Componentes de UI acessíveis: aprimoramento de um pacote inclusivo em SwiftUI

Alexandra Viana Zarzar[1], **Kiev Gama**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE – Brasil

avz@cin.ufpe.br, kiev@cin.ufpe.br

Abstract. *Mobile devices have become an essential part of the daily lives of countless individuals. It is essential to make user interfaces accessible to a diverse audience. However, it is not common for developers to include assistive support for people with visual impairments in their projects. Unfortunately, these professionals face several challenges when implementing inclusive technologies. There are few tools that help them in this process and those that do exist are often complex, requiring extensive prior knowledge. In this sense, this project proposes the improvement of a package of accessible UI components, with the aim of facilitating the experience of developers by adding accessibility to their codes.*

Resumo. *Dispositivos móveis tornaram-se uma parte essencial no cotidiano de inúmeros indivíduos. É imprescindível tornar as interfaces de usuário acessíveis a uma audiência diversificada. No entanto, não é recorrente que os desenvolvedores incluam suporte assistivo para pessoas com deficiência visual em seus projetos. Infelizmente, esses profissionais enfrentam diversos desafios para implementar tecnologias inclusivas. Há poucas ferramentas que os auxiliem nesse processo e as existentes são, frequentemente, complexas, exigindo um vasto conhecimento prévio. Nesse sentido, o presente projeto propõe o aperfeiçoamento de um pacote de componentes de UI acessíveis, com o objetivo de facilitar a experiência dos desenvolvedores ao adicionar acessibilidade em seus códigos.*

1. Introdução

A expansão contínua da presença dos dispositivos móveis tem redefinido substancialmente a forma como a população mundial conduz suas atividades diárias. Operações financeiras, planejamento de viagens, monitoramento de saúde, entretenimento e educação são apenas alguns exemplos de motivos para utilização de aplicativos móveis. No entanto, esse avanço tecnológico, apesar de revolucionário, evidenciou uma lacuna preocupante: a ausência de acessibilidade para usuários que apresentam alguma deficiência visual.

É fundamental que *softwares* móveis sejam desenvolvidos levando em consideração a inclusão de tecnologias assistivas para pessoas com visão limitada. A Organização das Nações Unidas defende a construção de uma sociedade igualitária, em que todos tenham acesso aos mesmos direitos e oportunidades [Organização das Nações Unidas 2006]. O uso pleno de tecnologias digitais é não só um direito dos indivíduos com dificuldades visuais, mas também um possível caminho

para melhor qualidade de vida, já que esses dispositivos podem facilitar atividades como leitura, pesquisa e localização [Khan and Khusro 2021] [Kim et al. 2016]. Tal discussão, envolvendo deficiência, acessibilidade e tecnologia, será aprofundada nas seções 2 e 3.

Conforme será discutido na seção 4, diversas ferramentas e diretrizes foram concebidas nos últimos anos com o objetivo de tornar aplicações móveis mais inclusivas. Porém, sua escassez, baixa adoção e complexidade têm limitado significativamente seu impacto [de Oliveira and Filgueiras 2018]. De acordo com [Di Gregorio et al. 2022], a maioria dos desenvolvedores não prioriza a acessibilidade em seus projetos, nem apresenta conhecimento amplo sobre o assunto. Assim, seria necessário bastante tempo e recursos para adaptar esses *softwares* [Abuaddous et al. 2016]. Dessa forma, infelizmente, muitos desenvolvedores e empresas negligenciam o uso das recomendações disponíveis para adaptação de seus aplicativos para um público diverso [de Almeida and Gama 2021].

Por isso, é essencial que que sejam desenvolvidas ferramentas que impulsionem a adoção de práticas inclusivas no desenvolvimento de aplicativos. [Sanchez-Gordon and Luján-Mora 2013] destacam as opções limitadas de recursos como um dos principais motivos para a ausência da aplicação de diretrizes inclusivas nos projetos digitais. [Huq et al. 2023] citam a dificuldade de encontrar bibliotecas de interface do usuário (UI) projetadas para abranger o público com deficiência.

Nesse contexto, [Silva 2023] formulou um conjunto de componentes acessíveis de interface do usuário. O *package* nomeado Code4All se propõe a auxiliar programadores de aplicativos a criarem telas acessíveis para pessoas com deficiência visual, de maneira simples e eficiente, sem que seja necessário uma bagagem de conhecimento prévio sobre tecnologias assistivas. Dado o êxito e a relevância de Code4All, o presente trabalho propõe contribuir e aprimorar essa ferramenta, visando impactar um número cada vez maior de desenvolvedores e usuários.

A seção 5 fornecerá um olhar aprofundado sobre como Code4All foi desenvolvido, destacando seus pontos fortes, bem como possíveis pontos de aperfeiçoamento. Em seguida, as propostas práticas de melhoria serão introduzidas na seção 6, juntamente aos novos projetos de componentes. Já na seção 7, será apresentado o método de avaliação do pacote desenvolvido, cujos resultados serão discutidos na seção 8. Por fim, na seção 9, será realizada a discussão final sobre o projeto.

2. Deficiência e acessibilidade

De acordo com a [Organização Pan-Americana da Saúde], deficiência é um termo que abrange uma variedade de impedimentos físicos, mentais, intelectuais ou sensoriais que podem dificultar a participação plena de indivíduos na sociedade. Ao longo do tempo, passou-se a entender a deficiência de uma maneira mais abrangente, não apenas olhando para as questões fisiológicas. Hoje, a [Organização Pan-Americana da Saúde] reconhece uma abordagem mais ampla, que considera a complexa interação entre o estado de saúde de uma pessoa e os vários fatores que influenciam seu ambiente, sejam eles sociais, políticos ou físicos.

A [World Health Organization] avaliou que cerca de 1.6 bilhões de pessoas vivem com uma ou mais deficiências, e é fundamental promover a inclusão desses indivíduos, só assim as metas globais de saúde e desenvolvimento sustentáveis serão alcançadas. A ONU destacou a importância do cumprimento dos direitos humanos das

pessoas com deficiência através da Convenção sobre os Direitos das Pessoas com Deficiência, que defende a participação plena deste grupo de indivíduos na sociedade [Organização das Nações Unidas 2006].

Já o termo acessibilidade, conforme definido pela [Secretaria de Defesa Social de Pernambuco], refere-se à inclusão da pessoa com deficiência em diversas atividades, como o uso de produtos, serviços e informações. O objetivo é construir uma sociedade mais inclusiva e justa, onde todos tenham igualdade de participação e acesso aos mesmos direitos e oportunidades.

De acordo [Barbosa et al. 2021], a acessibilidade está relacionada à capacidade do usuário de utilizar e interagir com um sistema, sem que a interface imponha obstáculos à sua utilização. Nesse contexto, a garantia da acessibilidade é essencial para promover a igualdade de oportunidade e participação de todos os indivíduos na sociedade.

Como apresentado nos parágrafos anteriores, avanços significativos foram realizados na discussão sobre acessibilidade. Porém, conforme será discutido na seção 4, apesar do progresso conquistado, ainda há muito trabalho a ser feito para atender às necessidades das pessoas com deficiência. Compreender a diversidade de experiências e desenvolver soluções para os desafios enfrentados por essa parcela da população é fundamental para promover uma realidade mais justa para a sociedade.

3. Deficiência visual e tecnologia

Adentrando o espectro das deficiências visuais, encontramos uma ampla gama de condições, abrangendo desde a cegueira até a visão subnormal, que se refere à capacidade visual reduzida, seja por rebaixamento significativo da acuidade visual, redução do campo visual, sensibilidade aos contrastes ou limitação de outras capacidades [Brasil 2000]. Globalmente, pelo menos 2,2 bilhões de pessoas lidam com algum tipo de dificuldade visual [World Health Organization 2023].

Enquanto a visão é um sistema-guia poderoso para a maioria das pessoas, os cegos e aqueles com visão subnormal dependem de outros sistemas, como pistas olfativas, auditivas e táteis, para se orientar no ambiente ao seu redor. Assim, quando precisam fazer uso de ferramentas que não são adaptadas para suas necessidades, podem enfrentar certas dificuldades que os limitam de realizar determinada tarefa.

Nesse contexto, dentre os desafios enfrentados por pessoas com deficiência visual, podemos destacar o uso de tecnologias digitais, que, muitas vezes, não são desenvolvidas levando em consideração a acessibilidade para todos os públicos. Essa situação é alarmante, uma vez que, cada vez mais, dispositivos como *smartphones* se tornam uma parte maior da vida cotidiana na sociedade. Como consequência da exclusão de práticas inclusivas no desenvolvimento de um aplicativo, a experiência de muitos usuários pode ser comprometida e até mesmo impossibilitada.

Assim, é essencial que as aplicações para dispositivos móveis sejam planejadas de forma que indivíduos com visão limitada possam interagir com a interface e entender o conteúdo em tela. Além disso, se desenvolvidas com esse objetivo em mente, as aplicações *mobile* podem se tornar verdadeiras aliadas na maneira como essa parcela de indivíduos realiza atividades como ler, localizar-se e efetivar compras, melhorando significativamente sua qualidade de vida [Khan and Khusro 2021].

A Figura 1 é uma representação traduzida do conteúdo apresentado por [Kim et al. 2016]. Ela explora a relação entre pessoas com deficiência visual e o uso de tecnologia assistiva para ajudá-las a completar tarefas diárias. Analisar essa imagem permite um melhor entendimento sobre o papel importante que dispositivos acessíveis podem representar na vida de alguns indivíduos.

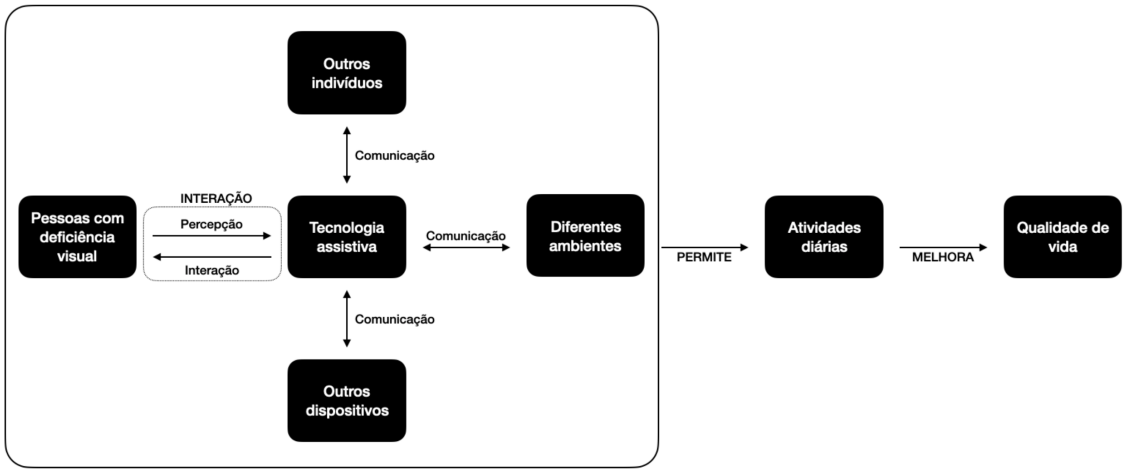


Figura 1. Modelo de funcionamento da tecnologia assistiva para pessoas com deficiência visual segundo [Kim et al. 2016]

4. Aplicação de práticas inclusivas no meio digital

4.1. Padrões e diretrizes de acessibilidade

Em resposta à necessidade de garantir que as tecnologias da informação e comunicação fossem utilizadas por uma ampla variedade de pessoas, surgiram as diretrizes de acessibilidade no meio digital. Essas diretrizes são conjuntos de padrões, recomendações e boas práticas para o *design*, desenvolvimento e manutenção de conteúdo digital, visando tornar esses recursos acessíveis a todos os usuários, independentemente de suas limitações.

