de dados por touch, notificações push, entre outros.
- **Media**: contém as tecnologias necessárias para implementar experiências multimídia (gráfica, aúdio e vídeo) em um aplicativo.
- **Core Services**: é a camada que contém serviços do sistema fundamentais para aplicativos, assim como tecnologias necessárias para a implementação de recursos de localização, nuvem, mídias sociais e conexão à internet.
- **Core OS**: contém recursos de baixo nível que fundamentam a base para outras tecnologias. Mesmo que um aplicativo não utilize diretamente recursos desta camada, provavelmente grande parte dos *frameworks* utilizados no desenvolvimento dele o faz.

Pela limitação de uso do iOS a hardwares da Apple, o número de usuários deste sistema está naturalmente ligado ao número de usuários de iPhone. Com a baixa competitividade

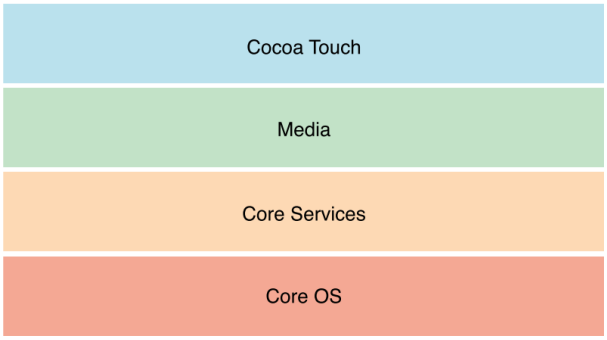


Figura 3.2: Camadas da Arquitetura iOS [3].

de preços de tais *smartphones* comparados a dispositivos de diversas outras fabricantes (Samsung, Motorola, Xiaomi, LG, entre outras), é natural que o número de usuários de Android seja maioria. Segundo pesquisa realizada pela Gartner, no primeiro trimestre de 2018 apenas 14.1% dos *smartphones* vendidos operavam em iOS, em comparação a 85.9% em Android [55].

A vantagem de tal limitação, entretanto, está no aumento da capacidade de integração entre o sistema operacional e o *smartphone* [56]. Como tanto o hardware quanto o software são completamente desenvolvidos pela Apple, a comunicação entre eles pode ser feita de forma otimizada e os recursos do dispositivo podem ser aproveitados ao máximo pelo sistema operacional e pelos aplicativos.

Aplicativos

Assim como os arquivos .apk no Android, aplicativos para iOS também são arquivos compactados que definem informações importantes para a execução de um aplicativo. A extensão de tais arquivos é a .ipa (iOS App Store Package). Estão contidos no pacote [57]:

- um arquivo iTunesMetadata.plist, que contém informações do desenvolvedor e do aplicativo (nome, data de lançamento etc)
- um arquivo iTunesArtwork, que define o ícone do app na App Store
- uma pasta chamada Payload, contendo diversos recursos necessários ao funcionamento do aplicativo, como assets, arquivos .json, entre outros.

A plataforma oficial de aplicativos da Apple é a App Store, lançada em julho de 2008 com mais de 500 aplicativos [58]. Já passou por uma quantidade de cerca de 2.2 milhões de aplicativos em 2017 [59], mas em 2020 este número diminuiu para cerca de 1.85 milhão de aplicativos [60], após a Apple iniciar um processo de remoção de antigos aplicativos

que não funcionavam como esperado ou que não seguiam as diretrizes estabelecidas por ela.

Até 2014, aplicativos para iOS eram escritos majoritariamente na linguagem de programação Objective-C [61], uma linguagem desenvolvida em 1984 e considerada bastante estável e madura. Em 2014, entretanto, a Apple lançou o Swift, uma linguagem desenvolvida especificamente para este sistema, com a promessa de ser uma ferramenta de fácil aprendizado, moderna, projetada para segurança, rápida e poderosa [62].

A pesquisa anual de 2020 do StackOverflow sobre a popularidade de linguagens de programação mostrou Swift com 59.9% de votos como linguagem mais amada, e Objective-C com apenas 23.4%. Além disso, 76.6% dos usuários que participaram desta pesquisa disseram que utilizavam Objective-C nos seus projetos, mas não planejavam utilizar mais futuramente, o que mostra Swift como uma linguagem em crescimento [63].

3.3 Desenvolvimento nativo x multiplataforma

Tendo em vista que não existe apenas um sistema operacional que englobe todos os dispositivos móveis atualmente, é natural que aplicativos mobile precisem ser adaptados para rodar em cada sistema, assim como acontece com programas e sistemas operacionais de computadores.

Para isso, desenvolvedores precisariam programar o mesmo aplicativo em diferentes linguagens: uma para cada sistema que quisessem abarcar. Essa realidade é bastante comum para aqueles que querem alcançar um número maior de usuários para suas aplicações.

Neste contexto, surgiram soluções para facilitar e baratear o desenvolvimento de aplicativos: tecnologias que tentam, com um único código, desenvolver para mais de um sistema operacional. Assim, surgem as noções de desenvolvimento nativo e desenvolvimento híbrido – ou multiplataforma.

Desenvolvimento nativo

Aplicativos desenvolvidos de forma nativa são aplicativos que foram escritos especificamente para uma determinada plataforma, para um determinado sistema operacional.

Estas aplicações utilizam uma linguagem característica definida pelo desenvolvedor do SO: aplicativos para Android são escritos em Java, Kotlin, entre outros; e aplicativos para iOS são escritos em Objective-C ou Swift.

Este tipo de desenvolvimento possui diversas vantagens, como por exemplo: melhor performance, maior segurança, mais fluidez na interatividade e melhor consistência visual do aplicativo com o sistema [64].

Como pontos negativos, entretanto, pode-se pontuar um aumento considerável do tempo de desenvolvimento da aplicação, pela necessidade de se criar um código para cada plataforma a que se destina; e uma maior dificuldade de encontrar desenvolvedores específicos para cada plataforma [64].

Somado a isso, o desenvolvimento de aplicativos para iOS ainda é limitado a usuários de computadores Mac, por um ambiente de programação de aplicativos iOS exclusivo, o XCode, tornando o desenvolvimento mais trabalhoso e mais custoso, considerando os preços de produtos da Apple [65].

Desenvolvimento multiplataforma

Aplicativos desenvolvidos de forma híbrida, ou multiplataforma, são aplicativos que foram projetados para rodar em diferentes plataformas utilizando o mesmo código, ou seja, sem que fosse necessário realizar ajustes para as peculiaridades de cada sistema operacional [65].

Este tipo de desenvolvimento ainda apresenta alguns problemas, como a inconsistência entre a comunicação de componentes nativos e não nativos, que gera aplicativos que podem ter sua performance afetada dependendo do sistema operacional onde está rodando [64].

Apesar disso, o desenvolvimento multiplataforma permite o alcance de um número consideravelmente maior de usuários e reduz bastante o tempo e custo de desenvolvimento de um aplicativo. O código é, em grande parte, reutilizável, e se torna mais fácil de manter e organizar [64].

Atualmente, vários são os *frameworks* que suportam desenvolvimento híbrido de aplicativos, com destaque para:

- React Native;
- Xamarin;
- Flutter;
- Adobe PhoneGap;
- Ionic.

3.4 Tecnologias Utilizadas

Levando em consideração que uma das metas do presente trabalho é atingir uma maior quantidade de usuários em variados contextos com um tempo limitado de produção aos

dois períodos letivos do desenvolvimento de um projeto final de conclusão de curso, utilizar um método de desenvolvimento multiplataforma foi a escolha mais adequada para o sucesso e universalidade do software.

Para a escolha de qual *framework* multiplataforma utilizar, foram consideradas três opções: React Native, Xamarin e Flutter.

O Xamarin, baseado em C#, é o mais antigo destes *frameworks* e oferece um bom desempenho tanto para Android quanto para iOS, além de possuir uma grande base de usuários em diversos setores. Contudo, o escasso acesso a bibliotecas *open-source* e a demora na atualização da plataforma, no que diz respeito à compatibilidade com as últimas versões dos sistemas operacionais, tornam o Xamarin uma tecnologia em desuso, tendo perdido boa parte de seus usuários nos últimos anos [66].

O React Native, apesar de ser um *framework* relativamente novo (lançado em março de 2015), é utilizado no desenvolvimento de diversos aplicativos famosos, como o Instagram, o Facebook e o Skype [67]. Um ponto negativo dele, entretanto, é que algumas funcionalidades não possuem implementação no React Native: por exemplo, para usar a câmera ou o acelerômetro em um aplicativo, será necessário utilizar componentes nativos dos sistemas, ou seja, haverá códigos diferentes para cada SO. Além disso, é conhecido por ter um tempo de debug grande, principalmente no Android, e por não possuir consistência de periodicidade nos updates [64].

Por fim, o Flutter, o *framework* mais novo da lista (lançado em 2017 pelo Google), foi incluído nas possibilidades por possuir alguns diferenciais importantes que o destacam frente a outros *frameworks* para este projeto. Dentre as vantagens, que serão melhor explicadas na próxima seção, pode ser citado que Flutter utiliza sua própria biblioteca de *widgets* para o desenvolvimento da interface, não sendo realizado nenhum tipo de conversão para o sistema a qual se destina: isso gera mais fluidez, consistência e maior controle da interface de usuário para o desenvolvedor [68].

Tendo em vista a preferência pela universalidade do aplicativo, somada às facilidades de se desenvolver em Flutter e ao fato de ser um *framework* oficial do Google, esta foi a ferramenta de desenvolvimento escolhida para a elaboração do software, juntamente com a linguagem de programação Dart. Estes são os focos das duas seções a seguir.

3.4.1 Dart

Dart é uma linguagem de programação que foi criada pelo Google em 2011 com o intuito de substituir o JavaScript para o desenvolvimento de scripts de páginas web. A linguagem segue o paradigma de orientação a objetos e é fortemente tipada, tendo nascido com os objetivos de ser [69]:

- Estruturada, mas flexível para programação web;
- Ser de fácil aprendizado para desenvolvedores;
- Entregar uma alta performance nos navegadores web modernos

A linguagem, entretanto, não conseguiu se estabelecer como substituta do JavaScript, e atualmente sua utilização se foca mais no âmbito do desenvolvimento de aplicações web e mobile em conjunto com o *framework* Flutter.

Para o desenvolvimento de aplicações mobile e web, Dart ainda conta com uma característica que lhe dá uma vantagem sobre as demais linguagens de programação do nicho. Ela pode ser compilada em ahead-of-time (AOT) e just-in-time (JIT).

A compilação AOT ocorre na hora de publicação do aplicativo (*deploy*), e é feita diretamente para ARM nativo, possibilitando ao app a performance de uma aplicação nativa. Por outro lado, a compilação JIT é feita diretamente para o dispositivo, com a aplicação em execução, o que possibilita um desenvolvimento muito mais rápido por meio de hot-reload [70].

3.4.2 Flutter

O Flutter é o SDK de desenvolvimento *mobile*, *web* e *desktop* do Google, projetado para desenvolver aplicações bonitas e nativamente compiladas com um único código [71]. Apesar de ser um *framework* bem recente, já possui uma grande base de usuários e é utilizado no desenvolvimento de grandes projetos, como no aplicativo do Nubank e da BMW [72].

Uma das características do desenvolvimento em Flutter que mais chama a atenção é a funcionalidade de *hot-reload*: a re-renderização de uma tela passa a ser feita em milissegundos, fornecendo um *feedback* ao vivo do que está sendo programado, em contraposição a outras ferramentas de desenvolvimento onde as aplicações precisam ser completamente recompiladas e carregadas no dispositivo depois que mudanças são realizadas, o que pode levar alguns minutos [68].

Outro fator a se considerar no desenvolvimento com Flutter é a consistência na interface do usuário (UI) entre diferentes plataformas, alcançada por meio da biblioteca de componentes de UI disponibilizada pelo próprio SDK. Além disso, como o Flutter não passa o papel de mostrar a interface do aplicativo para a plataforma nativa, a quantidade de problemas de compatibilidade entre recursos nativos e não nativos é reduzida e o *design* se torna mais estável entre plataformas [68].

A interface de usuário do Flutter é baseada no uso de *widgets*, que podem ser definidos como componentes gráficos que descrevem como a aplicação vai ser mostrada dependendo de sua configuração [73]. A biblioteca de *widgets* a ser utilizada neste projeto segue as

diretrizes de *design* do Material Design, mas existe também uma biblioteca projetada baseando-se no *design* iOS.

Widgets

Em Flutter, tudo são *widgets*: desde elementos da interface - como textos, imagens, e formas; até definições de layout - como margens, colunas, linhas e grades. Estes elementos se relacionam entre si por meio de uma hierarquia pai/filho e formam uma árvore de *widgets* que engloba o aplicativo inteiro.

Quando um novo *widget* é criado, ele é inserido no local adequado da árvore de *widgets*. No início do programa, a função *runApp()* gera a árvore utilizando o *widget* que foi passado a ela como a raiz da árvore, e os *widgets* subsequentes serão seus filhos, definindo a hierarquia da interface.

Um *widget* é, efetivamente, uma classe que pode receber como atributos:

- propriedades relacionadas a características visuais do *widget*, como *cor*, *tamanho* e *forma*;
- propriedades relacionadas ao funcionamento do *widget*, como *callbacks* a serem chamadas ao clicar num *widget* ou pressioná-lo;
- um *widget* filho (ou uma lista de *widgets* filhos).

Existem dois tipos de *widgets*: *StatelessWidgets* (ou *widgets* sem estado) e *StatefulWidgets* (ou *widgets* com estado). O estado de um *widget* é definido como uma informação que pode ser alterada após a inserção do *widget* na árvore de *widgets*. Por exemplo, se quisermos alterar dinamicamente a cor do texto de um botão ao clicarmos nele, depois de ele já estar presente na tela, precisaremos de um estado para a cor. Todas as alterações de estado devem ser notificadas ao *framework* por meio da chamada da função *setState*.

A função de hot reload, ou live reload, funciona verificando a diferença entre o estado antigo e o novo estado do *widget*, aplicando apenas as diferenças à tela. Isso diminui o custo e o tempo necessário para mostrar as alterações na interface de usuário, reduzindo o tempo de desenvolvimento do aplicativo.

Configurações Multiplataforma

O Flutter, apesar de ser um *framework* que suporta o desenvolvimento multiplataforma, define algumas configurações que precisam ser adaptadas para a correta execução em cada sistema operacional.

No caso deste projeto, o desenvolvimento do software foi centrado na plataforma Android, tendo sido completamente configurado e otimizado para seu uso. Esta escolha se

deu pela facilidade de obtenção de um aparelho Android para testes e pela disponibilidade do ambiente de desenvolvimento Android (a IDE Android Studio) gratuitamente para os principais sistemas operacionais.

Já para a consolidação do software na plataforma iOS, é necessário que as configurações sejam adaptadas e o aplicativo seja testado utilizando plataformas de desenvolvimento específicas da Apple, como o XCode, de uso limitado a seus computadores (Mac), e um iPhone. A indisponibilidade destes recursos impossibilitou a configuração do software para iOS, apesar de o código em si já ser compatível com o sistema operacional.

3.4.3 Ambiente de Desenvolvimento

O ambiente de desenvolvimento integrado (IDE) utilizado para a elaboração do presente projeto foi o Android Studio, ambiente oficial para desenvolvimento de aplicativos Android. É baseado na versão *open-source* do IntelliJ IDEA e conta com diversas ferramentas para otimizar o desenvolvimento, dentre as quais [74]:

- um emulador Android nativo para o desenvolvimento sem um dispositivo físico;
- ferramentas próprias de *debug*;
- auto-completar de trechos de código reutilizáveis (*snippets*) e de bibliotecas importadas;
- ferramentas de análise de código para reconhecer problemas de performance, usabilidade, compatibilidade de versão etc;
- integração com ferramentas de versionamento de código;
- auto-formatação de arquivos para um padrão de indentação.

Além disso, a IDE conta com um gerenciador de *plugins* que facilita a instalação de todas as ferramentas necessárias para o desenvolvimento de um aplicativo.

3.4.4 SQL

Um sistema de banco de dados, segundo Date [75], é um sistema de armazenamento de dados baseado em computador, um sistema cujo objetivo é registrar e manter informações.

Sendo assim, é natural que no contexto de desenvolvimento de aplicativos surja a necessidade de utilização de um banco de dados. Para o presente projeto, após analisar os requisitos do aplicativo, foi decidido implementar um banco de dados para armazenar informações das atividades do app, bem como permitir a adição de novas atividades pelo usuário.