A importância dessas iniciativas reside em promover a inclusão digital, permitindo que todos tenham igual acesso à informação, serviços e interações online. Dentre as diretrizes de acessibilidade existentes, o presente trabalho dá ênfase às seguintes:

- **Web Content Accessibility Guidelines (WCAG)**: uma das diretrizes mais reconhecidas e amplamente adotadas para garantir a acessibilidade de conteúdo na *web*, desenvolvida pelo World Wide Web Consortium (W3C). Fornece uma estrutura detalhada de critérios, organizados em quatro princípios fundamentais: Perceptível, Operável, Compreensível e Robusto. O WCAG define critérios específicos que os desenvolvedores devem seguir para garantir que seus projetos atendam às necessidades de diferentes usuários [W3C 2018].
- **Mobile Web Best Practices (MWBP)**: diretrizes específicas para garantir a acessibilidade e usabilidade *web* em dispositivos móveis [W3C 2008]. Essas diretrizes abordam questões como tamanho de tela, resolução, interações por toque, navegação simplificada e *design* responsivo. Assim, MWBP visa garantir que o conteúdo seja acessível e funcional em uma variedade de dispositivos móveis. Elas também

foram desenvolvidas pela W3C e têm muitos objetivos e similaridades em relação à WCAG, mas com abordagens diferentes [W3C 2009].

- **Human Interface Guidelines (HIG)**: conjunto de diretrizes projetadas pela Apple para garantir a consistência, usabilidade e acessibilidade das interfaces do usuário em seus sistemas operacionais e dispositivos [Apple Inc d]. Abrange aspectos específicos de inclusão, como *design* visual, navegação, interação por toque e *feedback* ao usuário. Dessa forma, contribui para uma experiência consistente e inclusiva em aplicativos e dispositivos da Apple [Apple Inc b].
- **Accessibility Developer Checklist**: documento desenvolvido especialmente para dispositivos Android. Fornece uma lista de verificação de requisitos, recomendações e considerações para ajudar desenvolvedores a garantir a inclusão de práticas assistivas em suas aplicações [Android Developers].
- **Modelo de Acessibilidade em Governo Eletrônico (eMAG)**: iniciativa do governo brasileiro que estabelece padrões e diretrizes para tornar os serviços públicos *online* acessíveis a todos os cidadãos. O eMAG define requisitos técnicos, sugestões de *design* e práticas recomendadas para o desenvolvimento de sites e aplicativos, visando garantir o cumprimento das leis de acessibilidade digital no Brasil. Tal iniciativa foi desenvolvida com base nas boas práticas descritas pelo WCAG, mas adaptando-o para o contexto de governo brasileiro [Departamento de Governo Eletrônico 2014].
- **Section 508**: este conjunto de padrões de acessibilidade é especificamente aplicável a tecnologias de informação e comunicação (TIC) nos Estados Unidos. Visa garantir que produtos e serviços do governo federal sejam acessíveis a pessoas com deficiência [U.S. General Services Administration 2017].
- **Accessibility Rich Internet Application (WAI- ARIA)**: conjunto de especificações técnicas projetadas para aprimorar a acessibilidade de conteúdos *online*. Embora aplicável a qualquer linguagem de marcação, o ARIA é particularmente adequado para o HTML. Esses atributos têm o propósito de facilitar a compreensão e interação de tecnologias assistivas, como leitores de tela, com o conteúdo dinâmico e interativo da *web* [W3C Web Accessibility Initiative 2006].

Essas diretrizes são essenciais para orientar o desenvolvimento de produtos e serviços digitais inclusivos, garantindo que todos os usuários possam interagir e acessar o conteúdo de forma eficaz. Ao seguirem essas diretrizes, os desenvolvedores e designers contribuem para a construção de um ambiente digital mais acessível e igualitário para todos.

4.2. Análise da aplicação de políticas de acessibilidade

Esta subseção tem o objetivo de entender se as diretrizes e recomendações de acessibilidade analisadas na anteriormente estão sendo aplicadas efetivamente. Para isso, trabalhos relacionados a esse tópico serão discutidos a seguir.

O estudo realizado por [Yan and Ramachandran 2019] examinou a acessibilidade de 479 aplicativos Android em 23 categorias diferentes, investigando suas estruturas de interface gráfica e conformidade com diretrizes de acessibilidade. Através do uso da ferramenta IBM Mobile Accessibility Checker (MAC), foram identificados diversos problemas de acessibilidade. Os resultados revelaram que uma parcela significativa das aplicações analisadas apresentava problemas de acessibilidade, sendo que cinco categorias

de *widgets* (TextView, ImageView, View, Button e ImageButton) foram responsáveis pela maioria dos problemas detectados. Esses problemas incluíam a falta de foco em elementos, descrições ausentes, baixo contraste de cores de texto, espaçamento insuficiente entre elementos e tamanhos de fontes menores do que o recomendado. Essas descobertas destacam a importância de abordar questões de acessibilidade na concepção e desenvolvimento de aplicativos móveis para garantir uma experiência inclusiva para todos os usuários.

[Carvalho et al. 2016] investigou 10 aplicativos móveis de municípios brasileiros, cujos objetivos eram de contribuir para a implementação de cidades inteligentes acessíveis aos cidadãos. Os resultados da inspeção de acessibilidade revelaram que muitos dos aplicativos não estavam em conformidade com as diretrizes, apresentando em média 57 instâncias de violações e 11,6 critérios diferentes violados por aplicação. Os problemas encontrados incluíam questões como falta de rotulagem de conteúdo não textual, cabeçalhos, identificação da localização do usuário, contraste de cores, habilitação de interação por meio de gestos de leitores de tela, visibilidade do foco e falta de adaptação de texto contido em imagem. Embora várias cidades tenham desenvolvido soluções importantes para prover serviços básicos de maneira remota para seus cidadãos, há uma forte necessidade de incluir acessibilidade no design desses aplicativos para que todos os indivíduos possam se beneficiar da mesma forma.

O trabalho de [Milne et al. 2014] verificou 9 aplicações móveis na área de saúde para iPhone, nenhum deles atendia aos critérios de acessibilidade baseados nas diretrizes da Apple ou da Seção 508 [U.S. General Services Administration 2017]. A análise mostrou que nenhum dos aplicativos estudados apresentava informações não-textuais em formato alternativo, o que prejudica os usuários com limitações visuais. Isso ressalta a importância de tornar as aplicações móveis acessíveis a todos os usuários, especialmente aquelas relacionadas à saúde, que podem ser particularmente úteis para acessar dados de sensores em *smartphones*.

Segundo o contexto da pesquisa de [de Oliveira and Filgueiras 2018], acessibilidade é pouco levada em conta no processo de construção de um *app* e apenas 7% dos desenvolvedores relataram que suas empresas implementam políticas destinadas a garantir a inclusão no meio digital. Já que muitos profissionais da área não têm conhecimento prévio, [Abuaddous et al. 2016] destaca que implementar práticas inclusivas levaria muito tempo e esforço. Por isso, muitos desenvolvedores relatam que se sentem desencorajados a adotar práticas acessíveis [de Almeida and Gama 2021], e a complexidade desse processo faz com que despriorizem as recomendações inclusivas desde o momento inicial do projeto [Huq et al. 2023].

Dessa forma, fica claro que o desenvolvimento de diretrizes de acessibilidade foi um avanço importante no caminho para tornar os aplicativos inclusivos para um público diverso. Contudo, como apresentado pelas pesquisas mencionadas há pouco, ainda há muito esforço a ser feito para que todos tenham igual acesso a aplicações móveis. Felizmente, existem iniciativas que procuram auxiliar os desenvolvedores a incluir as recomendações de acessibilidade em seus projetos, como veremos à frente.

4.3. Iniciativas que promovem acessibilidade digital

[Ceccarini and Prandi 2019] desenvolveram uma aplicação móvel que visa melhorar o acesso a serviços turísticos para pessoas com deficiência visual. Através de um questioná-

rio com 100 respostas, foram coletados requisitos e *feedbacks* dos usuários, que guiaram o *design* da aplicação final. Testes preliminares demonstraram conformidade com leitores de tela e apreciação dos usuários pelas funcionalidades oferecidas.

Outra pesquisa na área turística, realizada por [Mayordomo-Martínez et al. 2019], investigou a acessibilidade das praias na região de Múrcia, Espanha. O estudo identificou elementos-chave para tornar as praias acessíveis, realizou uma avaliação de campo e desenvolveu um aplicativo móvel para fornecer informações precisas sobre a acessibilidade das praias. O aplicativo proposto não apenas auxilia pessoas com deficiência, mas também sensibiliza as autoridades locais para a importância de criar serviços acessíveis.

[Darvishy et al. 2020] investiga como tornar os dados de aplicativos de mapas acessíveis e compreensíveis para pessoas com deficiência visual. Após identificar a falta de acessibilidade nas principais aplicações de mapas, o projeto propõe uma abordagem que combina saída de voz, sons ambientes e padrões de vibração para apresentar as informações de forma não visual. Um protótipo inicial foi desenvolvido e testado com usuários, apresentando um resultado otimista.

[Ghidini et al. 2016] aborda a complexidade inerente à inovação tecnológica, particularmente no que concerne à inclusão de pessoas com deficiência visual. A pesquisa em questão se propõe a investigar métodos pelos quais *smartphones*, podem tornar-se mais acessíveis, por meio de modalidades diversas de interação. Neste contexto, os autores examinam os hábitos de utilização de *smartphones* por parte de indivíduos com deficiência visual e concebem um protótipo de agenda eletrônica que oferece possibilidades de interação por meio de comandos de voz, teclado e tela sensível ao toque. A agenda eletrônica, concebida como ferramenta de gerenciamento de recursos e eventos, foi submetida à avaliação de potenciais usuários, que enfatizaram a importância do *feedback* sonoro adequado e a interface mais amigável em comparação aos calendários nativos disponíveis.

Uma pesquisa realizada por [WebAIM 2017] sobre leitores de tela, envolvendo 1792 pessoas, espalhadas por todos os continentes, com exceção da Antártida, mostrou que 75,6% deles usava essa ferramenta de acessibilidade em aparelhos móveis da Apple (iPhone, iPad e iPod Touch). Segundo essa análise, o uso de dispositivos iOS entre usuários de leitores de tela foi notavelmente maior que o uso de aparelhos Android. Além disso, quando questionados sobre quais leitores de tela eram mais usados, a ferramenta que apareceu em mais respostas foi o VoiceOver, da Apple. O VoiceOver descreve em voz alta o que está na tela do dispositivo, permitindo que os usuários saibam o que está acontecendo e naveguem pelos aplicativos, ícones, menus e outros elementos da interface.

Para que cada vez mais soluções acessíveis, como as vistas nos parágrafos anteriores, sejam disponibilizadas, é imprescindível disponibilizar mais ferramentas que auxiliem os programadores nesse processo [Sanchez-Gordon and Luján-Mora 2013]. Assim, mecanismos automáticos que contribuam com o uso de recomendações e recursos acessíveis, durante o desenvolvimento de aplicações, podem ajudar a promover um ambiente móvel mais inclusivo para pessoas com deficiência [de Oliveira and Filgueiras 2018].

Pensando nisso, [Silva 2023] propôs um pacote de componentes acessíveis de UI, chamado Code4All, que conta com um catálogo de elementos que visam facilitar a construção de interfaces inclusivas para dispositivos Apple. Tal projeto é de grande significância para a inclusão de pessoas com deficiência visual no mundo digital. Conforme

já mencionado, a maioria dos usuários de leitores de tela mobile utilizam o ecossistema Apple, o que torna essa iniciativa ainda mais relevante.

Dada a importância do projeto Code4All e sua fase ainda inicial, o presente trabalho se propõe a aprimorar esse pacote de elementos. Nas seções seguintes, entenderemos a situação atual do Code4All e, posteriormente, nos debruçaremos nas possibilidades de melhorias a serem implementadas.

5. A ferramenta Code4All

Como apresentado na seção anterior, o desenvolvimento de aplicações móveis adaptadas para pessoas com necessidades diversas ocorre com uma frequência menor do que o necessário. Por isso, foi proposto o pacote Code4All, que disponibiliza componentes assistivos de interface do usuário, com o objetivo de permitir que desenvolvedores *mobile* possam criar telas acessíveis de maneira simples.

A criação do Code4All teve como objetivo encorajar os desenvolvedores a contemplar a diversidade desde o estágio inicial de seus projetos, que, segundo [Huq et al. 2023], é um ponto essencial para facilitar a experiência desses profissionais ao implementar acessibilidade em seus projetos. Esse processo se torna eficiente e simples com o uso dessa ferramenta, facilitando a implementação de práticas de acessibilidade, que, muitas vezes, eram negligenciadas por conta do tempo e complexidade envolvidos no desenvolvimento [de Oliveira and Filgueiras 2018].

5.1. A implementação de Code4All

A ferramenta em questão foi escrita na linguagem Swift, especificamente utilizada para desenvolvimento de aplicações para dispositivos da Apple. Para isso, utilizou-se o Xcode como ambiente de desenvolvimento. Além disso, dentre os possíveis *frameworks* utilizados para construção de interfaces do usuário, foi selecionado SwiftUI, que, associado à ferramenta Accessibility Inspector, apresentada na Figura 3, garante inspeção da qualidade da acessibilidade do código. Ademais, a modularização dos componentes foi viabilizada pelo uso do Swift Package Manager, simplificando a integração e atualização dos elementos de UI em projetos externos.

5.1.1. SwiftUI

SwiftUI é um *framework* da Apple que permite aos desenvolvedores criar interfaces do usuário de forma declarativa e orientada a dados. Assim, pode-se destacar benefícios como código claro e eficiente, simplicidade para criar componentes, atualizações automáticas de UI e compatibilidade multiplataforma [Apple Inc f]. Ao criar uma biblioteca de componentes de UI modulares com SwiftUI, os desenvolvedores podem maximizar a reutilização de código, promover consistência de *design*, facilitar a manutenção do aplicativo e garantir que ele seja adaptável a diversos dispositivos.

Adicionalmente, o SwiftUI fornece uma ampla gama de recursos integrados para a criação de interfaces do usuário acessíveis. Sua integração nativa com tecnologias assistivas, como o VoiceOver, facilita a implementação de práticas de acessibilidade destinadas a atender às necessidades dos usuários com deficiência.

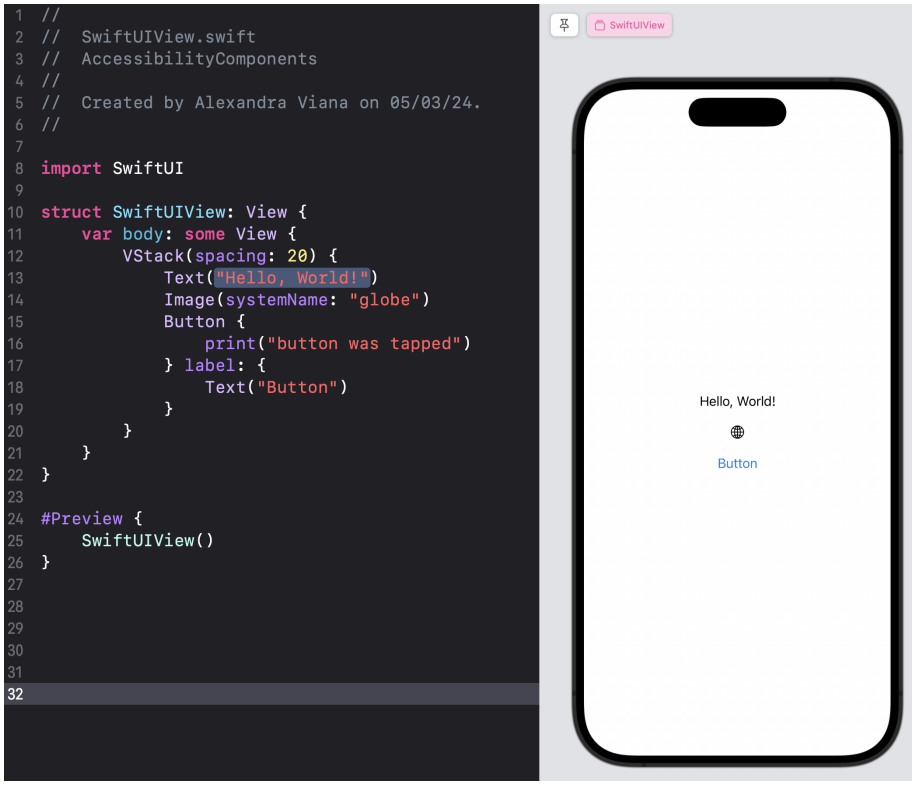


Figura 2. Exemplo de código em SwiftUI.

5.1.2. Accessibility Inspector

O Accessibility Inspector é uma ferramenta presente no Xcode, que desempenha um papel crucial na garantia da inclusão em aplicativos Apple, permitindo aos desenvolvedores identificar e resolver rapidamente questões de acessibilidade. Esta ferramenta possibilita a inspeção detalhada dos elementos da interface do usuário, garantindo rótulos adequados, ordem de foco e compatibilidade com tecnologias assistivas como VoiceOver e Voice Control [Apple Inc c].

Ao integrar o Accessibility Inspector em seus processos de desenvolvimento, os desenvolvedores podem assegurar que seus aplicativos atendam aos padrões de acessibilidade. Assim, fazer uso dessa ferramenta durante a implementação de Code4All, garantiu a eficácia do projeto.

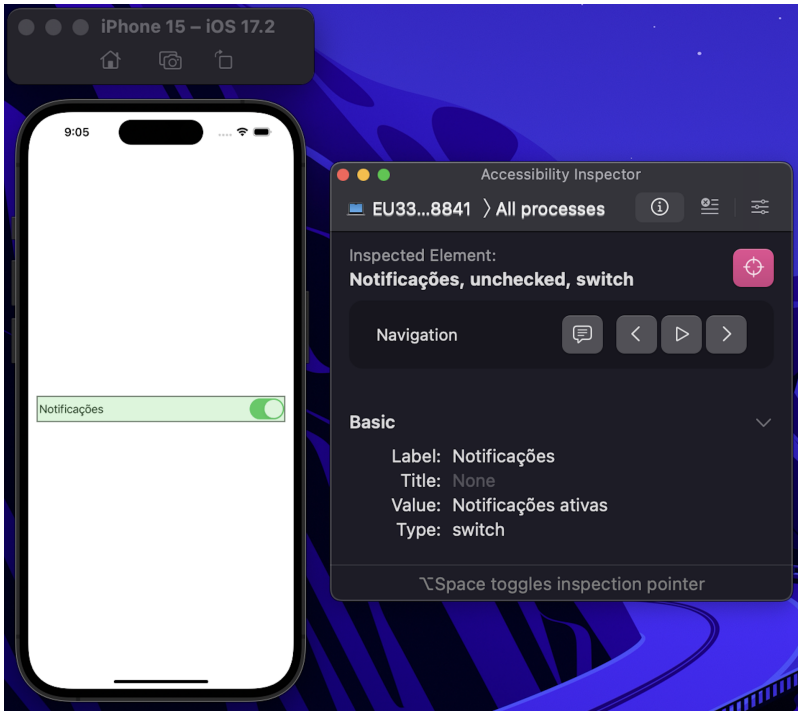


Figura 3. Accessibility Inspector sendo usado para analisar aplicativo rodando em simulador de um iPhone 15.

5.1.3. Swift Package Manager

O Swift Package Manager (SPM) é uma ferramenta desenvolvida pela Apple para gerenciar dependências em projetos Swift. Ele oferece uma abordagem simplificada para adicionar, atualizar e remover bibliotecas externas, facilitando a modularidade e reutilização de código [Apple Inc e]. O SPM proporciona uma maneira eficiente e intuitiva de gerenciar dependências, contribuindo para a simplificação do uso do pacote Code4All. Sua utilização promove uma estrutura organizada e coesa para projetos Swift.

5.2. Componentes disponíveis em Code4All

5.2.1. Text4All

O Text4All representa um componente destinado a exibir texto na interface do usuário, comumente empregado para apresentar rótulos, títulos e informações textuais simples. Esse componente foi idealizado para diferenciar-se do Text, nativo de SwiftUI, ao simplificar a incorporação de atributos de acessibilidade. Isso é alcançado por meio da disponibilização do parâmetro *accessibilityTextInfo*, especialmente concebido para leitores de tela. Adicionalmente, o componente oferece a flexibilidade de personalização de acordo com requisitos individuais, permitindo ajustes visuais como cor, tamanho e fonte.

```
Text4All(text: "Text", accessibilityTextInfo: "Etiqueta Acessível")
```

Figura 4. Exemplo de uso do Text4All.

5.2.2. TextField4All

O TextField4All é um elemento de entrada de texto e conta com uma descrição acessível de *placeholder*. Esse componente foi proposto como uma alternativa ao TextField nativo e faz proveito das características acessíveis já presentes no Text4All, a fim de que o *textHint* presente no campo de texto já inclua uma etiqueta de acessibilidade. Assim, a reutilização do Text4All no TextField4All não só garante a acessibilidade do componente, mas também mantém o princípio de reutilização dos componentes do pacote Code4All.

```
let text4All = Text4All(text: "Text", accessibilityTextInfo: "Etiqueta Acessível")
TextField4All(textHint: text4All, receivedText: $text)
```

Figura 5. Exemplo de uso do TextField4All.

5.2.3. Button4All

Botões são elementos fundamentais nas interfaces de aplicativos móveis e, por isso, foi criado o Button4All, que se destaca por assegurar que o usuário terá acesso de descrições acessíveis ao tentar executar ações específicas na aplicação. Tal funcionalidade é grande importância, já que botões nem sempre são inclusivos para leitores de tela.