Dentre as opções no que diz respeito à estrutura dos dados nos bancos de dados, duas se destacam: bancos relacionais e bancos não relacionais. O banco de dados relacional se organiza em tabelas: cada tabela representa uma entidade e possui colunas, representando atributos desta entidade, e linhas, representando os registros, instâncias da entidade [76]. No modelo relacional, as tabelas são bem definidas e fixas, conferindo integridade aos dados.

Já no banco de dados não-relacional, os dados podem ser armazenados como documentos, família de colunas, chave-valor e grafos. A diferença principal para o modelo de banco relacional é que os dados não precisam possuir um esquema fixo e há mais liberdade na montagem do banco. São bastante utilizados como soluções altamente escaláveis (para bancos de dados grandes, por exemplo) e bastante rápidos [76].

Para o aplicativo Prevenir, foi considerado o uso do Cloud Firestore, um banco de dados não-relacional criado pelo Firebase, do Google, pela facilidade de integração com o Flutter. Entretanto, por ser uma solução de banco de dados na nuvem, o aplicativo precisaria de internet para gerenciar o banco. Tendo em vista a necessidade de universalidade do software, essa solução foi descartada.

Com o intuito de permitir o uso do aplicativo sem conexão à rede e de conferir uma maior confiabilidade dos dados armazenados, decidiu-se por um banco de dados local. Assim, foi escolhido o sistema de gerenciamento de banco de dados SQLite, biblioteca que implementa um banco relacional extremamente leve e flexível [75]. É um sistema de fácil configuração, administração e de baixo consumo de recursos [77], e por isso é bastante utilizado no desenvolvimento de aplicativos [78], onde os recursos são limitados e o banco geralmente não toma proporções tão grandes.

Para a integração do Flutter com o SQLite, foi escolhido um plugin com suporte para iOS e Android chamado *sqflite*. Este plugin possui uma configuração extremamente simples, bem como diversos helpers disponíveis para inserção, atualização, busca e deleção de dados no banco [79].

Nesta seção, foram explicadas as ferramentas utilizadas na implementação do presente projeto. O aplicativo foi desenvolvido na linguagem Dart, em conjunto com o *framework* Flutter, no ambiente de desenvolvimento Android Studio; e o banco de dados foi implementado localmente com SQLite. A seção a seguir abrangerá definições acerca do padrão de projeto utilizado para estruturar o código desenvolvido.

3.5 BLoC

Na elaboração de um projeto, a organização é uma característica extremamente importante que auxilia na reusabilidade e testabilidade do sistema. Por isso, um conceito bastante estudado na área de desenvolvimento de sistemas é o padrão de projeto, um modelo pré-estabelecido que diz respeito, resumidamente, à organização de arquivos nos diretórios e à separação de funcionalidades do sistema.

Para o desenvolvimento do presente aplicativo, o padrão de projeto escolhido foi o BLoC, um padrão desenvolvido pelo Google e anunciado na conferência Google I/O de 2018. Surgiu a partir da necessidade de separar a interface do usuário (apresentação) e lógica de negócio de um sistema e foi projetado para ser fácil, poderoso e testável [80]. Em conjunto com Flutter, é bastante eficiente para realizar o gerenciamento de estados de *widgets* e para reaproveitar lógicas semelhantes em diferentes páginas.

O padrão BLoC se divide em três camadas: a interface (UI), o bloc (lógica de negócio) e a camada de dados [81].

Interface (UI) é a camada que contém a interface visual do aplicativo, cada uma das páginas com seus respectivos *widgets*. É responsável por realizar a comunicação do aplicativo com o usuário por meio do recebimento de eventos de entrada (movimentos de deslizar, navegação entre páginas, toques em botões etc).

Bloc é a camada que contém a lógica de negócio e que recebe os eventos da interface, emitindo estados como resposta à ela.

Camada de dados é a camada responsável por recuperar dados de um determinado lugar, como por exemplo um banco de dados. Pode ser dividida entre duas partes: repositório e provider.

O fluxo de informação ocorre através de eventos e estados. Um evento é uma ação que deve acarretar em uma resposta do aplicativo: por exemplo, tocar em um botão (evento) acarreta no redirecionamento de uma página (estado que descreve o redirecionamento).

Em um exemplo mais complexo: para ver uma lista de dados que estão armazenados num banco de dados, o usuário navega até a página da lista, emitindo um evento de início para o bloc da página. O bloc verifica o evento que chegou, emite o estado de carregamento para a interface e pede à camada de dados a lista de dados do banco. A camada de dados coleta essa informação do banco e devolve ao bloc, que por sua vez emitirá para a interface um estado de carregamento finalizado, passando a ela a lista de dados que foi pedida inicialmente.

No Flutter, a implementação do BLoC pode ser feita manualmente, com o uso dos *widgets* StreamBuilder e FutureBuilder, comumente utilizados para o manuseio de dados

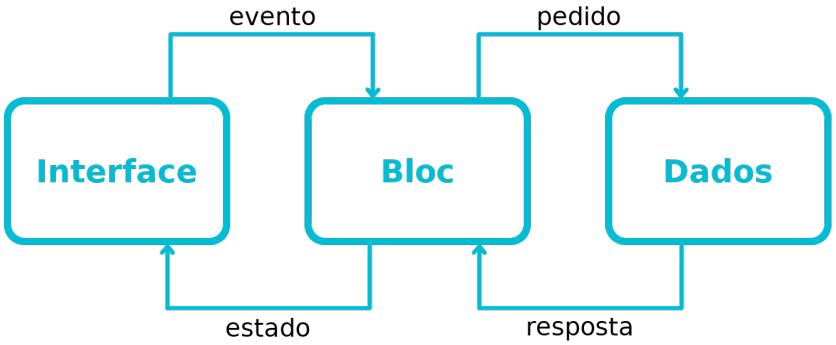


Figura 3.3: Fluxo de dados no padrão BLoC.

assíncronos; ou com auxílio do pacote flutter_bloc, que implementa a classe Bloc e alguns *widgets*, como BlocProvider, BlocBuilder, BlocListener e BlocConsumer [82]. Para o presente projeto, foi decidido pelo uso do pacote flutter_bloc.

Utilizando o flutter_bloc, cada bloc do projeto (geralmente, um por página de interface) deve estender a implementação da classe abstrata *Bloc*. Essa classe define um método *mapEventToState*, que deve ser sobrescrito definindo o mapeamento entre eventos e estados e fazendo as devidas chamadas à camada de dados, se necessário.

O Bloc deve ser instanciado utilizando o *widget BlocProvider*, utilizado como injeção de dependência para que uma única instância do bloc possa ser acessada pelos *widgets* da sub-árvore de filhos.

A interface que precisa reagir a estados deve ser construída dentro de um *BlocBuilder*, um *widget* que toma como parâmetros um bloc e uma função que construirá a resposta (builder), retornando um widget. É parecido com a implementação de StreamBuilder, mas reduz a quantidade de códigos de configuração desnecessários (*boilerplate*). Na função *builder*, os parâmetros *context* e *state* estão disponíveis, sendo possível implementar facilmente uma interface para cada estado com uma simples comparação if-else do *state*.

O *widget BlocListener* é útil para funcionalidades que devem acontecer apenas uma vez por mudança de estado, como por exemplo para redirecionar páginas, mostrar *snackbars*, entre outros. Recebe como parâmetros um bloc e uma função que responderá aos estados (listener), análoga à função builder do BlocBuilder.

O *BlocConsumer* é uma junção do BlocBuilder com o BlocListener, tomando como parâmetros um bloc, uma função builder e uma função listener.

3.6 Metodologia de Desenvolvimento

Na elaboração de grandes projetos, é comum a utilização de metodologias para guiar e organizar o processo de desenvolvimento. De maneira geral, as metodologias de desenvolvimento de software se dividem entre duas categorias: metodologias tradicionais e metodologias ágeis.

No desenvolvimento pautado por metodologias tradicionais, o projeto precisa ter o escopo fechado desde o início da implementação do software, não havendo, portanto, muito espaço para mudanças [83]. Já as metodologias ágeis são bastante interessantes para projetos que não possuem um escopo bem definido desde o início e onde a flexibilidade às mudanças é uma característica importante para o projeto [84].

Para a elaboração do aplicativo Prevenir, o escopo foi sendo incrementado progressivamente, à medida que novas atividades foram projetadas com auxílio dos professores especialistas em Educação Especial. Levando este fator em consideração, decidiu-se pela adoção de uma metodologia ágil para o desenvolvimento do projeto.

A metodologia implementada se baseou na metodologia ágil Kanban, que possui fácil implementação e boa organização visual. O Kanban tem por premissa a criação de um método visual de progresso de atividades, separando as demandas de implementação entre três categorias: pendente, em andamento e concluído [85].

Para a implementação da metodologia, foi utilizada a ferramenta Trello, um sistema que permite a criação de quadros de gerenciamento de projetos utilizando listas e cartões para organizar as atividades. Ao longo do desenvolvimento do presente projeto, as demandas foram sendo escritas em cartões e divididas entre três listas predefinidas: “A fazer”, “Em andamento” e “Concluído”.

A utilização dessa forma de organização permitiu a visualização clara das demandas que ainda não haviam sido atendidas no aplicativo e contribuiu bastante para uma melhor visão geral do andamento do projeto.

Capítulo 4

O Software Prevenir

O software elaborado no presente projeto consiste em um aplicativo que visa contribuir com o processo de aprendizagem de pessoas com Transtorno do Espectro Autista, alertando para os perigos existentes dentro de casa e objetivando a prevenção de possíveis acidentes domésticos. Os requisitos educacionais foram especificados pela psicopedagoga Maraísa Helena Borges Estevão Pereira, com larga experiência em educação especial: escolha da temática, concepção de telas e lições e realização de testes do produto.

O aplicativo pode ser integrado a um hardware complementar, o Cubinho [1], com o objetivo de oferecer uma variedade maior de motivadores para seu uso. O hardware é um robô em forma de cubo, construído utilizando a placa com microcontrolador Arduino, e emprega sons, luzes, movimento e imagens para encorajar o uso do aplicativo para a pessoa com TEA, funcionando como um reforçador para os acertos nas atividades.

O objetivo deste capítulo é apresentar o aplicativo Prevenir desenvolvido, explicando sua estrutura, organização, conteúdo e modo de uso.

4.1 As Atividades

As atividades do aplicativo foram projetadas por professores especialistas em Educação Especial, que definiram como deveria ser cada uma delas a partir das necessidades do público-alvo do software.

Foram definidos 3 tipos de atividade, todos com o intuito principal de apresentar os perigos de um objeto de casa. Os tipos das atividades se dividem entre:

- **Atividade 1**: é dividida em quatro partes, cada uma contendo uma instrução e com a intenção de manter o aluno longe do objeto de perigo.

- **Atividade 2**: é dividida em quatro níveis de dificuldade, com o intuito de diferenciar objetos entre seguros e não-seguros arrastando os objetos seguros para uma área determinada.
- **Atividade 3**: é dividida em quatro níveis de dificuldade, com intuito de diferenciar objetos entre seguros e não-seguros tocando nos objetos seguros.

O banco de dados do aplicativo foi inicialmente populado com objetos seguros e não-seguros, criando uma base inicial de atividades padrão. É possível adicionar novas atividades criando objetos personalizados a partir de um formulário, definindo textos de instrução e selecionando uma imagem do dispositivo (da câmera ou da galeria), com o objetivo de aumentar o alcance do software e a familiaridade do objeto de perigo com cada aluno especificamente. Além disso, é também possível habilitar e desabilitar os objetos no aplicativo, sejam eles personalizados ou não, para que não apareçam nas atividades 1, 2 e 3.

4.1.1 Especificações do Aplicativo

As telas do aplicativo seguiram especificações fornecidas por professores especialistas em Educação Especial, pensadas para tornarem o software o mais acessível e agradável possível, principalmente para pessoas com TEA. Dentre os requisitos, podem ser citados:

- As atividades devem ser o mais simples possível, removendo elementos que possam distrair o aluno;
- As atividades devem ter o fundo branco e instruções simples e diretas, escritas em preto, com todas as letras maiúsculas, no topo da atividade;
- Ao acertar uma atividade, o software deve retornar um *feedback* positivo, que pode variar entre: um som de campainha positivo, um vídeo de parabéns e outros *feedbacks* a partir do hardware complementar;
- O aplicativo deve ser altamente customizável, para atender às diferentes necessidades e preferências de cada aluno;
- As imagens utilizadas para os objetos no aplicativo devem ser fotos de objetos reais, para que ocorra a assimilação correta com a vida real.

4.2 Arquitetura do Software

Nesta seção, será apresentada a arquitetura do software, mais especificamente a estrutura que seguiu o banco de dados utilizado; a organização dos arquivos do código-fonte do

aplicativo, levando em conta o padrão BLoC; e uma explicação resumida da disposição das classes e a relação entre elas.

4.2.1 Estrutura do Banco de Dados

Para o banco de dados do aplicativo aqui descrito, foi necessária a criação de duas tabelas: ActivityObject e Activity, onde ActivityObject diz respeito aos objetos utilizados nas atividades, e Activity diz respeito às atividades em si. O modelo relacional do banco de dados implementado pode ser visualizado na Figura 4.1.

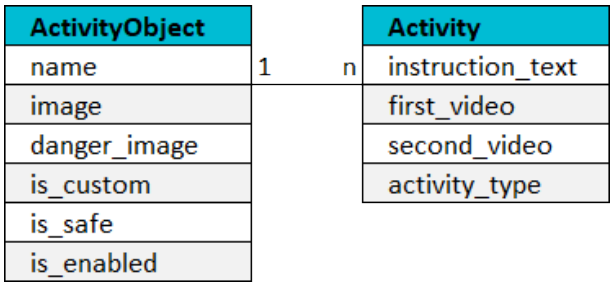


Figura 4.1: Modelo Relacional do Banco de Dados.

Para o objeto, foram armazenados os seguintes campos:

- **name**: armazena o nome do objeto como tipo TEXT;
- **image e danger_image**: armazena a imagem do objeto e sua imagem de perigo (por exemplo, fogão e fogão aceso) como tipos TEXT;
- **is_custom**: define uma flag como tipo INTEGER para diferenciar um objeto personalizado de um objeto padrão (onde 0 significa objeto padrão e 1 significa objeto customizado);
- **is_safe**: define uma flag como tipo INTEGER para diferenciar um objeto seguro de um objeto perigoso (onde 0 significa objeto perigoso e 1 significa objeto seguro);
- **is_enabled**: define uma flag como tipo INTEGER para habilitar ou desabilitar um objeto das atividades (onde 0 significa desabilitado e 1 significa habilitado).

Já para a atividade, foram armazenados:

- **instruction_text**: armazena a instrução da atividade como tipo TEXT;
- **first_video**: armazena a URL do primeiro vídeo da atividade como tipo TEXT;
- **second_video**: armazena a URL do segundo vídeo da atividade como tipo TEXT;

- **activity_type**: armazena o tipo da atividade como tipo TEXT;
- **activity_object_id**: define o relacionamento entre uma atividade e seu objeto, funcionando como FOREIGN KEY.

Para popular o banco de dados inicialmente, foi realizada uma análise de objetos domésticos em conjunto com professores especialistas em Educação Especial, com o intuito de criar uma lista definindo objetos considerados seguros e objetos considerados perigosos. Dentre os objetos definidos, pode-se citar (Figura 4.2):

OBJETOS SEGUROS	OBJETOS NÃO SEGUROS
Sofá	Chaleira
Cama	Cafeteira
Mesa	Liquidificador
Cadeira	Fogão
Escova de dentes	Máquina de lavar
Toalha	Torradeira
Travesseiro	Ferro elétrico
Pente de cabelo	Micro-ondas
Copo de plástico para água	Forno

Figura 4.2: Lista de objetos separados entre seguros e não-seguros.

4.2.2 Estrutura dos Arquivos

Em projetos em Flutter, os arquivos relacionados ao código-fonte do aplicativo ficam na pasta lib. Nela, foram criados diretórios para melhor organizar o código do software tendo em mente o padrão de projeto BLoC escolhido. Os arquivos ficaram estruturados da seguinte maneira:

Na pasta **blocs**, para cada bloc foi criado um diretório contendo 3 arquivos: nomeDoBloc_bloc.dart, nomeDoBloc_event.dart e nomeDoBloc_state.dart. Essa organização foi escolhida para atingir uma melhor organização e separação entre o bloc, seus eventos e seus estados.

Na pasta **database**, foram definidas as configurações do banco de dados para o pacote sqflite, como funções de criação do banco, das tabelas a serem utilizadas, e seeds para popular o banco inicial.

Na pasta **models**, foram especificadas as models para cada tabela do banco de dados, para tornar possível a tradução de um objeto buscado no banco (em formato chave-valor,

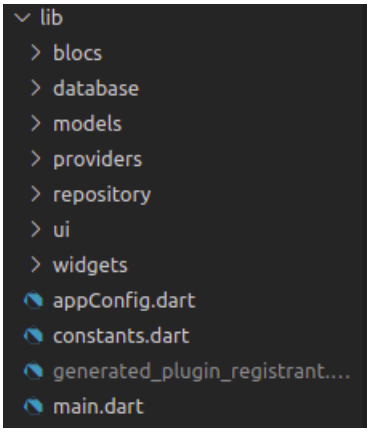


Figura 4.3: Estrutura de arquivos dentro da pasta lib.

como um json) para uma classe definida com atributos em Dart, facilitando o acesso a estes valores.

Na pasta **providers**, foi criado um arquivo para cada model como um DAO (Data Access Object), contendo funções que realizam operações genéricas de CRUD (criação, visualização, alteração e exclusão) no banco de dados.

A pasta **repository** contém um arquivo repositório para cada model, com funções mais específicas que chamam as funções genéricas do provider.

A pasta **ui** inclui todas os arquivos relacionados à interface do aplicativo. As páginas relacionadas às atividades em si ficaram dentro de um diretório "activities"; e as demais páginas (página inicial, configurações, menus, entre outros) ficaram soltas na pasta ui.

Na pasta **widgets** foram definidos *widgets* customizados que seriam utilizados em mais de um lugar do projeto, como o *widget OurVideoPlayer*, que define um player de vídeo já configurado da forma desejada; o *widget OurBottomNavigationBar*, que define a barra de navegação inferior presente em todas as telas das atividades; entre outros.

O arquivo **appConfig** define configurações que são rodadas no início do aplicativo: por exemplo, coletando informações do tamanho da tela para que tamanhos relativos de tela (porcentagens) possam ser utilizados posteriormente no aplicativo.

O arquivo **constants** armazena constantes utilizadas durante o aplicativo inteiro, como informações de estilo de botões, códigos de comunicação com o hardware complementar, entre outros.

4.2.3 Estrutura das Classes

Para cada tela do aplicativo, foi criada uma classe estendendo a implementação do *Stateless Widget* ou do *Stateful Widget* do Flutter. No Anexo I, pode ser visto um diagrama que apresenta a estrutura das classes mais relevantes do aplicativo.

A lógica de funcionamento das atividades, por ter sido implementada no padrão BLoC, é baseada em eventos e estados, ou ações e respostas. Cada atividade possui seu próprio bloc, seus tipos de evento e seus tipos de estado.

Os eventos mais importantes das atividades dizem respeito a respostas corretas e incorretas à lição. A classe principal da atividade, que é onde ficam os *widgets* a serem mostrados na tela, é implementada dentro de um *BlocConsumer*, o *widget* que escuta a mudança de estados do bloc. Assim, o aplicativo pode mostrar uma tela diferente baseado no estado em que o bloc se encontra.

Quando o usuário acerta ou erra uma atividade, um evento é gerado para o bloc. O bloc mapeia, então, o evento para um estado determinado (estado de atividade acertada ou errada, para simplificar) e o BlocConsumer mostra os *widgets* relativos ao estado atingido, através do parâmetro builder, ou reage às mudanças navegando para outra tela, através do parâmetro listener.

4.3 Telas do Aplicativo

Durante o desenvolvimento do projeto, foram utilizados vídeos ilustrativos no lugar dos vídeos instrucionais definitivos, que ainda não estavam prontos. As telas aqui mostradas foram capturadas durante a fase de desenvolvimento, e por isso ainda não possuem os vídeos definitivos.

4.3.1 Tela Inicial

A tela inicial do aplicativo (Figura 4.4) mostra um vídeo de boas-vindas com um breve resumo explicativo das funcionalidades, do objetivo e de possíveis configurações do software Prevenir. Na parte inferior do vídeo, o botão “Seguir para a atividade” redireciona para o menu principal.

4.3.2 Menu Principal

O menu principal (Figura 4.5) é a tela que mostra as atividades do aplicativo, um botão para criar e personalizar objetos e um botão para acessar as Instruções de Uso, uma tela que traz um texto explicativo sobre como utilizar o software.

Existe, ainda, na tela, um botão no canto inferior direito com um ícone de configurações que redireciona para o menu de configurações, a ser acessado pelo professor ou tutor do aluno.

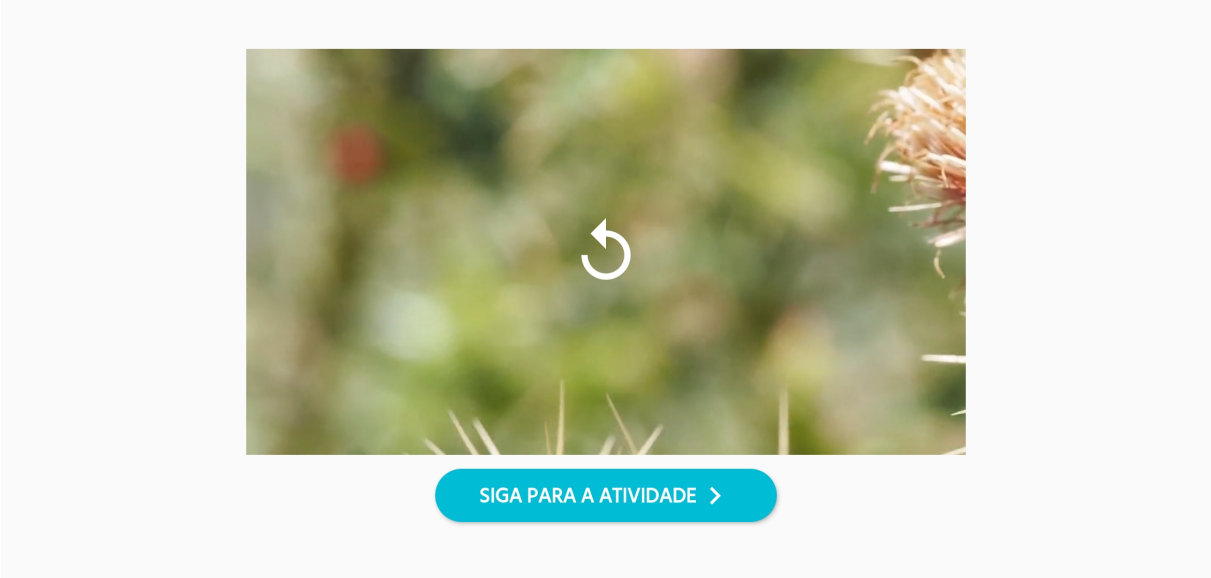


Figura 4.4: Tela inicial do aplicativo.

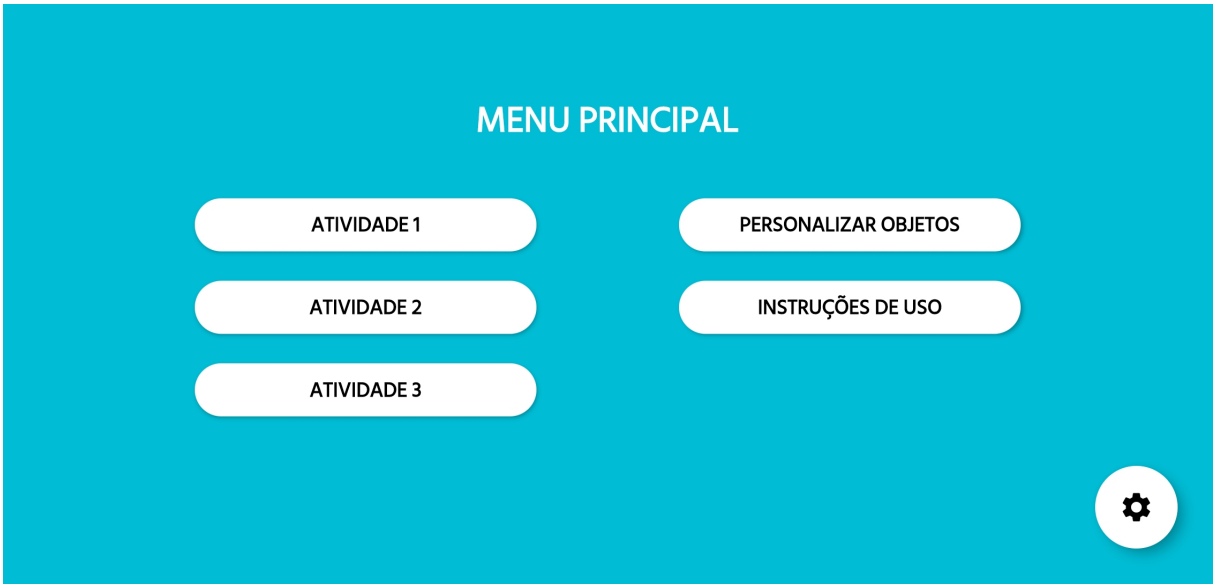


Figura 4.5: Menu principal do aplicativo.

4.3.3 Personalizar Objetos

A tela de personalizar objetos (Figura 4.6) possui uma lista com todos os objetos adicionados pelo usuário do aplicativo, sendo eles seguros ou não-seguros. Os objetos podem ser excluídos clicando no ícone de lixeira à direita. No canto inferior direito, um botão com o ícone "+" leva à página de formulário de objetos personalizados.

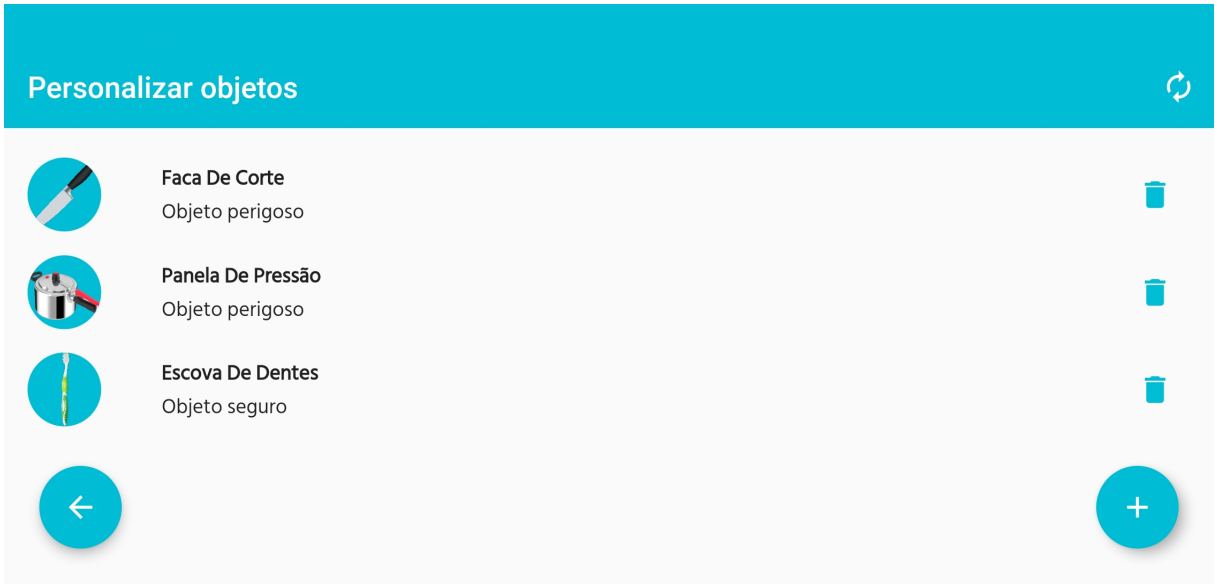


Figura 4.6: Menu de gerenciamento de objetos personalizados.

4.3.4 Formulário de Objeto Personalizado

Nesta tela (Figura 4.7), é possível preencher o formulário para adicionar um novo objeto ao banco de dados do aplicativo.

Para um objeto perigoso, é necessário preencher o nome do objeto, sua foto e deixar a caixa de seleção "Objeto seguro" desmarcada. Depois, devem ser preenchidas todas as quatro instruções da Atividade 1 (Figura 4.8). Todos os campos do formulário possuem um texto de referência a ser seguido.

Já para um objeto seguro, que só será utilizado nas atividades 2 e 3, devem ser preenchidos o nome do objeto, sua foto e marcar a caixa de seleção "Objeto seguro".

Após criar o objeto, o aplicativo redirecionará o usuário de volta para a tela de personalizar objetos.

4.3.5 Menu de Configurações do Aplicativo

Este menu (Figura 4.9) possui as configurações gerais do aplicativo. O primeiro botão, "Objetos Habilitados", navega para a tela de habilitar e desabilitar objetos no aplicativo. O segundo leva para a tela de configurações de *feedbacks* do app (vídeos e sons motivacionais); o terceiro, para a tela de configurações do hardware complementar Cubinho; e o último reseta o banco de dados para o estado inicial, fazendo uma verificação de segurança com o usuário (Figura 4.10).

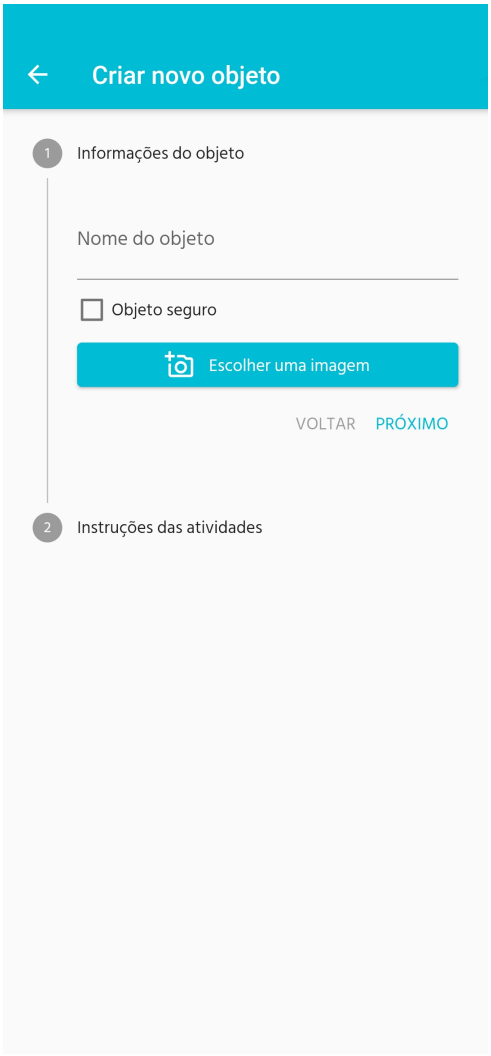


Figura 4.7: Formulário inicial de objeto personalizado.

4.3.6 Objetos Habilitados

Nesta tela (Figura 4.11), o usuário pode habilitar e desabilitar objetos para que eles não apareçam nas atividades do aplicativo. Os objetos personalizados aparecem com um ícone de engrenagem cinza depois do nome.

4.3.7 Tela de Configuração de *Feedbacks* do Aplicativo

Durante as atividades, o aplicativo possui *feedbacks* sonoros ao acertar uma resposta (som de campainha) e vídeos motivacionais entre lições. A tela de configurações dos *feedbacks* do aplicativo (Figura 4.12) é onde estes *feedbacks* podem ser ativados ou desativados.