O design do Button4All foi concebido apresentando a reutilização de código em mente, já que incorpora o Text4All para prover etiquetas de acessibilidade. Além disso, o Button4All é totalmente customizável para atender às necessidades específicas de cada aplicação.

```
let text4All = Text4All(text: "Text", accessibilityTextInfo: "Etiqueta Acessível")
Button4All(action: buttonTapped, textObject: text4All)
```

Figura 6. Exemplo de uso do Button4All.

5.2.4. Toggle4All

O Elemento Toggle4All é uma ferramenta que permite aos usuários alternar entre dois estados distintos, como "ativo"e "inativo", fornecendo descrições otimizadas para leitores de tela. Focado na clareza das mensagens transmitidas pelos *toggles* e nos valores de ativação e desativação, o Toggle4All garante uma representação mais elucidativa, substituindo os valores padrão "0"e "1"do componente nativo Toggle. A análise da implementação por meio do Accessibility Inspector revela que o Toggle4All proporciona uma inspeção mais abrangente em comparação com o Toggle nativo do SwiftUI, como pode ser verificado na Figura 7, retirada do projeto de [Silva 2023].

Figura 7. Análise comparativa entre o componente Toggle nativo e o Toggle4All. Reprodução [Silva 2023].

5.3. Pontos de melhoria e limitações de Code4All

Code4All marca um avanço significativo no caminho da facilitação e incentivo à adoção de práticas inclusivas no desenvolvimento de aplicativos Apple. Dessa forma, é importante examinar pontos de melhoria dentro desse conjunto de componentes de UI, visando aprimorar a experiência dos desenvolvedores que utilizam esta ferramenta.

5.3.1. Nomeação dos componentes

Para destacar e diferenciar os componentes propostos dos nativos de SwiftUI, foi utilizado o sufixo "4All", que faz referência ao potencial de acessibilidade dos elementos. Como visto anteriormente, paralelo ao Button já existente no *framework* da Apple, foi proposto o Button4All, por exemplo. Porém, se todos os componentes do pacote, atuais e futuros, apresentarem esse sufixo, é possível que algumas nomeações fiquem muito compridas e redundantes. Por isso, seria interessante que fosse apresentada uma alternativa para que os componentes do Code4All continuassem representativos, mas com nomes simplificados.

5.3.2. Funcionalidade dos componentes existentes

Apesar do Text4All ter apresentado o atributo *accessibilityTextInfo* pensando em disponibilizar um texto adaptado ao VoiceOver, essa condição merece ser repensada, considerando que o Text nativo do SwiftUI já é adaptado para o leitor de texto, eliminando a

necessidade de uma descrição assistiva. Outra característica do Text4All que merece mais atenção é o fato de ele ter um tamanho fixo, o que pode ser um problema para usuários que customizam o tamanho da fonte em seus dispositivos. Ao alterar o elemento Text4All, será necessário atualizar também os elementos que o utilizam como parâmetro, sendo eles Button4All, Toggle4All e TextField4All.

Uma característica que gera uma lacuna de acessibilidade no componente TextField4All é o fato de não ser informado ao usuário se é obrigatório ou não preencher o campo de texto. Esse é um atributo essencial para tornar uma interface inclusiva para pessoas com deficiência visual [Ferreira et al. 2012].

Dessa forma, pode-se constatar que seria importante propor melhorias aos elementos atuais de Code4All, para torná-los ainda mais eficientes. Na seção 6, os possíveis aperfeiçoamentos a esses componentes serão discutidos em profundidade.

5.3.3. Quantidade reduzida de componentes

A primeira versão de Code4All disponibiliza quatro elementos diferentes de interface do usuário. Embora o projeto inicial desse pacote já seja de grande relevância, a quantidade limitada de opções de componentes é um ponto de melhoria crucial. Uma variedade mais ampla de opções é essencial para garantir flexibilidade e promover uma experiência de usuário mais rica aos desenvolvedores. A expansão do conjunto de componentes disponíveis acelera o desenvolvimento, reduz a necessidade de soluções personalizadas e contribui para a consistência das interfaces em diferentes aplicativos e plataformas.

5.3.4. Organização do código

Atualmente, na versão inicial de Code4All, todos os componentes estão implementados num mesmo arquivo de código. Porém, tendo como objetivo que o pacote seja construído da melhor forma possível, seria interessante que os componentes fossem escritos em arquivos individuais e organizados por pastas de acordo com suas funcionalidades. Esta reorganização garantiria encapsular funcionalidades específicas em unidades independentes. Aplicar a modularidade permite gerenciar melhor a complexidade do código e promover uma arquitetura mais limpa e escalável, facilitando manutenção e versionamento do código [Gamma et al. 1994].

6. Propondo melhorias a Code4All

Como mencionado anteriormente, o propósito deste trabalho é propor aperfeiçoamentos e soluções para as limitações de Code4All. Dessa forma, espera-se que essa ferramenta se torne cada vez mais completa e possa auxiliar desenvolvedores de aplicações Apple a construir interfaces acessíveis, de maneira simples e com frequência. Nas seções a seguir, serão discutidas de que forma essas melhorias podem ser implementadas.

6.1. Novas nomenclaturas

Inicialmente, propõe-se uma mudança na forma como os componentes do pacote Code4All são nomeados, substituindo o sufixo "4All"(lê-se *for all*, em inglês) por um

prefixo "FA "(iniciais de *for all*) nos nomes de elementos. A transição para um prefixo "FA"proporcionará nomes mais curtos e coesos, facilitando a identificação e o uso dos componentes por parte dos desenvolvedores. Uma boa experiência de usabilidade pode ser um fator determinante para que um *package* seja bem-sucedido.

Seguindo essa linha de pensamento, recomenda-se que o nome do pacote seja mudado para CodeForAll, já que, além de garantir a conformidade com o prefixo FA proposto, não é comum o uso de números em nomeações de *frameworks* do ecossistema Apple. Sendo assim, nas próximas seções deste trabalho, usaremos o nome CodeForAll para nos referir à forma aperfeiçoada da ferramenta Code4All.

6.2. Reanálise e aperfeiçoamento dos componentes existentes

6.2.1. FAText: readaptação do Text4All

O componente anteriormente denominado Text4All tinha a característica de receber não só um texto como entrada, mas também uma descrição assistiva. Porém, o texto nativo de SwiftUI já é adaptado para ser lido pelo VoiceOver. Assim, adicionar uma etiqueta com a mesma informação já presente no texto é uma prática redundante. Para exemplificar essa ocorrência, vamos recriar o exemplo fornecido por [Silva 2023] usando o elemento Text nativo de SwiftUI e compará-lo, usando o Accessibility Inspector, ao Text4All.

Na Figura 8 está replicado o código proposto para exemplificar Code4All no trabalho de [Silva 2023]. A Figura 9 mostra a inspeção das características acessíveis do elemento resultante do uso desse exemplo de Text4All. Enquanto isso, a Figura 10 apresenta um exemplo de uso do elemento Text nativo, recriando o conteúdo da Figura 8. A inspeção desse elemento Text pode ser visualizada na Figura 11. Dessa forma, ao comparar as imagens, pode-se constatar que as diferentes implementações de texto apresentadas em SwiftUI resultam em funcionalidades assistivas idênticas.

```
Text4All(text: "Título da Página", accessibilityTextInfo: "Título da Página")
```

Figura 8. Exemplo de uso de Text4All exemplificada por [Silva 2023].

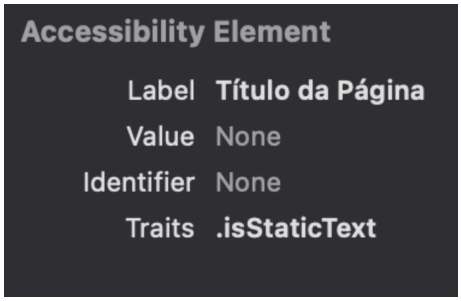


Figura 9. Inspeção da acessibilidade do elemento da Figura 8.

```
Text("Título da Página")
```

Figura 10. Implementação de um texto nativo de SwiftUI.

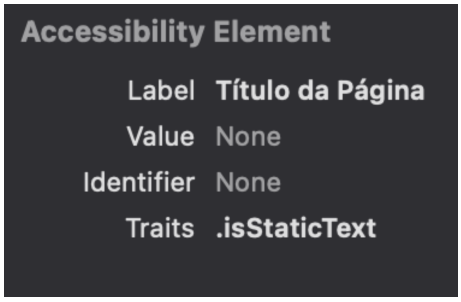


Figura 11. Inspeção da acessibilidade do elemento da Figura 10.

Assim, pode-se analisar que não é necessário um parâmetro de descrição para que esse elemento textual seja acessível. Sendo assim, retirou-se o atributo *accessibilityTextInfo* e, analisando as diretrizes da Apple, foi possível adicionar mais algumas melhorias ao FAText, versão aperfeiçoada do Text4All.

O Text4All permitia que o usuário escolhesse uma fonte com tamanho arbitrário. Embora essa característica forneça flexibilidade ao desenvolvedor, é extremamente recomendado que o texto apresente uma escala dinâmica de tamanho para atender às necessidades do usuário [Matausch et al. 2012]. Por isso, os tamanhos de fonte do FAText são pré-definidos e automaticamente redimensionados de acordo com a preferência do usuário do aplicativo. Alguns exemplos desses valores previamente determinados são: *body*, *title*, *footnote*, *caption* e *headline*. Dada essa mudança, também será assegurado que a fonte utilizada será nativa do sistema, já que garante melhores práticas de inclusão, e sempre sem serifa como recomendado por [Matausch et al. 2012] e [Siebra et al. 2015].

Outra questão importante que o Text4All não abrangia era o peso da fonte utilizada. Assim, o FAText suporta a adaptação do texto de acordo com o peso escolhido pelo desenvolvedor. As opções foram selecionadas previamente, durante o desenvolvimento de CodeForAll, para que garantam boa usabilidade para o público com deficiência visual. Dessa forma, já que a Apple não recomenda o uso de fontes com peso *Ultralight*, *Thin*, e *Light*, nas suas diretrizes, o desenvolvedor usuário do *package* pode escolher entre as opções *Regular*, *Medium*, *Semibold*, ou *Bold*, devido a facilidade aumentada de enxergá-las.

Ainda entre as melhorias implementadas no FAText, está a identificação de se ele é um *header* (cabeçalho) ou não. O uso de *headers* em aplicativos móveis desempenha um papel crucial na acessibilidade e usabilidade [Colorado State University], proporcionando uma estrutura clara e hierárquica para o conteúdo. Cabeçalhos funcionam como pontos de referência, facilitando a navegação e a localização de seções relevantes, especialmente para usuários de leitores de tela. Esses elementos são essenciais para a compatibilidade com recursos de acessibilidade, como *rotors* em dispositivos iOS [Apple Inc a]. Por isso, o FAText passa a apresentar um parâmetro opcional em sua inicialização, chamado *isHeader*, que alerta o sistema quando determinado texto funciona como cabeçalho.

```
FAText("Acessibilidade",
       color: .black,
       font: .headline,
       fontWeight: .bold,
       isHeader: true)
```

Figura 12. Exemplo de uso do novo elemento FAText.