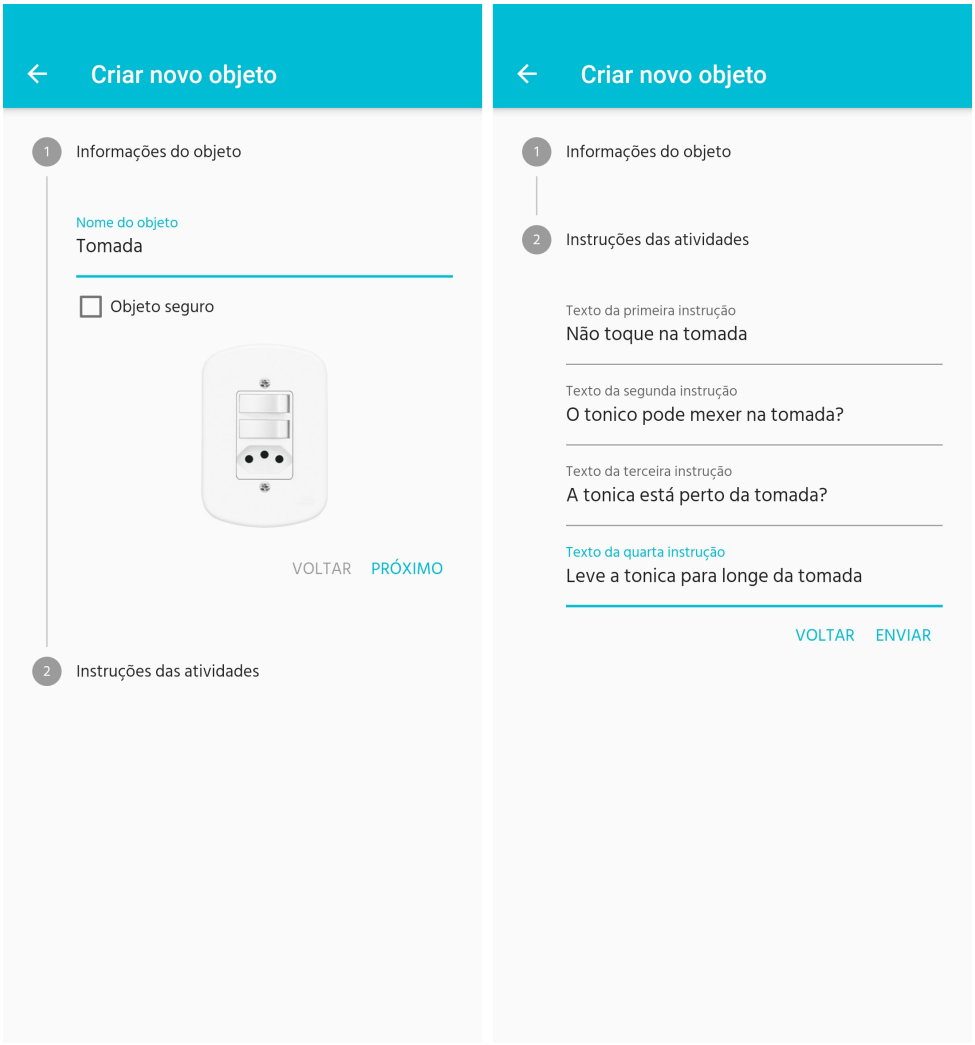


Figura 4.8: Formulário de objeto personalizado preenchido.

4.3.8 Tela de Configuração do Hardware

Esta tela (Figura 4.13) diz respeito às configurações necessárias para que o hardware complementar funcione corretamente. Primeiro, deve-se conectar o dispositivo ao Cubinho através do bluetooth ("Configurar Bluetooth") e, depois, configurar seus *feedbacks* ("Configurar *Feedback*").

Configurar bluetooth

A tela de configuração do bluetooth (Figura 4.14) existe para realizar a conexão do aplicativo com o hardware motivacional complementar, o Cubinho.

Inicialmente, a tela possui um botão para ativar ou desativar o bluetooth do dispositivo. Ao ser ativado, os dispositivos bluetooth disponíveis podem ser vistos pelo aplicativo,

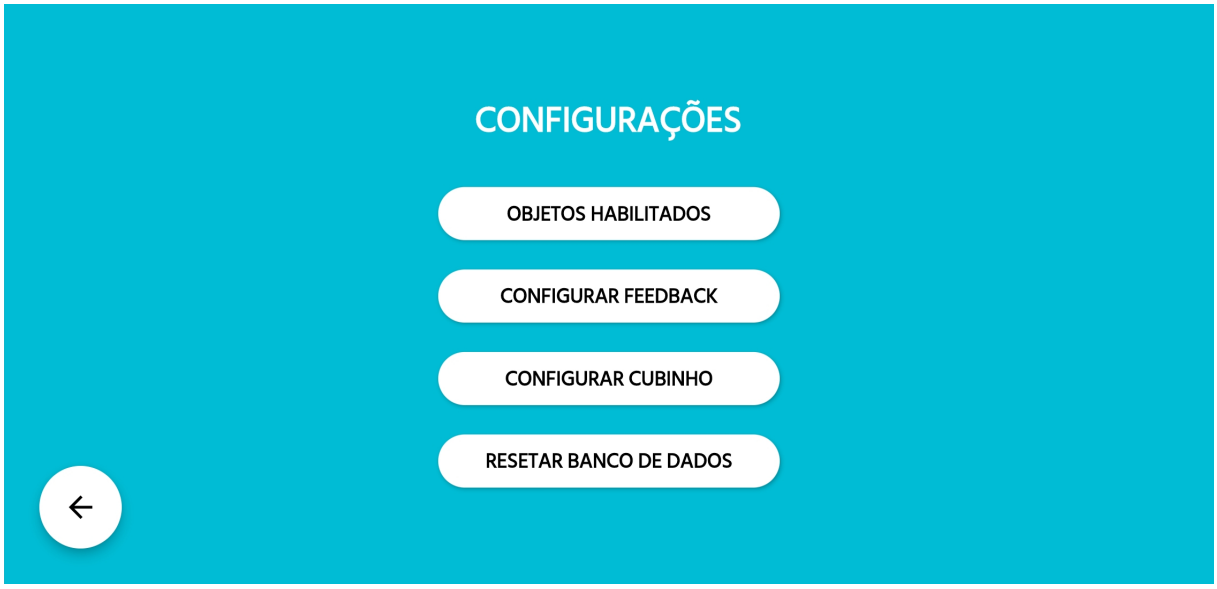


Figura 4.9: Menu de configurações do aplicativo.

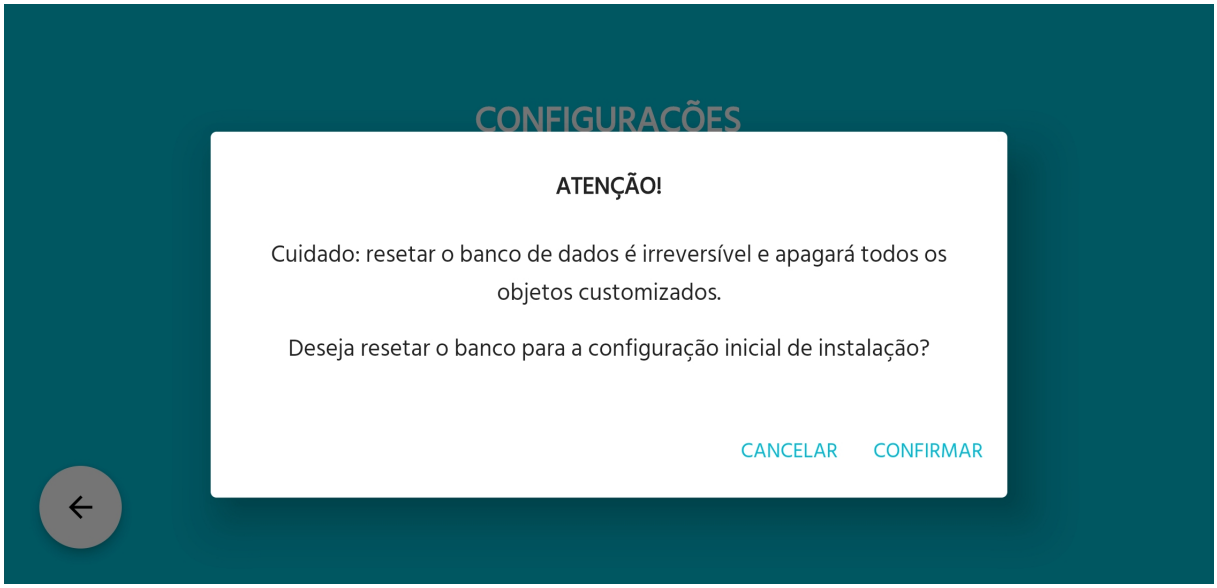


Figura 4.10: Alerta de confirmação para resetar banco de dados.

separados entre dispositivos pareados e outros dispositivos. Após parear um dispositivo, o usuário é capaz de vê-lo na lista de pareados e tocar no botão “CONECTAR” para fazer a conexão do software com o hardware complementar. É possível ver que a conexão foi bem sucedida na primeira seção da tela, “Dispositivo conectado”.

Figura 4.11: Tela de habilitar e desabilitar objetos.

Figura 4.12: Tela de configurar *feedbacks* do aplicativo.

Configurar *feedbacks*

É a tela responsável por customizar e atribuir os *feedbacks* positivos e negativos gerados pelo hardware complementar ao aplicativo (Figura 4.15). Ao clicar em "ALTERAR" em qualquer uma das seções, uma modal é aberta com todas as configurações possíveis (Figura 4.16). Após selecionar uma opção, basta clicar em "SALVAR" para guardar as preferências.

Figura 4.13: Tela de configurações do hardware.

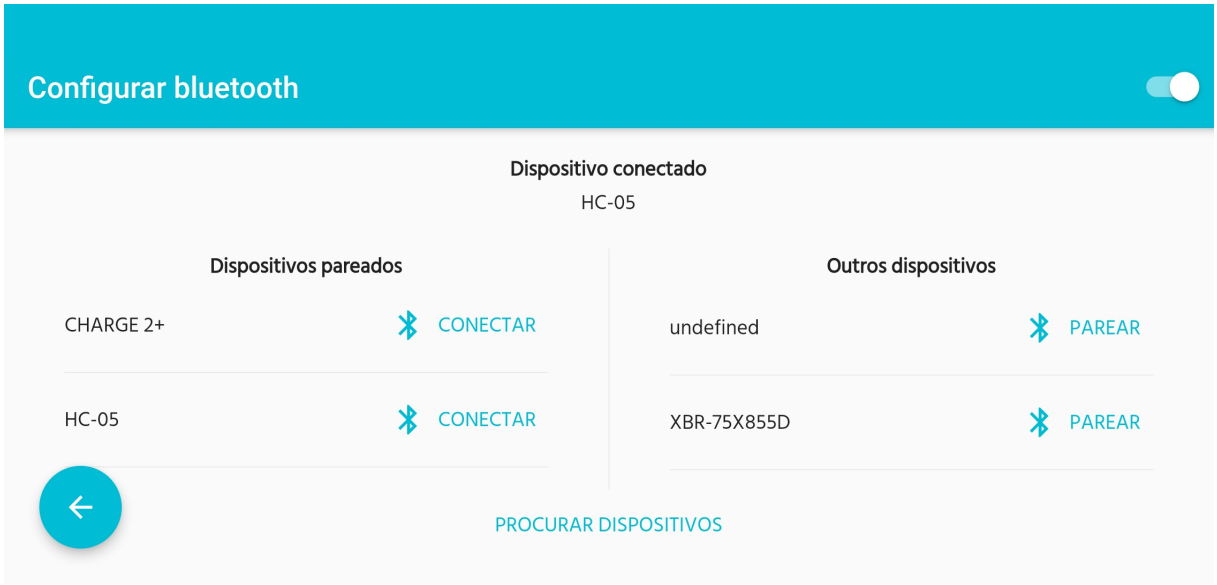


Figura 4.14: Tela de configurações do bluetooth.

4.3.9 Menu da Atividade 1

Como a Atividade 1 é uma atividade focada em um único objeto por vez, foi criado um menu para ela (Figura 4.17), contendo um botão que redireciona para o menu da Atividade 1 para objetos personalizados e botões que redirecionam para a Atividade 1 de cada objeto que vem por padrão no aplicativo.

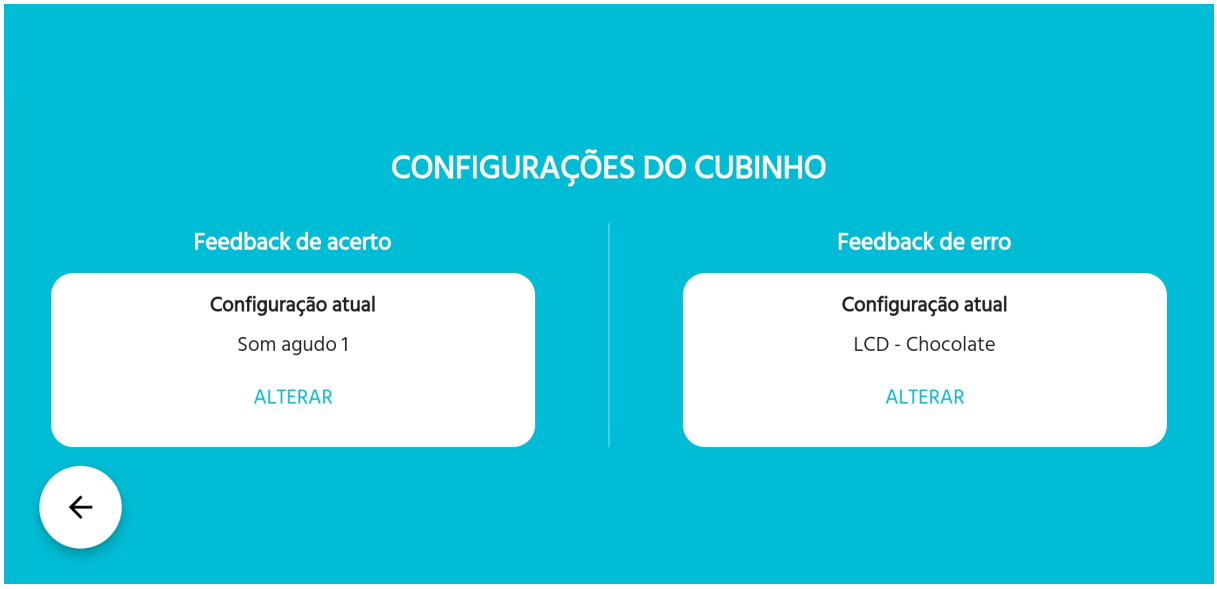


Figura 4.15: Tela de configurações do hardware complementar.

4.3.10 Menu da Atividade 1 - Objetos Personalizados

Esta tela mostra o menu da Atividade 1 para objetos personalizados (Figura 4.18) e possui um botão para a Atividade 1 de cada objeto personalizado adicionado pelo tutor ou professor.

4.3.11 Tela de Parabéns

A tela de parabéns do aplicativo (Figura 4.19) é a tela motivacional que aparecerá quando o aluno acertar uma atividade. Funciona como um reforço positivo e contém um vídeo parabenizando-o pelo acerto.

4.3.12 Atividade 1

Apresentação do objeto

A primeira tela da atividade (Figura 4.20) consiste na apresentação do objeto perigoso, com o intuito de familiarizar o aluno. Contém a imagem do objeto no centro, bem como seu nome no topo da tela, com todas as letras em maiúscula e a inicial vermelha, um padrão muito utilizado na alfabetização infantil.

Quando o aluno estiver pronto para prosseguir com a atividade, deve clicar em "AVANÇAR", na barra de navegação inferior.

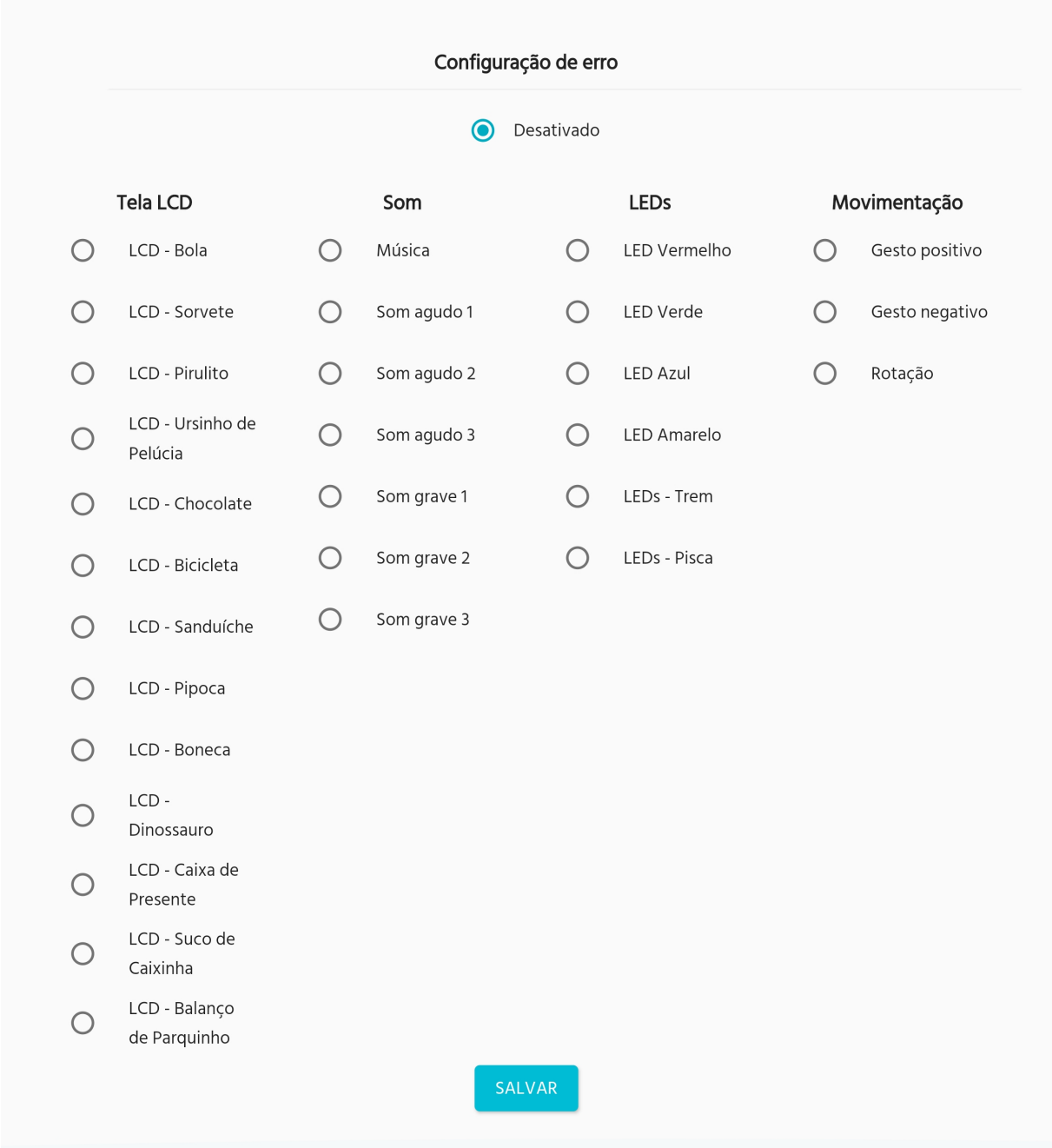


Figura 4.16: Modal de configurações do *feedback* do hardware complementar.

Ambientação na atividade

A segunda tela da atividade (Figura 4.21) consiste na ambientação do objeto como algo perigoso. Para as atividades padrão, há um vídeo na metade esquerda da tela que explica o que é o objeto e instrui o aluno a não tocar nele. Na metade direita da tela, fica uma imagem do objeto de perigo, que não deve ser tocada por 10 segundos após o término do vídeo explicativo. Para as atividades customizadas, não há vídeo explicativo, apenas um

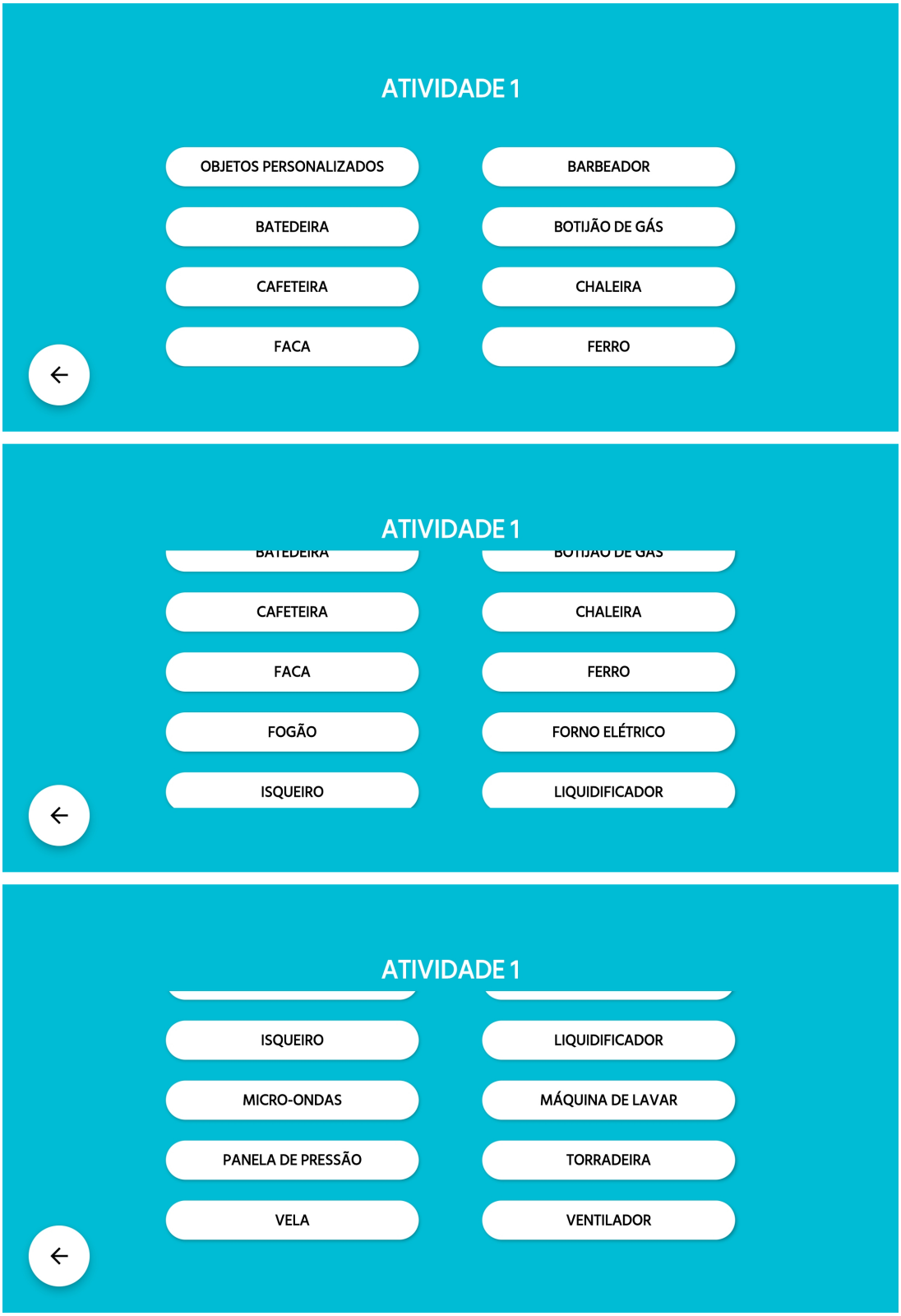


Figura 4.17: Menu de objetos para a Atividade 1.

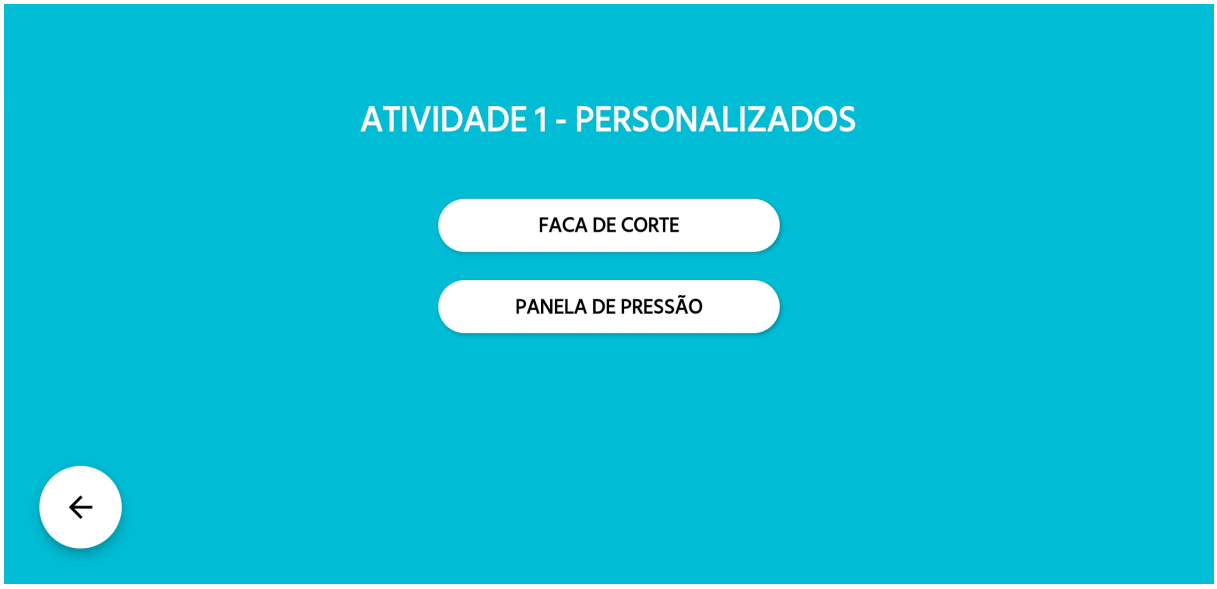


Figura 4.18: Menu da Atividade 1 para objetos personalizados.

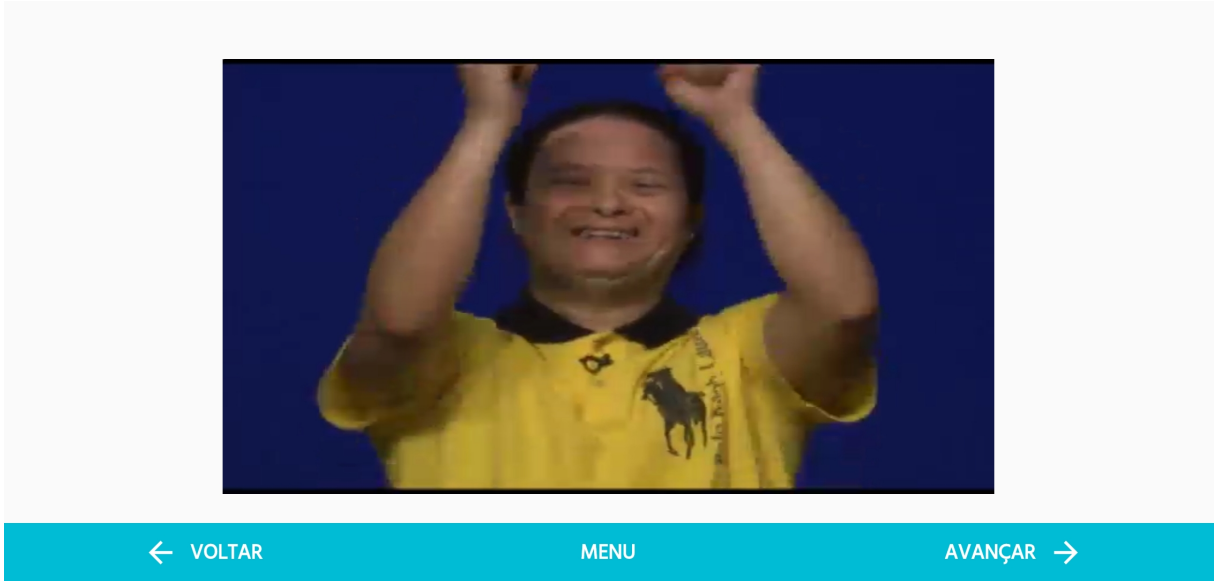


Figura 4.19: Tela de parabéns.

botão que deve ser tocado para dar início à atividade (Figura 4.22).

Se o objeto for tocado, outro vídeo toma o lugar do vídeo inicial (ou no caso das atividades personalizadas, toma o lugar do botão de início da atividade), instruindo o aluno de que esta não foi a resposta correta, e o tempo de 10 segundos é reiniciado. Se o aluno tocar mais de 3 vezes no objeto, o aplicativo redireciona para uma tela com um vídeo reforçando, mais uma vez, que o objeto não pode ser tocado, e vai para a próxima atividade. Se o objeto não for tocado, o aplicativo redireciona para a tela de "Parabéns"

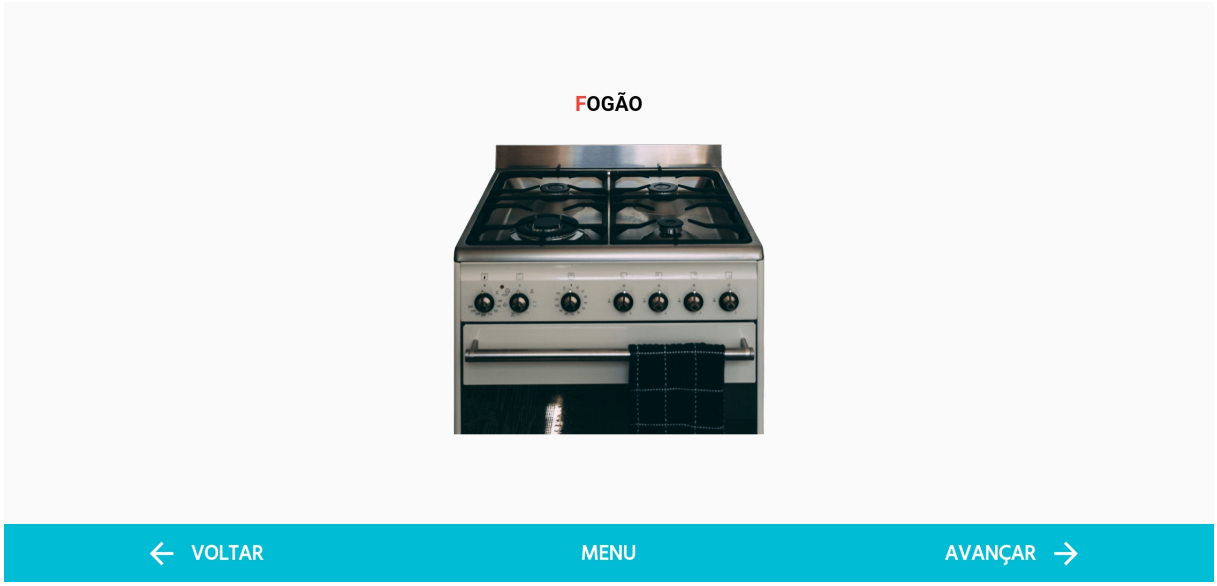


Figura 4.20: Tela de apresentação do objeto.

após 10 segundos, com um *feedback* positivo.

Além disso, se o hardware complementar tiver sido configurado para o aplicativo e estiver conectado ao dispositivo móvel, os *feedbacks* positivos e negativos também são enviados a ele.

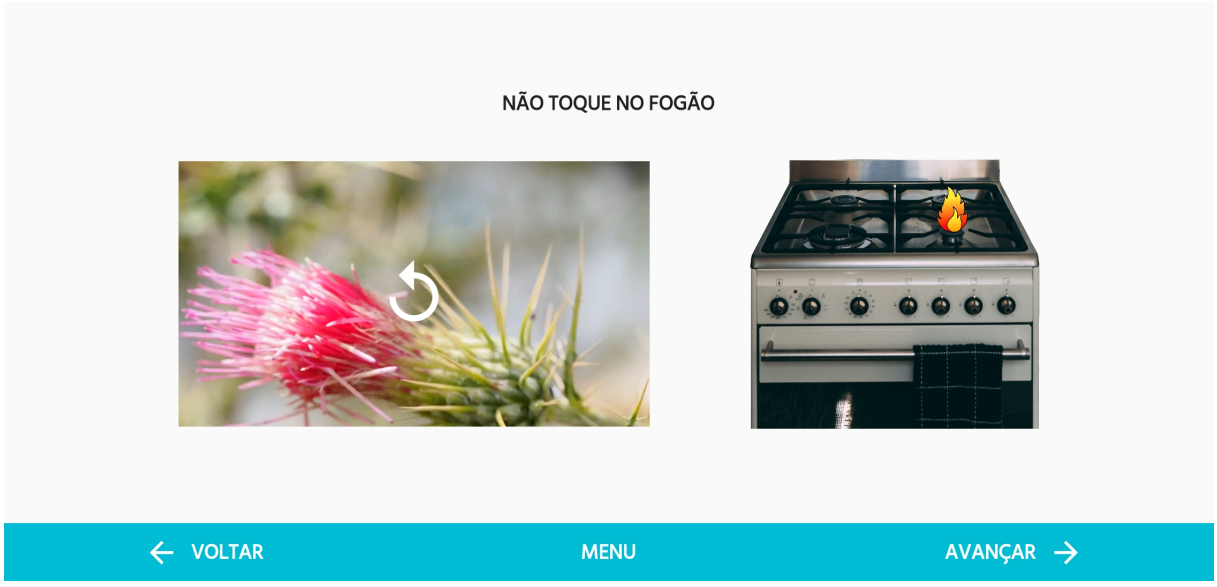


Figura 4.21: Tela de ambientação na atividade para objetos padrão.

Figura 4.22: Tela de ambientação na atividade para objetos personalizados.

Primeiro reforço

A terceira tela da atividade (Figura 4.23) pode ser definida como um primeiro reforço. No topo da tela, há um vídeo instrutivo da atividade, bem como a instrução escrita à direita. Além disso, é mostrada na tela a mesma imagem do objeto de perigo, com duas opções de resposta à pergunta da instrução.

Se o aluno clicar mais de uma vez na resposta errada, o aplicativo dá uma dica de qual é a resposta correta circulando-a com um círculo vermelho (Figura 4.24). Ao acertar a resposta, o aplicativo faz um som de campainha positivo e redireciona para a tela de "Parabéns".