6.2.2. FATextField: readaptação do TextField4All

Inicialmente, o TextField4All foi renomeado para FATextField e deixou de receber um elemento do tipo Text4All, passando a receber um FAText. Adicionalmente, o componente passou a informar se é obrigatório que o usuário preencha o campo de texto. Para isso, basta que o desenvolvedor use o parâmetro *isMandatory* na inicialização do FATextField. Segundo [Ferreira et al. 2012], a nova funcionalidade mencionada é de grande importância, e para que seja efetivada, um asterisco foi adicionado ao texto assistivo do componente quando necessário, de forma que seja percebido pelo leitor de tela.

```
FATextField(
    label: FAText("Nome"),
    receivedText: $textInput,
    isSecureField: $ocultar,
    isMandatory: true
)
```

Figura 13. Exemplo de uso do novo elemento FATextField.

6.2.3. FATextButton: readaptação do Button4All

Como o Button4All recebia um elemento Text4All como parâmetro, agora ele recebe um elemento do tipo FAText, continuando a garantir a reutilização de código no pacote CodeForAll. Foi adicionado o atributo *hint*, para fornecer detalhes da função do botão para o usuário.

As outras características flexíveis desse botão foram mantidas, para que o programador esteja livre para escolher diferentes configurações como cores e bordas. Ademais, para enquadrar-se a renomeação proposta, esse elemento passa a ser chamado de FATextButton, que destaca também o fato desse botão ser representado por um texto.

```
FATextButton(action: {enviarForm()},
             textLabel: FAText("Eviar"))
```

Figura 14. Exemplo de uso do novo elemento FATextButton.

6.2.4. FAToggle: readaptação do Toggle4All

De maneira análoga ao que foi feito com o Button4All, o Toggle4All passa a receber um elemento do tipo FAText na sua inicialização. Além disso, passa a ser chamado de FAToggle.

Uma adição feita ao FAToggle é que o desenvolvedor pode escolher quais serão as etiquetas lidas quando o componente estiver ativo ou inativo. Apesar do Toggle4All já apresentar um valor mais descritivo que o nativo, ele não permitia que o programador usasse identificações diferentes de "enabled"e "disabled". Por isso, o FAToggle permite que esses valores sejam nomeados da forma que o desenvolvedor preferir, porém "enabled"e "disabled"continuam sendo os valores *default*.

```
FAToggle(
    enableToggle: $isEnabled,
    toggleLabel: FAText("Notificações"),
    enabledLabel: "Notificações ativas",
    disabledLabel: "Notificações inativas"
)
```

Figura 15. Exemplo de implementação do novo elemento FAToggle.

6.3. Propondo novos componentes

Idealizando a construção de um pacote de componentes de UI eficiente, foi considerada a importância de disponibilizar uma ampla variedade de opções de elementos modulares. Reduzir a necessidade de criação de soluções personalizadas por parte dos desenvolvedores resulta numa experiência mais ágil e produtiva de desenvolvimento. [Gamma et al. 1994] enfatizam como a modularização e reuso de código são valiosos em um projeto, promovendo uma arquitetura de software mais robusta e fácil de manter.

Dessa forma, um dos pontos principais deste trabalho é desenvolver novos elementos para contribuir com o catálogo de opções de Code4All. Oferecer aos desenvolvedores uma gama mais ampla de ferramentas prontas para uso simplifica o desenvolvimento de *softwares* acessíveis. Ao analisar pontos importantes relacionados à acessibilidade de componentes visuais em aplicativos e compará-los ao que já está disponível no pacote inicial, propõe-se 4 novos componentes, que serão apresentados a seguir.

6.3.1. FADescriptiveImage

Ao usar conteúdo visual, como fotos e ilustrações, em aplicativos, é fundamental garantir que ele seja acessível a todos [University of California San Francisco] [W3C Web Accessibility Initiative 2022]. Por isso, é importante assegurar que o conteúdo exista em um formato alternativo, para que pessoas com dificuldades visuais possam também perceber ou compreender o que está disposto na interface [Siebra et al. 2015]. Sendo assim, a Apple, através da Human Interface Guidelines, recomenda que os desenvolvedores forneçam descrições alternativas para todas as imagens que transmitam algum

significado, a fim de que os usuários do VoiceOver experimentem totalmente o aplicativo [Apple Inc d].

Nativamente, em SwiftUI, é possível usar o componente Image e passar o nome do arquivo que deverá ser mostrado, bem como um componente Text para servir como a descrição daquele elemento. Contudo, essa é só uma das formas de inicializar uma imagem e apresenta limitações, já que não é possível inicializar um ícone nativo da Apple, chamado de SF Symbol, juntamente a uma descrição inclusiva. Essa informação precisaria ser adicionada posteriormente.

Essas várias formas de inicializar uma imagem em SwiftUI podem não deixar explícita a maneira ideal de adicionar uma figura com suporte assistivo em um projeto. Dessa forma, pode-se dizer que um componente que facilite a implementação de uma imagem acessível seria um grande ganho para desenvolvedores do ecossistema Apple.

Como visto antes, a primeira proposta de Code4All não conta com um componente para imagens descritivas, e essa é uma peça fundamental na criação de interfaces inclusivas para usuários com deficiências [Siebra et al. 2015]. Dessa forma, o presente trabalho propõe o elemento FADescriptiveImage, que permite que o desenvolvedor possa facilmente adicionar descrições a uma figura, garantindo que seu projeto seja acessível a pessoas com problemas de visão.

O FADescriptiveImage oferece uma abordagem simplificada para tornar imagens acessíveis. Ao fornecer uma descrição textual através da propriedade *accessibilityDescription*, os desenvolvedores podem garantir que as imagens sejam compreendidas por leitores de tela e outros dispositivos assistivos.

Um dos destaques desse componente é sua flexibilidade para receber tanto símbolos nativos do sistema como imagens adicionadas pelo usuário do *package*. Para isso, basta usar a *flag* de *isSystemImage* para identificar aquele elemento. Por padrão, ele não será considerado como símbolo nativo da Apple, a menos que o desenvolvedor indique o contrário.

Figura 16. Exemplos visuais do FADescriptiveImage, primeiro com uma imagem externa ao sistema e, em seguida, com uma imagem nativa (a). Em (b) foi mostrado o código utilizado para obter o resultado de (a). A inspeção de acessibilidade dos elementos dos exemplos foi introduzida, respectivamente, por (c) e (d).

6.3.2. FADecorativeImage

Diferente do apresentado na seção anterior, quando uma imagem é meramente decorativa e não possui informação relevante, é preferível ocultá-la das tecnologias assistivas. Essa sugestão é a mesma para o desenvolvimento *web* segundo [W3C Web Accessibility Initiative 2022] e para o desenvolvimento iOS segundo a Human Interfaces Guideline [Apple Inc b].

Um exemplo de situação em que seria inadequado uma descrição assistiva para uma imagem seria um ícone de erro ao lado do texto "erro", como mostrado na Figura 17. Se essa imagem for acessível, numa aplicação Apple, o usuário do VoiceOver ouvirá "erro"duas vezes, o que prolonga e frustra a navegação, além de tornar redundantes as informações apresentadas.

Figura 17. Exemplo em que a descrição assistiva de uma imagem seria inadequada.

Nesse sentido, propõe-se adicionar um novo componente ao pacote Code4All, chamado FADecorativeImage. Ao integrá-lo em um aplicativo, fica garantido que as imagens decorativas sejam tratadas de maneira apropriada, sem interferir na experiência de

acessibilidade. Este elemento foi desenvolvido de forma que não será lido pelo VoiceOver.

O FADecorativeImage traz um benefício semelhante ao do DescriptiveImage, que é o de usar uma mesma estrutura de inicialização, seja a imagem um um ícone nativo (SF Symbol) ou não. Novamente, basta usar o parâmetro *isSystemImage*, para identificar o elemento. Essa qualidade é importante, visto que, mais uma vez, SwiftUI fornece formas diferentes para o usuário usar uma imagem decorativa. Essa questão pode ser confusa para um desenvolvedor que não pesquisou a fundo sobre o tópico, e a complexidade pode desencorajá-lo a aplicar acessibilidade [Huq et al. 2023].

Portanto, para usar esse elemento, basta fornecer o nome do arquivo, ou do símbolo que será usado, e informar se é um ícone nativo. Assim, ao adotar o FADecorativeImage, o desenvolvedor eficientemente garante que imagens que não contêm informações relevantes para a compreensão do conteúdo sejam interpretadas corretamente por leitores de tela.

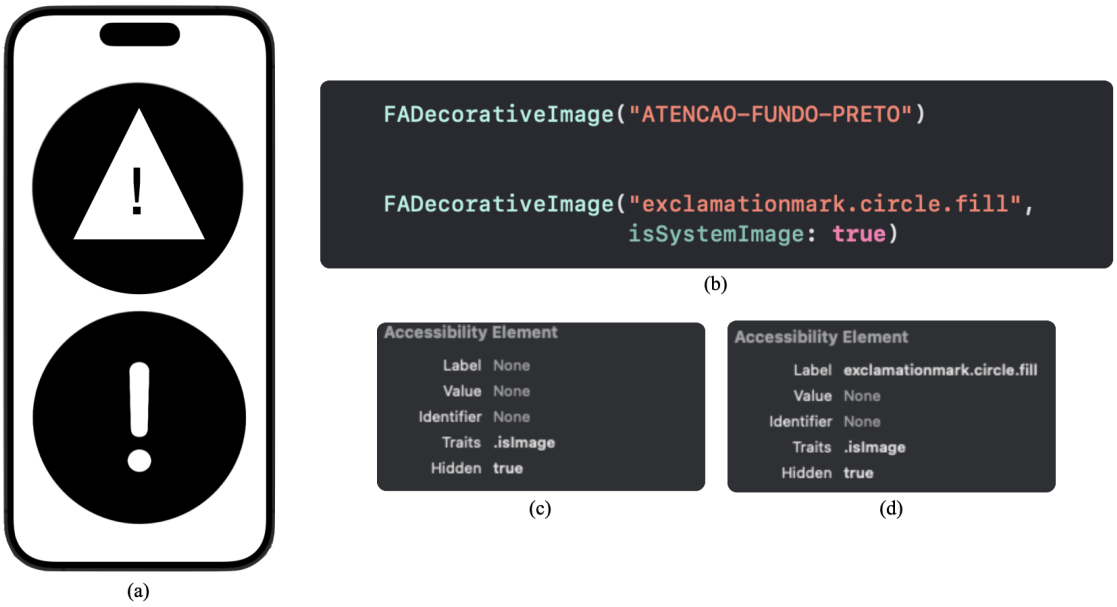


Figura 18. Exemplos visuais do FADecorativeImage, primeiro com uma imagem externa ao sistema e, em seguida, com uma imagem nativa (a). Em (b) foi mostrado o código utilizado para obter o resultado de (a). A inspeção de acessibilidade dos elementos dos exemplos foi introduzida, respectivamente, por (c) e (d), destaca-se aqui o parâmetro que oculta a etiqueta assistiva.

6.3.3. FAImageButton

Muitos padrões de *design* em aplicativos móveis usam botões baseados em imagens para as funcionalidades principais [Ross et al. 2018]. Um componente-chave da acessibilidade para usuários de leitores de tela é a rotulagem desse tipo de botão. Porém, infelizmente, esses elementos não recebem etiquetas assistivas na frequência recomendada [Ross et al. 2020].

A pesquisa [Ross et al. 2018], que analisou 5,753 aplicativos Android, revelou

que 2,638 aplicativos (45,9%) possuem pelo menos 90% de seus botões com imagens sem rótulos. Apesar dessa pesquisa ter sido realizada em dispositivos Android, a falta de descrições adequadas representa um desafio significativo para a acessibilidade de pessoas com deficiência em todas as plataformas de aplicativos móveis.

Assim, ainda pensando em adaptar o pacote Code4All para comportar figuras acessíveis, é importante que exista um componente para imagens funcionais, ou seja, botões que, ao invés de representação textual, apresentam uma figura ou ícone. Nesse contexto, surge o componente FAImageButton.

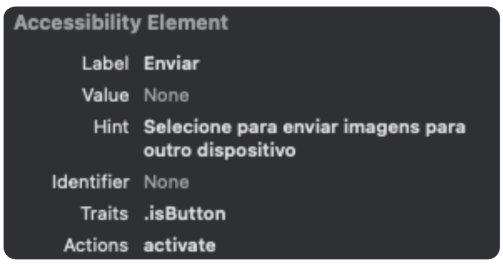
É essencial fornecer alternativas de texto claras e descritivas para imagens funcionais, garantindo que os usuários de leitores de tela possam compreender completamente as funcionalidades do aplicativo. Por isso, o componente FAImageButton desempenha um papel crucial ao permitir a criação de botões com figuras acompanhados por descrições claras e inclusivas da ação associada ao elemento.

Para implementar o FAImageButton, é necessário fornecer a ação que ele executará e um componente do tipo DescriptiveImage, o qual já carrega uma imagem com descrição assistiva. Essa abordagem promove a reutilização de código e simplifica a incorporação de práticas acessíveis, poupando os desenvolvedores do esforço de criar estruturas personalizadas para garantir a inclusão.

```
FAImageButton(
    action: {},
    image: FADescriptiveImage("paperplane.fill",
                              imageAccessibilityDescription: "enviar",
                              isSystemImage: true),
    hint: "selecione para enviar imagens para outro dispositivo"
)
```

(a)

(b)



(c)

Figura 19. Implementação do FAImageButton (a), juntamente com o resultado visual do código apresentado (b). Em (c) foi realizada a inspeção das características acessíveis do componente.

6.3.4. FALoadingView

A entrega de *feedbacks* durante o carregamento de informações é crucial para garantir uma experiência de usuário satisfatória [Siebra et al. 2015]. No entanto, essa comunicação representa um desafio significativo para pessoas com deficiência, especialmente para

aquelas que dependem de leitores de tela. O componente ProgressView nativo do SwiftUI não oferece avisos e informações claras aos usuários que utilizam VoiceOver, o que pode resultar em uma experiência frustrante e confusa durante o processo de carregamento de dados.

Por isso, este projeto inclui o componente FALoadingView ao catálogo do CodeForAll, cuja importância reside na sua capacidade de informar aos usuários de leitores de tela sobre um processo de carregamento em andamento. Além disso, eles serão notificados quando o carregamento for concluído. Isso promove uma experiência mais inclusiva e transparente para todos, fornecendo *feedback* claro e oportuno sobre o estado do carregamento de informações. Isso não apenas ajuda os usuários com visão limitada a entender o que está acontecendo no aplicativo, mas também reduz a ansiedade e a incerteza ao aguardar o carregamento de conteúdo.

Para utilizar o FALoadingView num projeto, o desenvolvedor precisa fornecer algumas informações essenciais como parâmetros. Elas incluem uma descrição da *view* em questão, uma explicação opcional que descreve a ação de carregamento, bem como mensagens de anúncio para indicar quando o carregamento aparece e desaparece, que serão lidas pelo VoiceOver. Esses parâmetros são fundamentais para tornar o FALoadingView mais informativo e adaptável às necessidades específicas de cada aplicativo. Em seguida, vamos analisar um exemplo de uso da FALoadingView na Figura 20.

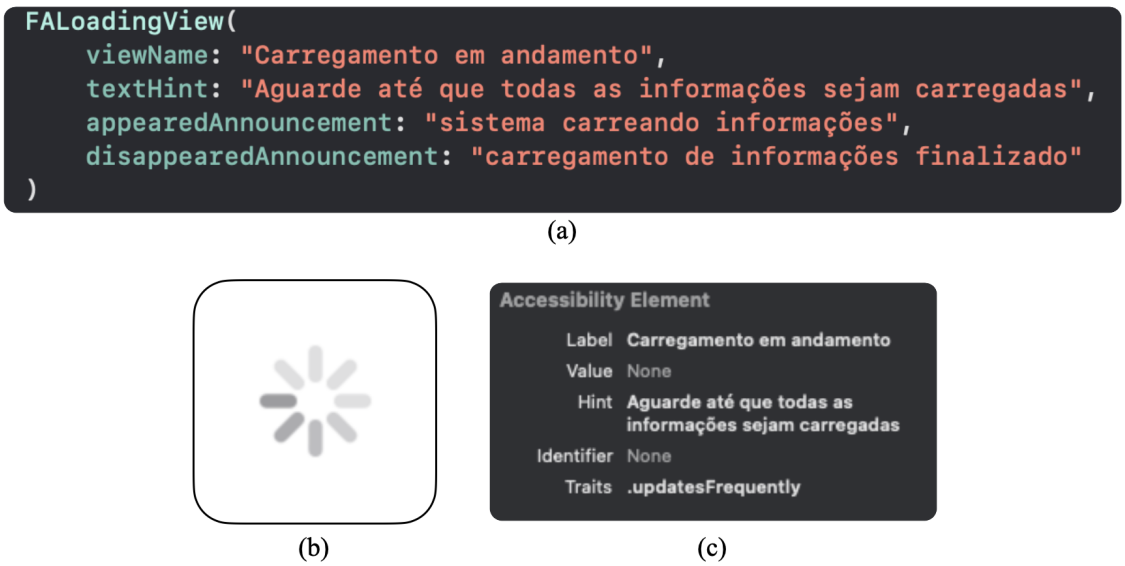


Figura 20. Implementação do FALoadingView (a), juntamente com o resultado visual do código apresentado (b). Em (c) foi realizada a inspeção das características acessíveis do componente.

O resultado de usar o trecho de código de Figura 20(a) seria a exibição de um indicador de progresso acessível, que alerta aos usuários de leitor de tela que o aplicativo está carregando informações. Quando o elemento apresentado na Figura 20 aparece na tela, o VoiceOver anuncia "sistema carregando informações", indicando o processo em andamento. Da mesma forma, quando o FALoadingView desaparece da tela, o VoiceOver anuncia "carregamento de informações finalizado", informando ao usuário que o processo

foi concluído com sucesso.

É importante destacar que esse componente não só entrega uma funcionalidade que não estava disponível em SwiftUI para *views* de carregamento, mas também possibilita que desenvolvedores apliquem acessibilidade a esses elementos de forma muito simples e rápida. Sem FALoadingView, seria necessário que esses profissionais tivessem conhecimento prévio sobre notificações e descrições assistivas para implementar funções semelhantes.

Dessa forma, conclui-se que FALoadingView representa uma iniciativa valiosa tanto para desenvolvedores, como para usuários do VoiceOver. Esse componente fornece *feedback* auditivo claro sobre o *status* do aplicativo, garatindo compreensão plena sobre o processo de carregamento de informações. Assim, a adoção do FALoadingView contribui para um ambiente digital mais inclusivo e amigável para todos os usuários.

6.4. Organização estrutural do projeto

O projeto inicial de Code4All, representado na Figura 21, apresentava um único arquivo de código que comportava todos os componentes. Quando realizada a validação da versão inicial do pacote estudado, um dos *feedbacks* recebidos foi de que poderia haver um arquivo individual para cada componente, de maneira que o desenvolvedor reconhecesse rapidamente quais funcioalidades estão disponíveis no *package*.

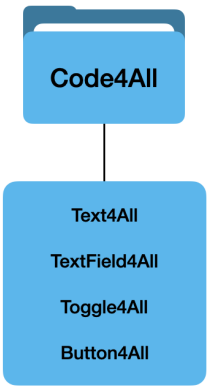


Figura 21. Hierarquia de arquivos de Code4All.

Além da identificação mais ágil dos componentes disponibilizados pelo pacote, essa sugestão contribui para a modularização do projeto, promovendo uma compreensão mais clara, organizada e intuitiva do código. A fragmentação dos componentes em arquivos distintos facilita a localização, compreensão e alteração de partes específicas do código, conferindo uma melhor experiência para desenvolvedores que possam utilizá-lo ou revisá-lo no futuro [Gamma et al. 1994]. Essa abordagem foi aplicada no CodeForAll, como pode ser visto na Figura 22, e não apenas otimiza o processo de desenvolvimento, mas também contribui para a qualidade e sustentabilidade do projeto como um todo.

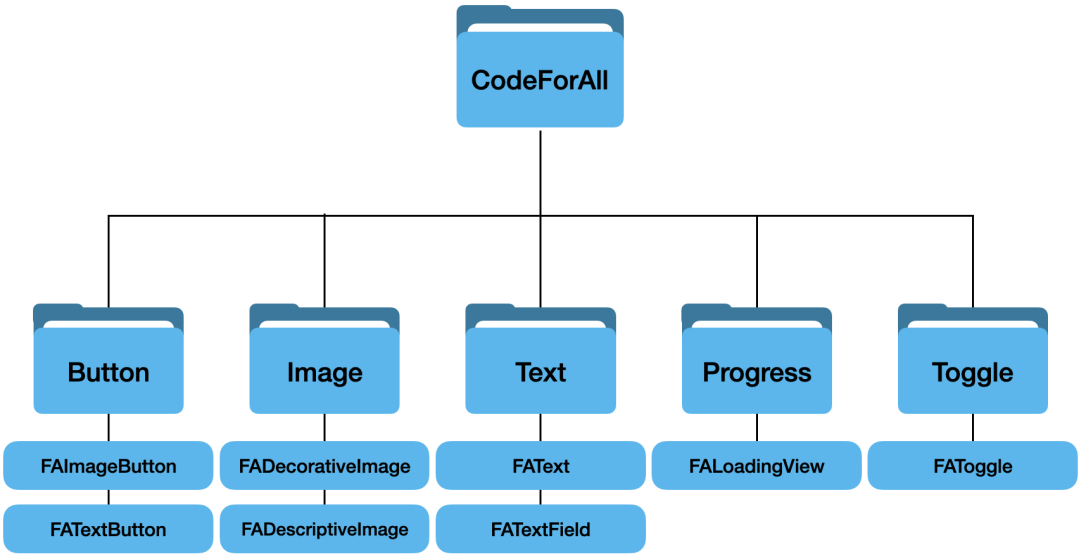


Figura 22. Hierarquia de pastas e arquivos de CodeForAll.

A Figura 23 esquematiza os elementos de Code4All em comparação a CodeForAll, que apresenta 4 versões refinadas dos elementos do primeiro projeto, bem como 4 novos componentes. Além disso, nesse esquema também é possível visualizar a divisão do conteúdo do pacote em arquivos de código diferentes.