Segundo reforço

Esta tela (Figura 4.25) é parecida com a anterior, contendo os mesmos elementos, com a adição da imagem de uma pessoa perto do objeto de perigo. A instrução pergunta se a pessoa está perto do objeto, e ao ser respondida corretamente, altera a posição dos elementos (objeto e pessoa) para ser respondida novamente (Figura 4.26). Após três posições, o aplicativo redireciona para a tela de "Parabéns" e vai para a próxima tela.

Da mesma forma que a tela anterior, ao errar uma resposta mais de uma vez, o aplicativo dá uma dica ao aluno circulando a resposta correta com um círculo vermelho.

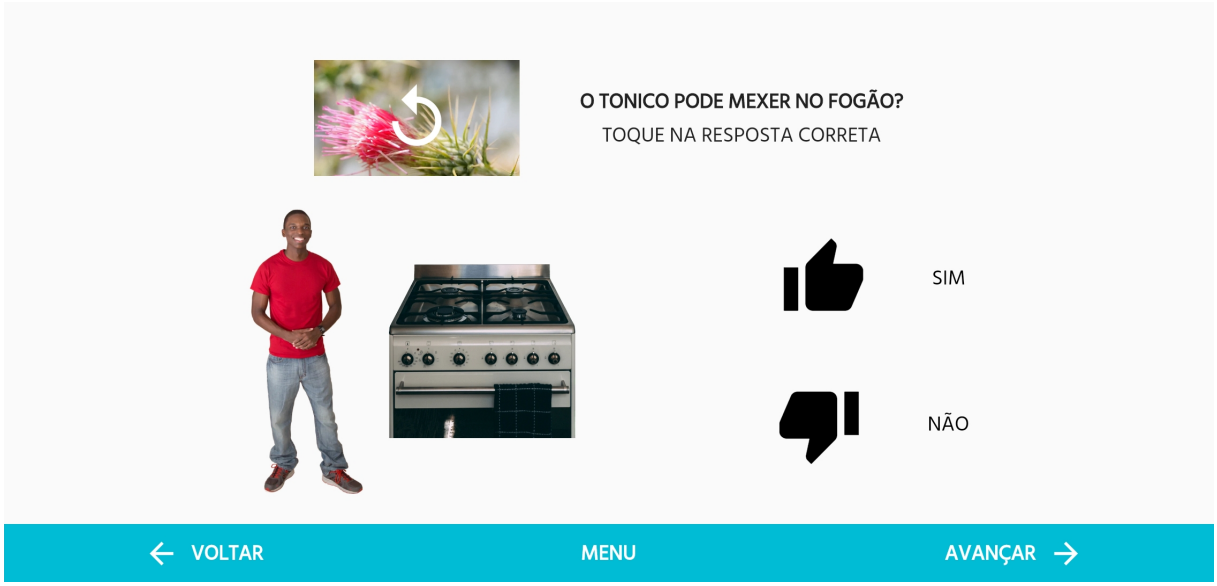


Figura 4.23: Primeiro reforço da Atividade 1.

Figura 4.24: Dica ao tocar mais de uma vez na resposta errada na Atividade 1.

Terceiro reforço

O último reforço possui um vídeo instrutivo, a instrução da atividade por escrito, a imagem de uma menina e do objeto de perigo. Inicialmente, o aluno deve tocar na mão da menina (Figura 4.27), e depois levá-la para longe do perigo, arrastando-a pela tela (Figura 4.28). Ao acertar, os elementos são colocados em outra posição e a atividade deve ser realizada novamente para a nova localização (Figura 4.29). Ao finalizar, a atividade redireciona,

Figura 4.25: Segundo reforço da Atividade 1.

Figura 4.26: Segundo reforço da Atividade 1 - posição diferente.

então, para a tela de "Parabéns" e a atividade 1 termina, voltando para o menu principal.

4.3.13 Atividade 2

Esta atividade tem por objetivo diferenciar objetos seguros e não-seguros (ou perigosos) e conta com quatro objetos a serem trabalhados. Com o intuito de não existirem elementos surpresa na tela, ou seja, elementos que aparecem sem serem introduzidos, todos os quatro

Figura 4.27: Terceiro reforço da Atividade 1 - tocar.

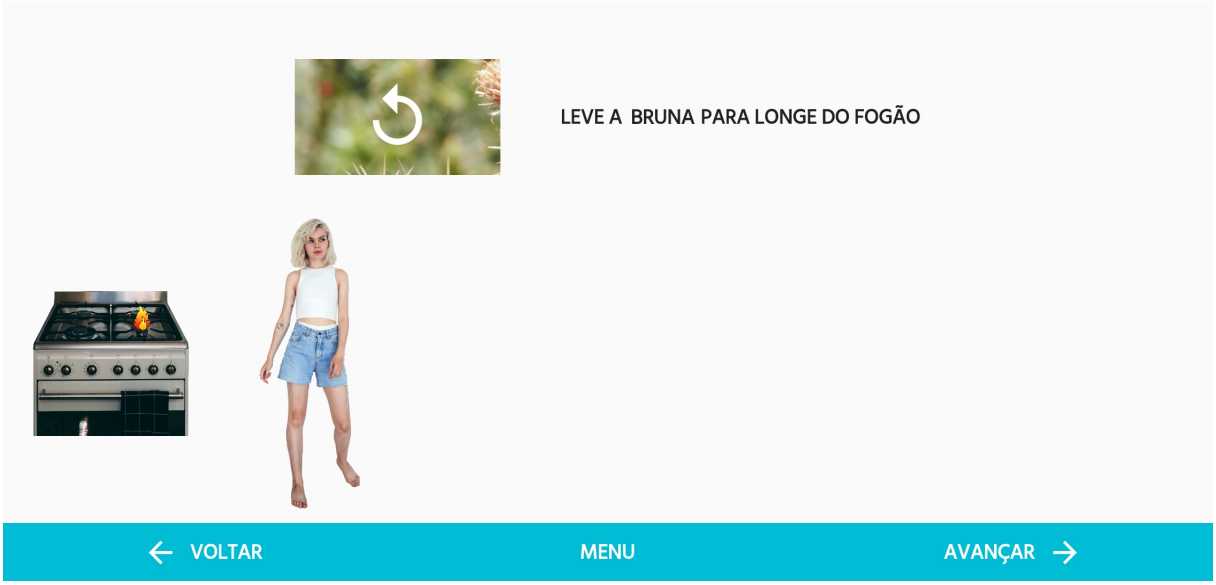


Figura 4.28: Terceiro reforço da Atividade 1 - arrastar.

objetos devem passar pela tela de apresentação do objeto antes de serem inseridos no contexto da atividade. Este requisito é bastante mencionado nas diretrizes de ensino especial para pessoas com TEA e também foi sugerido pelos professores especialistas que acompanharam o desenvolvimento do projeto.

Inicialmente, é mostrada uma tela de apresentação para o primeiro objeto, que é um objeto seguro escolhido aleatoriamente do banco de dados (Figura 4.30).

Ao avançar para a próxima tela (Figura 4.31), é mostrada uma instrução de “arraste os

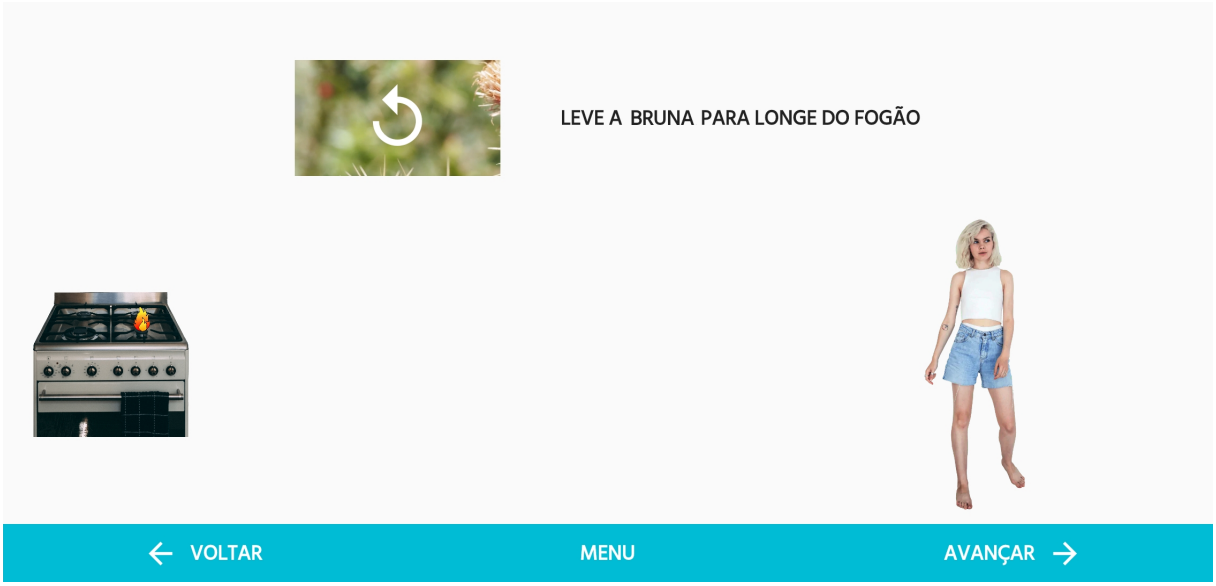


Figura 4.29: Terceiro reforço da Atividade 1 - posição diferente.

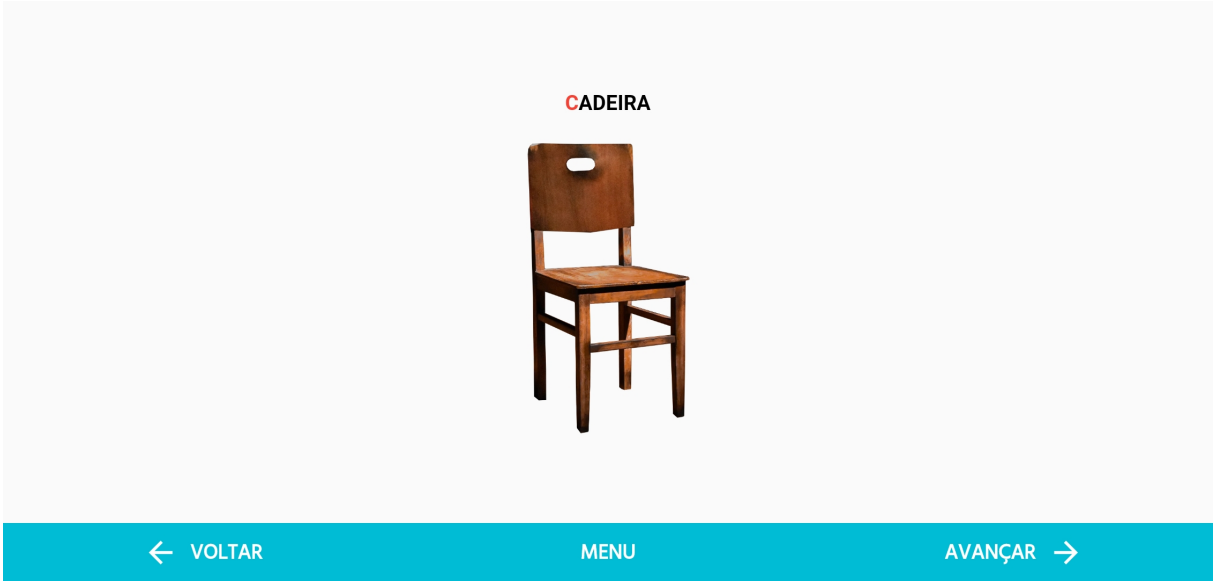


Figura 4.30: Tela de apresentação do objeto.

objetos seguros para o local correto". Do lado esquerdo, é mostrado o objeto apresentado na tela anterior; e do lado direito, uma área com título de "Objetos Seguros", que delimita a área para onde os objetos seguros podem ser arrastados.

Quando o aluno arrastar corretamente um objeto para a área (Figura 4.32), é tocada a campainha de *feedback* positivo, bem como enviado o *feedback* do hardware complementar, se existente. A atividade redireciona, então, para a apresentação do próximo objeto.

Após a apresentação do novo objeto, mostra-se novamente a tela de arrastar objetos,

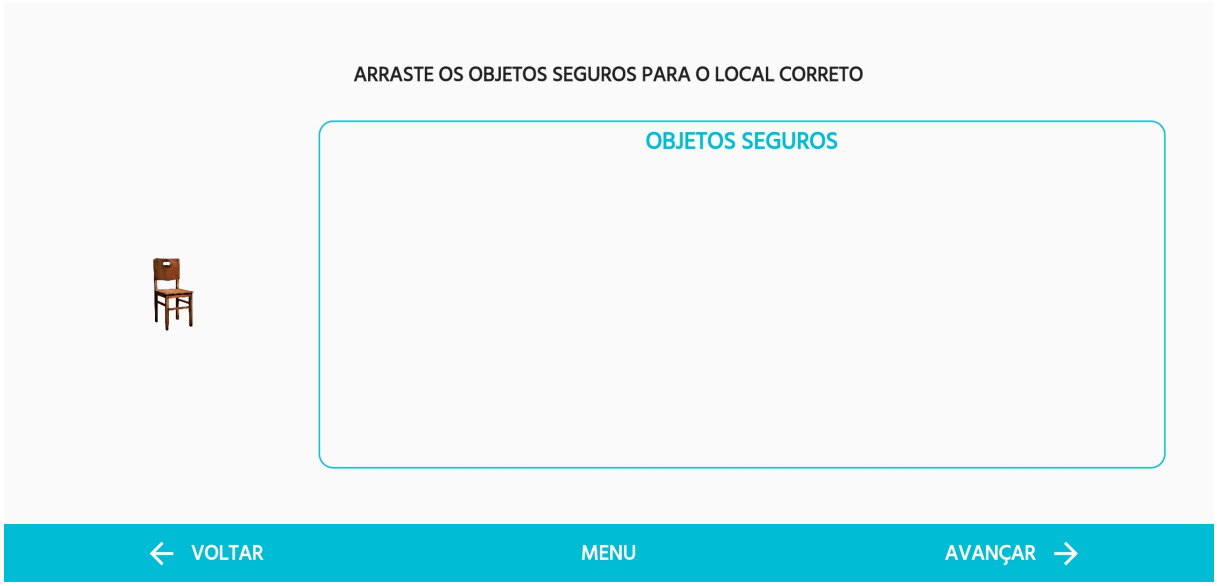


Figura 4.31: Tela de arrastar objetos - 1 objeto.

Figura 4.32: Objeto arrastado para a área de objetos seguros.

agora com os dois objetos previamente apresentados podendo ser arrastados. Após tentar arrastar um objeto perigoso duas vezes, o aplicativo circula os objetos seguros com uma borda vermelha para indicar qual é a resposta correta e o objeto perigoso se torna fixo na tela (Figura 4.33).

Quando todos os objetos seguros presentes na tela forem colocados na área correta, a atividade segue para a próxima apresentação. Este processo ocorre para os quatro objetos que serão trabalhados ao longo da atividade (Figura 4.34 e Figura 4.35). Na Figura 4.36,

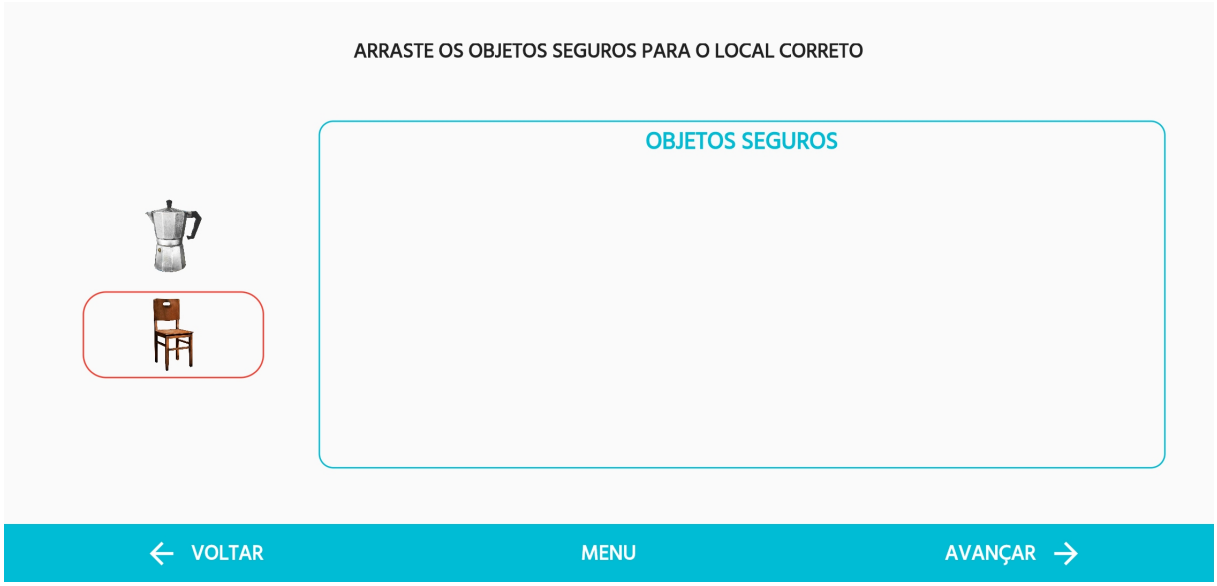


Figura 4.33: Dica de objeto seguro circulado para indicar a resposta correta.