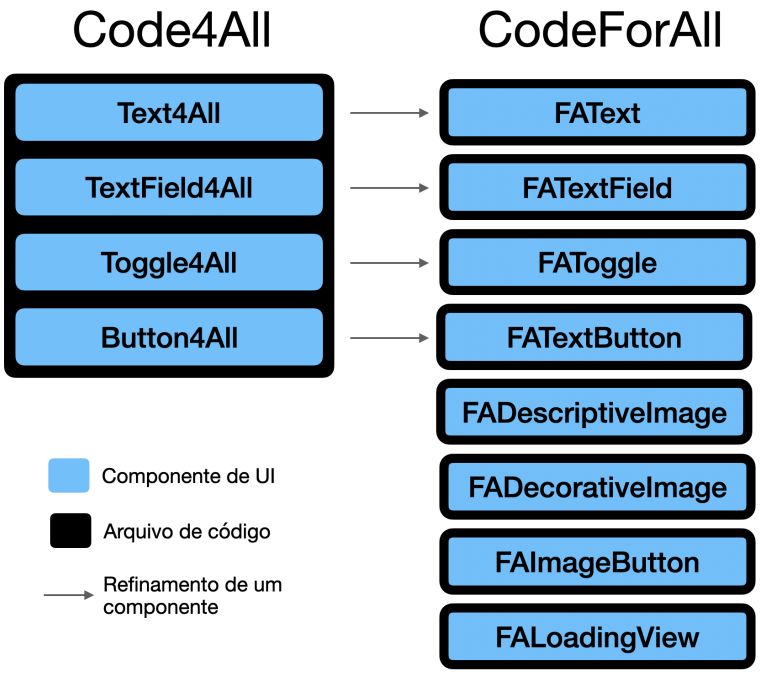


Figura 23. Comparação de componentes de Code4All e CodeForAll.

7. Método de avaliação

A metodologia utilizada envolve uma pesquisa empírica quantitativa e qualitativa. A coleta de dados foi realizada por meio de um questionário digital estruturado em diferentes seções, com o objetivo de obter *insights* sobre a usabilidade e a eficácia dos componentes de CodeForAll. O público-alvo abrangeu profissionais que atuam como desenvolvedores e analistas de aplicativos do ecossistema Apple. A Tabela 1 identifica a profissão, o tempo de experiência com a linguagem Swift, tempo de experiência especificamente com o *framework* SwiftUI e o gênero de cada participante da pesquisa. Vale ressaltar que a maioria desses indivíduos atuava na área de TI anos antes de iniciar a programar em Swift.

	Experiência: Swift	Experiência: SwiftUI	Ocupação	Gênero
P1	4 anos	4 anos	Eng. de Software	Masculino
P2	4 anos	2 anos	Eng. de Software Sênior	Masculino
P3	2 anos	2 anos	Analista de Qualidade iOS	Feminino
P4	2 anos	2 anos	Gerente Técnico de Software	Masculino
P5	4 anos	4 anos	Eng. de Software	Feminino
P6	6 anos	4 anos	Líder Técnico	Masculino
P7	5 anos	4 anos	Eng. de Software Sênior	Masculino
P8	4 anos	3 anos	Eng. de Software Especialista	Masculino
P9	6 anos	2 anos	Eng. de Software Júnior	Masculino
P10	6 meses	6 meses	Eng. de Software Sênior	Masculino
P11	3 anos	3 anos	Eng. de Software Sênior	Masculino

Tabela 1. Listagem dos profissionais que participaram da pesquisa.

Inicialmente, o questionário apresentou cada um dos componentes do pacote CodeForAll aos participantes da pesquisa, que foram instruídos a tentar utilizá-los e a inspecionar seus atributos por meio do Accessibility Inspector para *previews* no Xcode. Esse processo teve como objetivo permitir uma avaliação prática da acessibilidade e funcionalidade dos componentes, identificando possíveis falhas ou áreas de melhoria.

Após a avaliação prática dos componentes, o formulário apresentou algumas perguntas para entender algumas opiniões e percepções gerais dos desenvolvedores ao utilizar o CodeForAll. O principal objetivo dessa seção foi analisar se o pacote foi visto pelos profissionais em questão como uma ferramenta realmente útil e prática, no contexto de impulsionar a adoção de práticas acessíveis no desenvolvimento de aplicativos móveis.

Por último, para examinar a usabilidade do CodeForAll, foi realizada a aplicação do questionário System Usability Scale (SUS)[Brooke 1995]. Esta escala consiste em 10 perguntas que avaliam a usabilidade dos componentes percebida pelos desenvolvedores. As perguntas do SUS são graduadas em uma escala em formato Likert [Likert 1932] de cinco pontos, variando de "discordo fortemente"a "concordo fortemente".

Ao final do processo de coleta de dados, as respostas serão analisadas para identificar padrões, tendências e áreas de melhoria nos componentes do pacote CodeForAll. Os resultados da avaliação serão utilizados para orientar ajustes e melhorias nos componentes, visando torná-los mais eficientes para os desenvolvedores Swift.

7.1. Colocando os componentes em prática

A parte inicial do questionário propôs que os participantes fizessem uso dos componentes do CodeForAll e analisassem, na aba de inspecionar o *preview* fornecido por SwiftUI, se as informações de acessibilidade coincidiam com o elemento criado. Em seguida, algumas perguntas eram feitas sobre o uso daquele componente específico e o desenvolvedor poderia responder entre "Sim"ou "Não". As três perguntas feitas para cada um dos componentes eram as seguintes:

- Você conseguiu usar o componente?
- As informações acessíveis coincidem com o elemento que você criou?
- Faria sentido, como desenvolvedor, usar esse componente nos seus próximos projetos?

Já que o desafio proposto para o componente FATextField envolvia a criação de um campo de texto obrigatório, foi adicionada uma quarta pergunta. Essa adição tinha o objetivo de pedir para que o profissional verificasse se o sistema identificava que o preenchimento do campo textual era necessário.

7.2. Percepções gerais

Ao finalizar os desafios práticos de implementação, três questionamentos foram levantados para entender se o uso do pacote CodeForAll poderia atingir positivamente a forma como os participantes do experimento desenvolvem aplicativos. As perguntas desta seção do formulário foram inspiradas nas mesmas questões levantadas por [Silva 2023] e as respostas possíveis eram "Sim", "Não"e "Talvez". Foram elas:

- Você acredita que o uso do CodeForAll facilita a implementação de componentes acessíveis?
- O uso dos componentes de CodeForAll ajudou você a entender como um aplicativo pode ser mais acessível?
- Com o CodeForAll, você se sentiria mais propenso a tornar seus projetos acessíveis?

Por fim, os entrevistados foram questionados, ainda, sobre sua preferência de uso de nomenclatura do pacote. Essa pergunta teve o objetivo de esclarecer se a comunidade de desenvolvedores preferia a nomeação antiga ou a proposta por esse projeto. Assim, eles deveriam escolher entre uma das duas alternativas seguintes:

- Como desenvolvedor, prefiro a versão atual do pacote, que se chama CodeForAll, e cujos elementos apresentam o prefixo "FA"(FAText, FAToggle, ...)
- Como desenvolvedor, prefiro a versão antiga do pacote, que se chama Code4All, e cujos elementos apresentam o sufixo "4All"(Text4All, Toggle4All, ...)

7.3. Escala de Usabilidade do Sistema

A última parte do questionário teve o objetivo de qualificar a usabilidade da ferramenta CodeForAll. Para isso, utilizou-se a escala chamada System Usability Scale (SUS) proposta por [Brooke 1995]. Esta escala envolve 10 itens que são, por sua vez, classificados de acordo com uma escala em formato Likert [Likert 1932] de cinco pontos, variando de "discordo fortemente"a "concordo fortemente". [Bangor et al. 2009] descobriram que a escala SUS apresenta um alto nível de confiabilidade numa vasta gama de interfaces, como celulares, televisões e *web*. Os dez itens avaliados foram os mesmos propostos por [Brooke 1995], sendo eles:

1. Acho que gostaria de utilizar este produto com frequência
2. Considerei o produto mais complexo do que o necessário
3. Achei o produto fácil de utilizar
4. Acho que necessitaria de ajuda de um técnico para conseguir utilizar este produto
5. Considerei que as várias funcionalidades deste produto estavam bem integradas
6. Achei que este produto tinha muitas inconsistências
7. Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este produto
8. Considerei o produto muito complicado de utilizar
9. Senti-me muito confiante ao utilizar este produto
10. Tive que aprender muito antes de conseguir lidar com este produto

O cálculo de pontuação de usabilidade SUS é obtido por meio da soma da contribuição individual de cada item. Para os itens ímpares, subtrai-se 1 ponto do valor atribuído à resposta, que pode ter sido entre 1 e 5. Para os itens pares, o cálculo é feito ao se subtrair o valor atribuído a resposta (também entre 1 e 5) do total de 5 pontos. Para o cálculo do *score* total, os valores obtidos a partir dos itens pares e ímpares foram somados e multiplicados por 2,5.

Após o profissional responderem aos itens da escala SUS, foi adicionada uma pergunta de resposta expositiva e opcional, em que o participante era questionado se gostaria de fornecer uma crítica ou sugestão para o CodeForAll.

8. Resultados

8.1. Desafios práticos com componentes

O pacote CodeForAll disponibiliza 8 componentes que foram avaliados por 11 desenvolvedores diferentes. Assim, temos um total de 88 avaliações de componentes de UI realizadas. Como mencionado anteriormente, cada análise levou em conta se o participante do experimento conseguiu usar o componentes, se os parâmetros de acessibilidade estavam funcionando corretamente e se faria sentido para aquele profissional fazer uso do elemento em questão futuramente.

Apenas o profissional P7 relatou dificuldade ao utilizar um dos componentes: o FALoadingView. Nas outras 87 avaliações realizadas, a implementação dos componentes de CodeForAll foi bem-sucedida e as respostas indicam que todas as informações acessíveis coincidem com os elementos criados. Porém, o participante P7 relatou que, ao testar o componente FATextField, não foi observado um asterisco na etiqueta assistiva do componente.

Acredita-se que esse problema teve relação com o fato de não ter ficado explícito, para o desenvolvedor P7, que ele deveria usar o parâmetro opcional *isMandatory* ao realizar a análise do componente de campo de texto. Essa discussão foi levantada por outros participantes que, apesar de terem conseguido cumprir o desafio, relataram ter dificuldade de notar que deveriam utilizar esse parâmetro.

Cada avaliação incluía também o questionamento de se faria sentido, para desenvolvedor, usar o componente apresentado nos seus próximos projetos. Das respostas, apenas 7.9% foram negativas e foram referentes aos componentes FAText, FATextField, FADecorativeImage, FATextButton e FAImageButton, sendo FADecorativeImage o único que recebeu mais de uma resposta "Não".

Portanto, foi observado que, de maneira geral, os profissionais participantes da pesquisam tiveram êxito ao tentar utilizar CodeForAll e validaram seu caráter acessível. Ademais, em mais de 90% das vezes em que foram perguntados se usariam um dos componentes futuramente, os desenvolvedores afirmaram que sim.