é possível ver diferentes conjuntos de objetos que podem aparecer na atividade.

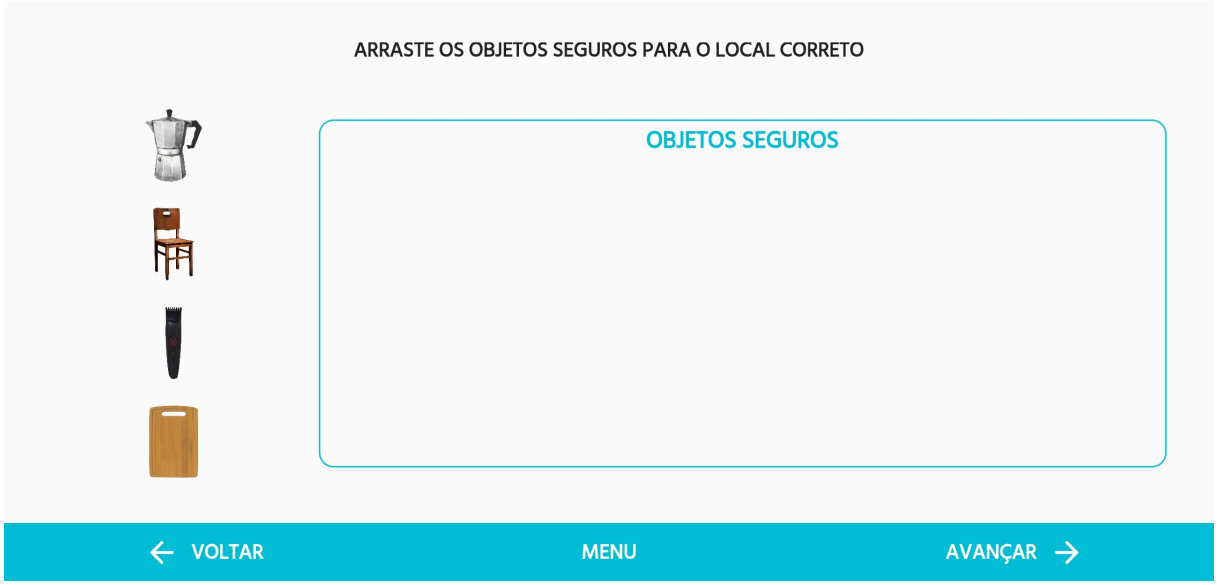


Figura 4.34: Tela de arrastar objetos - 4 objetos.

4.3.14 Atividade 3

A Atividade 3 tem objetivo similar ao da Atividade 2, de diferenciar objetos seguros e não-seguros. Inicialmente, o primeiro objeto (seguro) é apresentado, e ao avançar na atividade, a tela mostra a instrução “toque nos objetos que não oferecem perigo” (Figura 4.37).

Figura 4.35: Tela com objetos arrastados.

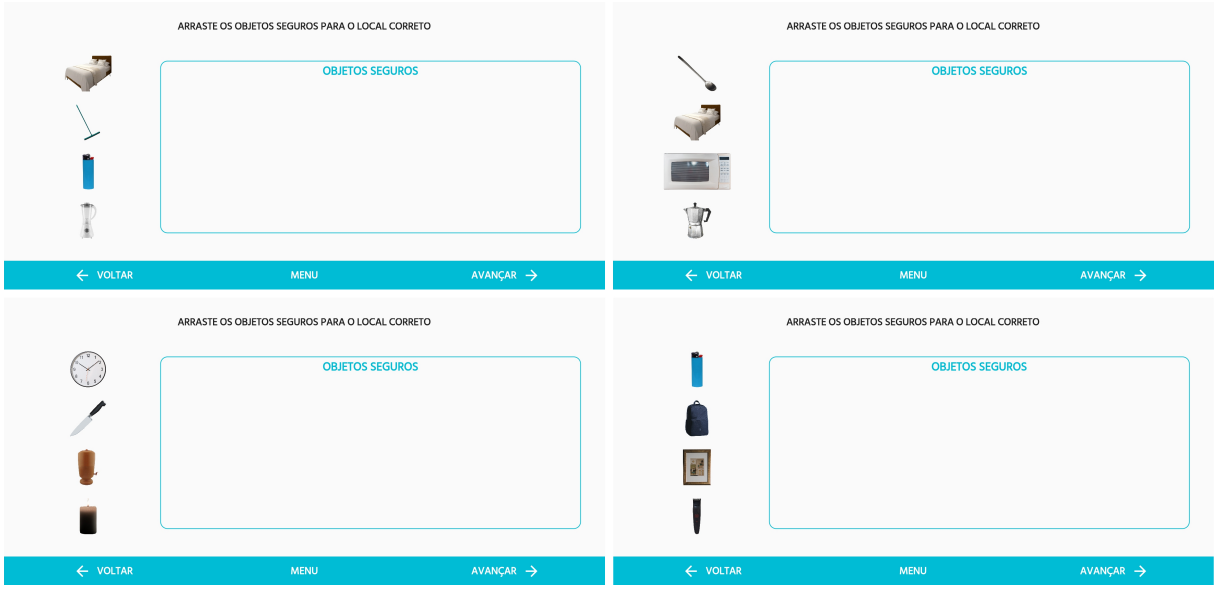


Figura 4.36: Diferentes conjuntos de objetos na Atividade 2.

Nesta tela, analogamente ao movimento de arrastar da Atividade 2, o aluno deve tocar nos objetos seguros mostrados.

Assim como na atividade anterior, cada objeto é apresentado e depois inserido no contexto. Sendo assim, este processo ocorre para cada objeto até mostrar todos os 4 objetos da atividade (Figura 4.38).

Um objeto seguro tocado fica marcado com um ícone de verificação verde (Figura 4.39) e os *feedbacks* positivos são acionados. Já após 2 tentativas de toque nos objetos perigosos,

Figura 4.37: Tela de tocar nos objetos seguros - 1 objeto.

Figura 4.38: Tela de tocar nos objetos seguros - 4 objetos.

os objetos seguros são destacados com uma borda vermelha para indicar a resposta correta (Figura 4.40). A atividade espera que todos os objetos seguros sejam tocados para ser finalizada.

Outros conjuntos de objetos para a Atividade 3 podem ser visualizados na Figura 4.41.

Figura 4.39: Objeto seguro tocado na Atividade 3.

Figura 4.40: Dica na Atividade 3 ao tocar em objetos perigosos.

4.4 Cubinho: Hardware Complementar

O Cubinho pode ser visualizado na Figura 4.42. Seu exterior é formado por um cubo de papel panamá, um papel rígido com espessura de 2.5mm, e pode ser customizado de acordo com a preferência da pessoa com TEA.

Seu objetivo é oferecer uma gama de reforçadores para o comportamento esperado, o reconhecimento de situações de perigo no ambiente doméstico, tendo em vista que

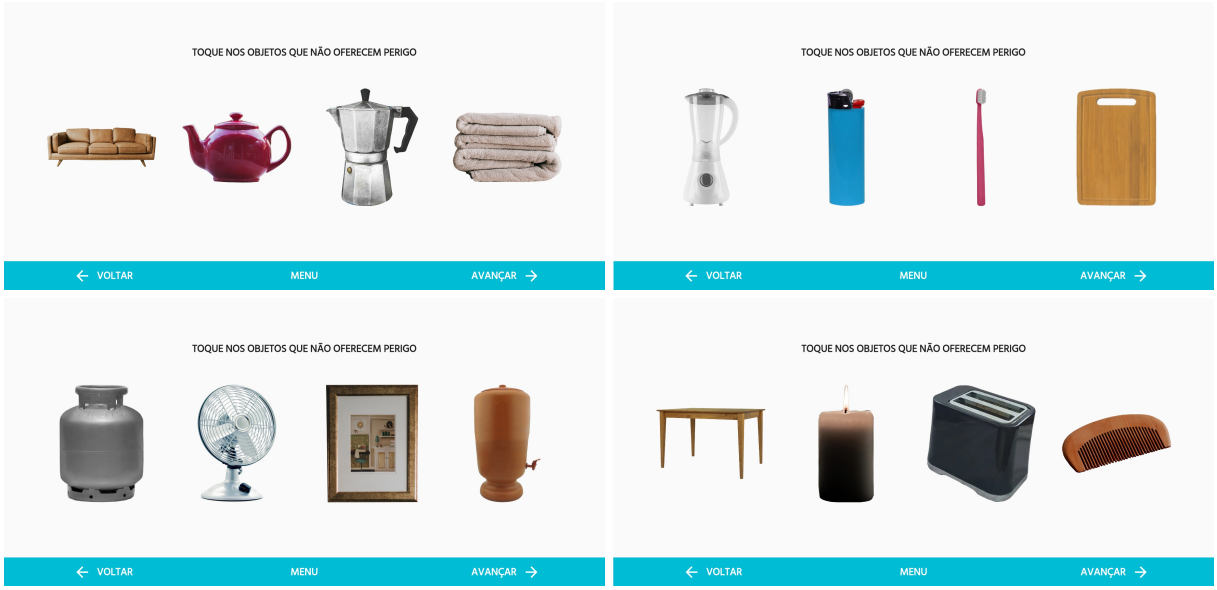


Figura 4.41: Diferentes conjuntos de objetos da Atividade 3.

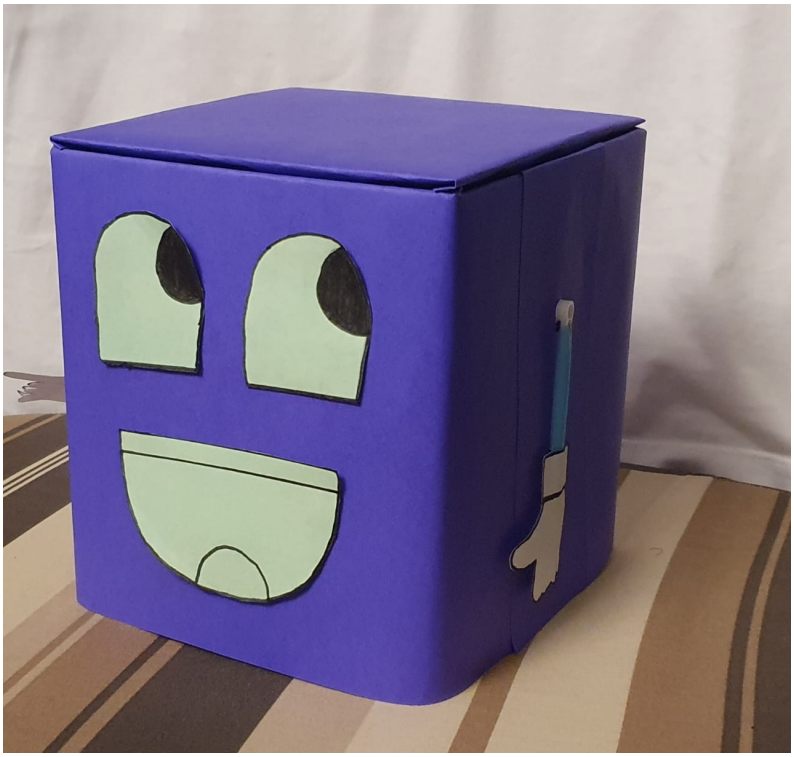


Figura 4.42: Cubinho - hardware motivacional complementar ao aplicativo Prevenir.

reforçadores para pessoas neurotípicas nem sempre funcionam para pessoas do espectro autista.

A placa microcontroladora do Cubinho está conectada a diversos componentes eletrônicos, como motores DC, LEDs, alto-falantes, motores servo e um display LCD. O Cubinho pode:

- Utilizar luzes de diferentes cores e em diferentes padrões;
- Utilizar sons com diferentes tons e melodias;
- Exibir imagens motivacionais;
- Se movimentar realizando gestos em resposta ao acerto ou erro da atividade.

O hardware se comunica com o software através de uma comunicação serial implementada no aplicativo com auxílio do pacote *flutter_bluetooth_serial*, um pacote para Flutter que facilita o uso de conexão bluetooth no app.

Capítulo 5

Conclusão

Os acidentes domésticos são responsáveis por boa parte das internações hospitalares e mortalidade no Brasil, podendo ser ainda mais recorrentes no caso de pessoas com Transtorno do Espectro Autista. As diferentes formas e ritmos de aprendizado de pessoas com TEA em comparação a pessoas neurotípicas resultam, muitas vezes, na dificuldade de reconhecimento de situações perigosas, tornando necessária a existência de tecnologias assistivas específicas que auxiliem no processo de aprendizado.

Neste contexto, foi desenvolvido o presente projeto, que resultou na criação de um aplicativo para dispositivos móveis chamado Prevenir, cujo objetivo é o de auxiliar na educação de pessoas com TEA no que diz respeito à percepção de situações de perigo e à precaução de possíveis acidentes domésticos. Espera-se que este software possa aumentar a autonomia de pessoas do espectro autista e sirva como uma ferramenta de suporte para os pais e responsáveis por pessoas com TEA.

A temática de prevenção de acidentes domésticos está em consonância com o currículo funcional do MEC e é inédita no Brasil no que diz respeito ao seu uso em aplicativos para o público-alvo definido para este projeto.

Por meio de diferentes atividades, o aplicativo busca apresentar objetos encontrados em casa que possam oferecer algum nível de perigo, como por exemplo: fogão, liquidificador ou barbeador; explicitar ao aluno as possíveis situações perigosas por eles causadas; e orientar acerca de como proceder perante tais cenários.

Este projeto, que envolve o uso de tecnologia em um contexto de acessibilidade no âmbito da Educação Especial de pessoas com Transtorno do Espectro Autista, apresentou algumas dificuldades em seu desenvolvimento. A mais expressiva delas se deu na construção do *design*, da estrutura e do conteúdo do software, que precisou abranger todo um espectro de pessoas com autismo, com uma gama de necessidades, singularidades e preferências que tiveram que ser elaboradas e implementadas no aplicativo.

Devido à pandemia do COVID-19, não foi possível realizar a validação dos conceitos teóricos utilizados no desenvolvimento do projeto no âmbito prático. Assim, a validação do uso do aplicativo com pessoas com Transtorno do Espectro Autista é a principal sugestão para trabalhos futuros.

A configuração do aplicativo para iOS é também uma sugestão relevante, dependendo apenas da disponibilidade de dispositivos da Apple para ser realizada. A adição de novas atividades, de novos objetos perigosos ou seguros e de novas opções de customização do visual do aplicativo também são sugestões válidas para aumentar a robustez e o alcance do projeto.

Referências

[1] Reis, João Anselmo Bandeira dos: *Cubinho: Artefato robótico motivacional para pessoas com transtorno do espectro autista utilizando o aplicativo prevenir*. TCC (graduação) – Engenharia de Computação, Universidade de Brasília, 2021. v, 31

[2] Android: *Platform architecture*. `https://developer.android.com/guide/platform`. ix, 17, 18

[3] Apple: *ios technology overview*, 2014. `http://docshare01.docshare.tips/files/25082/250827276.pdf`. ix, 19, 20

[4] CAMINHA, Vera Lúcia, Julliane Yoneda HUGUENIN, Lúcia Maria de Assis e Priscila Pires Alves: *Autismo: vivências e caminhos*. São Paulo: Blucher, 11, 2016. 1, 11

[5] Britto, Talita Cristina Pagani: *Gaia: uma proposta de guia de recomendações de acessibilidade web com foco em aspectos do autismo*. 2016. 1, 12

[6] Fonseca, Juliana Tavares dos Reis e Carolina Rizzotto Schirmer: *Tecnologia assistiva: aplicativos para dispositivos móveis, uma contribuição tecnológica para aprendizagem de crianças autistas*. Revista Educação e Cultura Contemporânea, 17(51):155–175, 2020. 1

[7] Mulher, da Família e dos Direitos Humanos Ministério da: *Cartilha de prevenção aos acidentes domésticos*. 2020. 2, 10

[8] Cunha, Eugênio: *Autismo e inclusão: psicopedagogia e práticas educativas na escola e na família*. Digitaliza Conteudo, 2020. 5

[9] Kanner, Leo *et al.*: *Autistic disturbances of affective contact*. Nervous child, 2(3):217–250, 1943. 5

[10] Asperger, Hans e Uta Trans Frith: *'autistic psychopathy' in childhood*. 1991. 5

[11] Realidade, Autismo e: *Quatro médicos que mudaram a visão do mundo sobre autismo*, 2019. `https://autismoerealidade.org.br/2019/11/27/quatro-medicos-que-mudaram-a-visao-do-mundo-sobre-autismo/`. 5

[12] Realidade, Autismo e: *O que é o autismo? marcos históricos*. `https://autismoerealidade.org.br/o-que-e-o-autismo/marcos-historicos/`. 6

[13] Donvan, John e Caren Zucker: *Outra sintonia: a história do autismo*. Editora Companhia das Letras, 2017. 6

[14] Association, American Psychiatric *et al.*: *DSM-5: Manual diagnóstico e estatístico de transtornos mentais*. Artmed Editora, 2014. 6

[15] SBP, Sociedade Brasileira De Pediatria: *Transtorno do espectro do autismo*. 2019. https://www.sbp.com.br/fileadmin/user_upload/Ped._Desenvolvimento_-_21775b-MO_-_Transtorno_do_Espectro_do_Autismo.pdf. 6, 7

[16] Realidade, Autismo e: *O que é o autismo?* https://autismoerealidade.org.br/o-que-e-o-autismo/. 6, 7

[17] Mello, Ana Maria S: *Autismo: guia prático*. 2007. https://www.autismo.org.br/site/images/Downloads/7guia%20pratico.pdf. 7, 8

[18] OPAS: *Transtornos do espectro autista*, 2017. https://www.paho.org/bra/index.php?Itemid=1098. 7

[19] Carothers, Douglas E. e Ronald L. Taylor: *Como pais e educadores podem trabalhar juntos para ensinar habilidades básicas de vida diária para crianças com autismo*, 2004. 8

[20] Santos, Juliane F. S.: *Análise do comportamento auxilia no tratamento de tea*. Instituto de Psicologia - USP, 2019. https://sites.usp.br/psicousp/analise-do-comportamento-auxilia-no-tratamento-de-tea/. 8

[21] PECS-Brasil: *Sistema de comunicação por troca de figuras*. https://pecs-brazil.com/sistema-de-comunicacao-por-troca-de-figuras-pecs/. 9

[22] Brasil, Presidência da República do: *Constituição da república federativa do brasil*. http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm. 9

[23] Brasil, Presidência da República do: *Estatuto da criança e do adolescente*. http://www.planalto.gov.br/ccivil_03/leis/L8069.htm. 9

[24] Brasil, Presidência da República do: *Estatuto do idoso*. http://www.planalto.gov.br/ccivil_03/leis/2003/l10.741.htm. 9

[25] Brasil, Presidência da República do: *Lei berenice piana, 12.764/12*. http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l12764.htm. 9

[26] Brasil, Presidência da República do: *Estatuto da pessoa com deficiência*. http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm. 9

[27] Brasil, Presidência da República do: *Convenção internacional sobre os direitos das pessoas com deficiência*. http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2009/decreto/d6949.htm. 9

[28] Senado, Agência: *Sancionada lei que inclui conceito de educação e aprendizagem ao longo da vida na ldb*, 2018. https://www12.senado.leg.br/noticias/materias/2018/03/07/sancionada-lei-que-inclui-conceito-de-educacao-e-aprendizagem-ao-longo-da-vida-na 9

[29] Brasil, Presidência da República do: *Lei 8.899/94*. http://www.planalto.gov.br/ccivil_03/leis/l8899.htm. 9

[30] Brasil, Presidência da República do: *Lei 8.742/93*. http://www.planalto.gov.br/ccivil_03/leis/l8742.htm. 9

[31] Brasil, Presidência da República do: *Decreto 7.611/11*. http://www.planalto.gov.br/ccivil_03/_Ato2011-2014/2011/Decreto/D7611.htm. 9

[32] Brasil, Presidência da República do: *Lei 7.853/89*. http://www.planalto.gov.br/ccivil_03/leis/l7853.htm. 9

[33] Brasil, Presidência da República do: *Lei 10.048/00*. http://www.planalto.gov.br/ccivil_03/leis/l10048.htm. 10

[34] MS, Ministério Da Saúde: *Política nacional de redução da morbimortalidade por acidentes e violências*. http://bvsms.saude.gov.br/bvs/publicacoes/politica_reducao_morbimortalidade_acidentes_2ed.pdf. 10

[35] Chiarelli, Amanda, Davi Mamblona Marques Romão, Fernando Camargo Filho, Isabela Salles, Jorge Otávio Maia Barreto, Laura dos Santos Boeira, Marcel Henrique de Carvalho, Tainá Raiol, Vahíd Shaikhzadeh Vahdat, Walter Ramalho *et al.*: *Prevenção de acidentes domésticos no distrito federal*. 2019. 10

[36] RNPI, Rede Nacional Primeira Infância: *Plano nacional pela primeira infância*. 2020. http://primeirainfancia.org.br/wp-content/uploads/2020/10/PNPI.pdf. 10

[37] Sociedade Brasileira De Pediatria, SBP: *Os acidentes são evitáveis e na maioria das vezes, o perigo está dentro de casa!* 2020. https://www.sbp.com.br/fileadmin/user_upload/_22337c-ManOrient_-_Os_Acidentes_Sao_Evitaveis__1_.pdf. 10

[38] SEDH, Secretaria Especial Dos Direitos Humanos: *Ata vii reunião do comitê de ajudas técnicas – cat corde / sedh / pr*, 2007. https://www.assistiva.com.br/Ata_VII_Reuni%C3%A3o_do_Comite_de_Ajudas_T%C3%A9cnicas.pdf. 11

[39] SNPDPCD, Subsecretaria Nacional De Promoção Dos Direitos Da Pessoa Com Deficiência: *Tecnologia assistiva*. Brasília: Corde, 2009. 11

[40] Fernandes, Flávia Gonçalves, Luciene Chagas de OLIVEIRA, Mylene Lemos Rodrigues e Stéfano Schwenck Borges Vale Vita: *Sistema para auxílio na alfabetização de crianças com autismo utilizando realidade aumentada para dispositivos móveis*. 2014. http://www.ceel.eletrica.ufu.br/artigos2014/ceel2014_artigo007_r01.pdf. 11

[41] W3C, World Wide Web Consortium: *Cartilha de acessibilidade na Web: fascículo 3-Conhecendo o público-alvo da acessibilidade na web*. CGI. br, 2018. 11

[42] Magaton, Heloise Cristini e Silvia Amélia Bim: *Recomendações para o desenvolvimento de softwares voltados para crianças com transtorno do espectro autista*. Revista Brasileira de Informática na Educação, 27(02):112, 2019. 14

[43] ONU, Organização Das Nações Unidas: *Report of the special rapporteur on the promotion and protection of the right to freedom of opinion and expression, frank la rue**, 2011. https://www2.ohchr.org/english/bodies/hrcouncil/docs/17session/A.HRC.17.27_en.pdf. 15

[44] IBGE: *Uso de internet, televisÃo e celular no brasil*, 2020. https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html. 15

[45] VALENTE, Jonas: *Brasil é o 3º país em que pessoas passam mais tempo em aplicativos*, 2020. https://agenciabrasil.ebc.com.br/geral/noticia/2020-01/brasil-e-o-3o-pais-em-que-pessoas-passam-mais-tempo-em-aplicativos. 15

[46] Department, Statista Research: *Number of apps available in leading app stores as of 4th quarter 2020*, 2021. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. 16

[47] Nosrati, Masoud, Ronak Karimi e Hojat Allah Hasanvand: *Mobile computing: principles, devices and operating systems*. World Applied Programming, 2(7):399–408, 2012. 16

[48] Statista: *Mobile operating systems' market share worldwide from january 2012 to january 2021*. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/. 16

[49] Tanenbaum, Andrew S e Herbert Bos: *Modern operating systems*. Pearson, 2015. 17, 18

[50] AppBrain: *Number of android apps on google play*, 2021. https://www.appbrain.com/stats/number-of-android-apps. 19

[51] Chebbi, Ajay: *Choosing the best programming language for mobile app development*, 2019. https://developer.ibm.com/technologies/mobile/articles/choosing-the-best-programming-language-for-mobile-app-development/. 19

[52] Shafirov, Maxim: *Kotlin on android. now official*, 2017. https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/. 19

[53] Lardinois, Frederic: *Kotlin is now google's preferred language for android app development*, 2019. https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/. 19

[54] James Nunns, TechMonitor: *What is ios?*, 2017. https://techmonitor.ai/what-is/what-is-ios-4899783. 19

[55] Gartner: *Gartner says worldwide sales of smartphones returned to growth in first quarter of 2018*, 2018. https://www.gartner.com/en/newsroom/press-releases/2018-05-29-gartner-says-worldwide-sales-of-smartphones-returned-to-growth-in-firs 20

[56] Cervantes, Edgar: *8 things ios does better than android*, 2021. https://www.androidauthority.com/ios-vs-android-1068950/. 20

[57] John, Neil: *Ipa file – an in-depth view*, 2016. https://www.hexnode.com/blogs/ipa-file. 20

[58] AppleInsider: *Apple's app store launches with more than 500 apps*, 2008. https://appleinsider.com/articles/08/07/10/apples_app_store_launches_with_more_than_500_apps. 20

[59] Statista: *Number of available apps in the apple app store from july 2008 to january 2017*, 2017. https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/. 20

[60] Iqbal, Mansoor: *App download and usage statistics*, 2021. https://www.businessofapps.com/data/app-statistics. 20

[61] DevMountain: *What languages are ios apps written in?* https://blog.devmountain.com/what-languages-are-ios-apps-written-in/. 21

[62] Apple: *Swift: The powerful programming language that is also easy to learn.* https://developer.apple.com/swift/. 21

[63] StackOverflow: *Most loved, dreaded, and wanted languages.* https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted. 21

[64] MANCHANDA, Amit: *Where do cross-platform app frameworks stand in 2021?*, 2020. https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/. 21, 22, 23

[65] CHEBBI, Ajay: *Choosing the best programming language for mobile app development*, 2019. https://developer.ibm.com/technologies/mobile/articles/choosing-the-best-programming-language-for-mobile-app-development/. 22

[66] Altexsoft: *The good and the bad of xamarin mobile development*, 2020. https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/. 23

[67] Native, React: *Who's using react native?*, 2021. https://reactnative.dev/showcase. 23

[68] AHMED, ASAD JIBRAN: *Hybrid power: Flutter advantages and benefits.* https://www.toptal.com/flutter/hybrid-power-flutter-advantages. 23, 24

[69] Bak, Lars: *Dart: A language for structured web programming*, 2011. https://blog.chromium.org/2011/10/dart-language-for-structured.html. 23

[70] Dart: *Dart overview*. https://dart.dev/overview. 24

[71] Flutter: *Flutter - beautiful native apps in record time*. https://flutter.dev/. 24

[72] Flutter: *Apps take flight with flutter*. https://flutter.dev/showcase. 24

[73] Flutter: *Introduction to widgets*. https://flutter.dev/docs/development/ui/widgets-intro. 24

[74] Android: *Meet android studio*, 2021. https://developer.android.com/studio/intro. 26

[75] Date, Christopher J: *Introdução a sistemas de bancos de dados*. Elsevier Brasil, 2004. 26, 27

[76] Jatana, Nishtha, Sahil Puri, Mehak Ahuja, Ishita Kathuria e Dishant Gosain: *A survey and comparison of relational and non-relational database*. International Journal of Engineering Research & Technology, 1(6):1–5, 2012. 27

[77] Kreibich, Jay: *Using SQLite*. " O'Reilly Media, Inc.", 2010. 27

[78] Feiler, Jesse: *Introducing SQLite for Mobile Developers*. Apress, 2015. 27

[79] Tekartik: *sqflite*, 2021. https://pub.dev/packages/sqflite. 27

[80] Bloc, Flutter: *Flutter bloc: Why bloc*. https://bloclibrary.dev/#/whybloc. 28

[81] Bloc, Flutter: *Flutter bloc: Architecture*. https://bloclibrary.dev/#/architecture. 28

[82] Bloc, Flutter: *Flutter bloc: Core concepts*. https://bloclibrary.dev/#/flutterbloccoreconcepts. 29

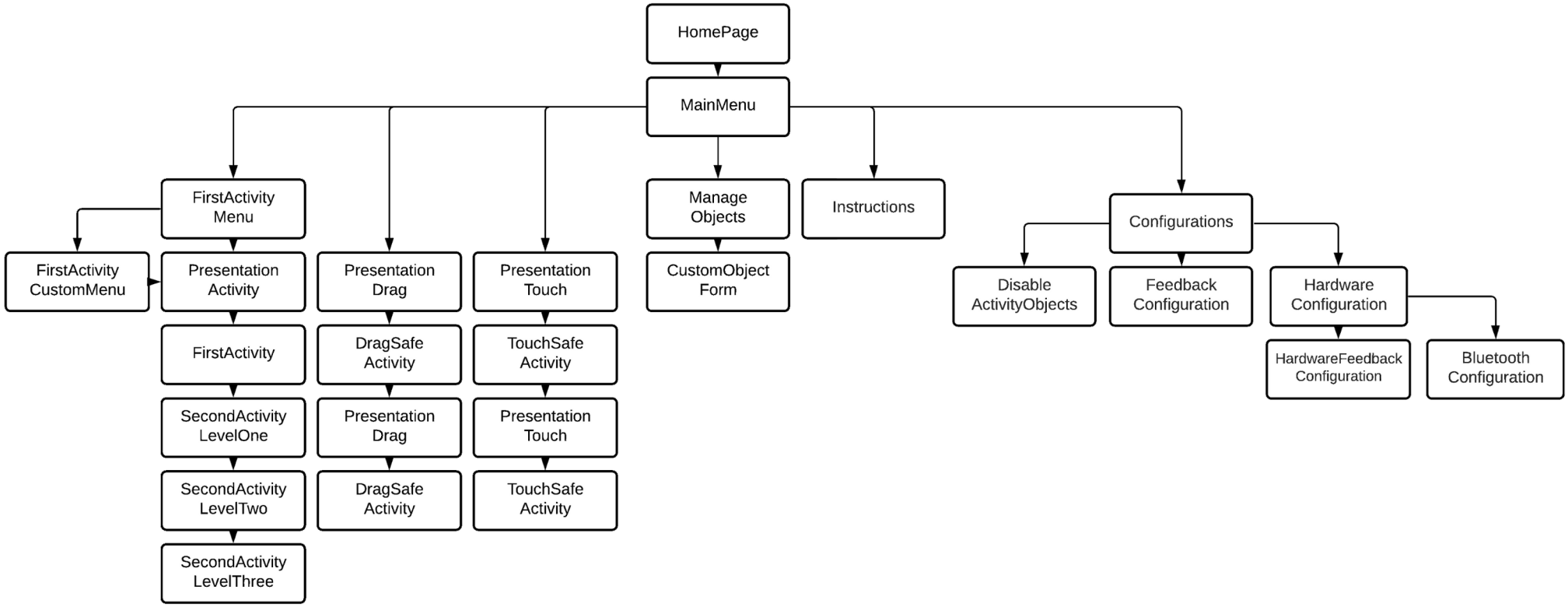
[83] Koscianski, André e Michel dos Santos Soares: *Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. Novatec Editora, 2007. 30

[84] Santos Soares, Michel dos: *Comparação entre metodologias ágeis e tradicionais para o desenvolvimento de software*. INFOCOMP Journal of Computer Science, 3(2):8–13, 2004. 30

[85] Espinha, Roberto Gil: *O que é kanban: guia completo*, 2021. https://artia.com/kanban/. 30

Anexo I

Estrutura de Classes do Aplicativo


HomePage
MainMenu
FirstActivity Menu
FirstActivity CustomMenu
Presentation Activity
FirstActivity
SecondActivity LevelOne
SecondActivity LevelTwo
SecondActivity LevelThree
Presentation Drag
DragSafe Activity
Presentation Drag
DragSafe Activity
Presentation Touch
TouchSafe Activity
Presentation Touch
TouchSafe Activity
Manage Objects
CustomObject Form
Instructions
Configurations
Disable ActivityObjects
Feedback Configuration
Hardware Configuration
HardwareFeedback Configuration
Bluetooth Configuration