8.2. Percepções gerais

Dos 11 participantes da pesquisa, todos responderam que o uso de CodeForAll os ajudou a entender como um aplicativo pode se tornar mais acessível. A unanimidade entre os profssionais também foi atingida quando questionados se o pacote proposto fez com que se sentissem mais proprensos a tornar seus projetos futuros inclusivos para pessoas com deficiências visuais.

Já a pergunta "você acredita que o uso do CodeForAll facilita a implementação de componentes acessíveis?"recebeu 10 respostas "Sim"e 1 resposta "Talvez". Um caso semelhante ocorreu nas respostas à pergunta quanto a preferência de nomeação do pacote e dos componentes, apenas um participante elegeu a nomenclatura antiga como favorita.

Em suma, os resultados sugerem que CodeForAll foi bem recebido pelos desenvolvedores, promovendo uma maior conscientização sobre acessibilidade e incentivando a implementação de tecnologias inclusivas em seus projetos. Um dos grandes desafios do presente trabalho era de auxiliar profissionais no processo de desenvolvimento de aplicativos inclusivos. Felizmente, os resultados da pesquisa indicam que a adotar práticas acessíveis na programação tornou-se mais tangível e eficiente através de CodeForAll. Além disso, as respostas permitem considerar assertiva a mudança de nomeação proposta por esse projeto.

8.3. Usabilidade de CodeForAll

Segundo [Bangor et al. 2009] o *score* de usabilidade total obtido pelo questionário SUS, que varia entre 0 a 100 pontos, pode ser categorizado, conforme esquematizado na Figura 24. Se a pontuação média for inferior a 50, o sistema não possui um nível aceitável de usabilidade. Uma pontuação entre 50 e 70 é classificada como indicativa de aceitabilidade, enquanto uma pontuação superior a 70 é aceitável.

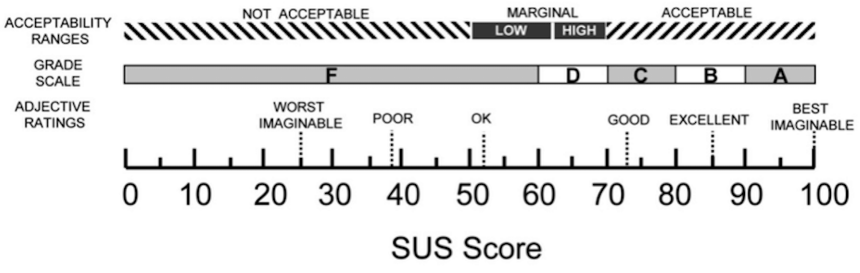


Figura 24. Classificações de pontuações de aceitabilidade e escala de notas do SUS segundo [Bangor et al. 2009].

A Figura 25 apresenta a frequência absoluta de cada pontuação SUS obtida através das respostas do questionário proposto. A menor pontuação de usabilidade do CodeForAll foi de 72,5. A maior pontuação foi 100, a maior possível. Já 92,5 foi o *score* obtido mais

vezes. Conforme direcionado por [Brooke 1995], foi realizada uma média aritmética para obter a pontuação final, que foi de **86,36**.

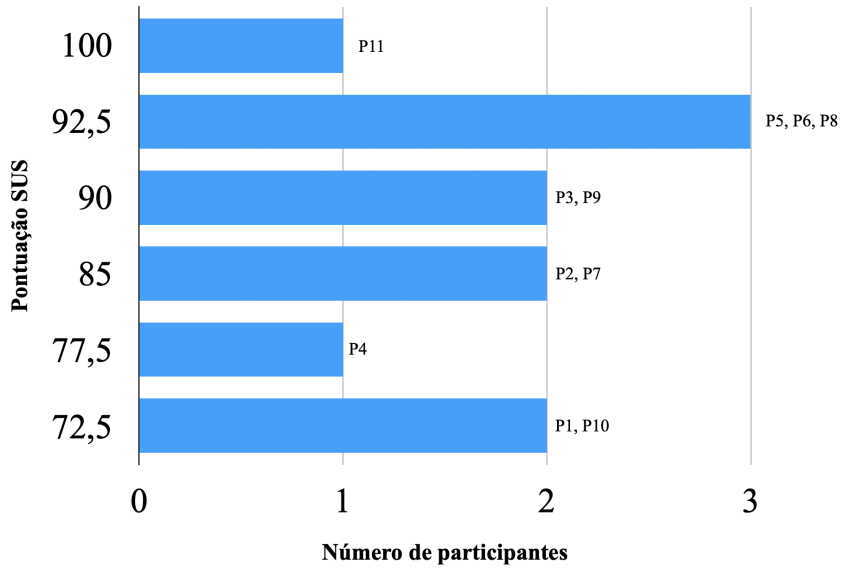


Figura 25. Distribuição por frequência absoluta das pontuações obtidas pela Escala SUS através do questionário realizado.

Dessa forma, conclui-se que os profissionais em questão concordariam em usar o CodeforAll com frequência, por oferecer uma experiência de usuário satisfatória. Além disso, pode-se dizer que o pacote proposto foi considerado uma ferramenta de fácil usabilidade. Esse resultado é muito valioso, uma vez que o projeto em questão se propôs a entregar um material que simplificasse o uso de práticas acessíveis no desenvolvimento de aplicativos.

8.4. Sugestões e possíveis pontos de melhoria

O *feedback* dos desenvolvedores obtido em relação ao CodeForAll revela *insights* valiosos para aprimorar a usabilidade e a funcionalidade do pacote. Foram identificadas diversas oportunidades de melhoria e sugestões que podem otimizar a utilização e a integração do código com o ecossistema da Apple. Em seguida, serão discutidas algumas recomendações apresentadas.

Na inicialização dos componentes FADescriptiveImage e FADecorativeImage, foi sugerida uma nomenclatura que se assemelhe mais aos padrões existentes na Apple. Nesse sentido, seria vantajoso adotar duas formas de inicialização distintas para um mesmo componente, sendo uma para imagens nativas do sistema e uma para imagens externas. Assim, seria evitada a necessidade de um booleano para indicar a origem da imagem.

Ainda sobre a forma com que o CodeForAll introduz imagens, foi levantado que o carregamento de imagens assíncronas pode ser uma atividade limitada para desenvolvedores que utilizarem esse *package*. Nesse contexto, foi sugerido o uso de modificadores que seriam aplicados a uma imagem qualquer de SwiftUI.

O componente FATextField adiciona um asterisco à *label* assistiva quando é obrigatório o preenchimento do campo de texto. Foi sugerido, pela participante P3, analista de qualidade iOS, que esse asterisco ficasse também disposto no texto visível para usuários do aplicativo em desenvolvimento.

Outra importante questão levantada foi em relação ao suporte ao *Dark Mode*. Essa funcionalidade destacou-se como uma valiosa oportunidade de trabalho futuro. A adaptação da tela ao *Dark Mode* é importante não só por proporcionar uma experiência mais personalizável aos usuários, mas também pode ser mais acessível para pessoas com sensibilidade à luz ou com dificuldades visuais, proporcionando maior contraste e facilitando a leitura.

Ademais, um ponto de melhoria destacado pela participante P5 foi a possível criação de uma documentação para o CodeForAll. Dessa forma, seria possível fornecer informações detalhadas sobre o uso de cada componente disponível e como eles seriam úteis para pessoas com deficiência visual.

As sugestões fornecidas pelos participantes da pesquisa realizada revelaram informações valiosas sobre como o CodeForAll pode ser aprimorado, de forma a melhorar cada vez mais a experiência dos desenvolvedores que o utilizarem. Todas as recomendações obtidas são grandes oportunidades a serem consideradas para trabalhos futuros.

9. Considerações finais

Apesar da existência de diretrizes de acessibilidade bem estabelecidas, é surpreendente observar que apenas uma minoria de aplicativos móveis faz uso efetivo delas [Yan and Ramachandran 2019]. Esta lacuna na implementação de práticas acessíveis é um desafio significativo que afeta diretamente a experiência de uso de dispositivos móveis por indivíduos com deficiência visual.

Esse trabalho discutiu como é imprescindível que a acessibilidade seja considerada desde o início do desenvolvimento de um aplicativo. Porém, muitos desenvolvedores relatam sentir-se desestimulados a adotar práticas inclusivas, seja pelo pouco conhecimento que têm no assunto, pela quantidade reduzida de recursos para realizar esse tipo de projeto ou pela complexidade relativa às ferramentas já existentes. Nesse contexto, surgiu a idealização de criar um pacote de componentes de UI acessíveis proposto por [Silva 2023], chamado de Code4All.

O pacote Code4All foi projetado para auxiliar desenvolvedores na criação de aplicativos acessíveis para pessoas com deficiência visual, de maneira que o uso dos componentes fornecidos fosse simples e eficiente. Dada a relevância desta inciativa, o presente trabalho teve como objetivo revisitar o Code4All e propor uma nova versão mais completa e aprimorada dele.

Inicialmente, introduziu-se uma nova nomenclatura, renomeando o *package* para CodeForAll e substituindo o sufixo "4All"nos componentes pelo prefixo "FA". Em seguida, realizou-se uma análise aprofundada dos elementos existentes, sugerindo algumas melhorias em conformidade com as diretrizes de acessibilidade mencionadas ao longo do projeto. Além do refinamento nos componentes já disponíveis, foram introduzidos 4 novos elementos para ampliar as possibilidades oferecidas pelo CodeForAll, foram eles: FADescriptiveImage, FADecorativeImage, FAImageButton e FALoadingView. Por fim,

propôs-se uma nova organização do projeto, reestruturando os arquivos e pastas do CodeForAll.

Para avaliar o *package* desenvolvido, foi formulado um questionário digital, que foi respondido por 11 profissionais da área. Os participantse da pesquisa foram desafiados a implementar cada um dos componentes do CodeForAll e, após a atividade prática, responderam uma série de perguntas sobre suas percepções do pacote.

Os resultados da pesquisa realizada demonstraram uma recepção extremamente positiva em relação ao CodeForAll. Os participantes destacaram que essa ferramenta seria de grande valor para ajudá-los a desenvolver aplicativos acessíveis para pessoas com deficiência visual. Além disso, a pontuação alta obtida na escala de usabilidade SUS (86,36) reforça a eficácia do CodeForAll como uma ferramenta viável para promover a acessibilidade em aplicativos móveis.

A pesquisa também identificou sugestões valiosas e possíveis pontos de melhoria para o CodeForAll. Questões como a necessidade de uma documentação aprofundada, aprimoramentos na inicialização dos componentes e a implementação de suporte ao *Dark Mode* foram levantadas pelos participantes. Essas sugestões fornecem *insights* valiosos para orientar futuros projetos e garantir que o CodeForAll continue a evoluir como uma ferramenta eficaz para promover a acessibilidade digital. Como limitação da avaliação realizada, é interessante citar o possível viés de resposta, resultante do contato profissional prévio entre a autora e os participantes da pesquisa.

Ademais, para garantir a consistência e a qualidade dos componentes propostos por CodeForAll, também é fundamental considerar, como melhoria futura, a implementação de testes automáticos. Esse tipo de validação ajudaria a identificar rapidamente possíveis falhas e a garantir que os componentes atendam aos padrões de acessibilidade estabelecidos. Assim, seria garantido que o *software* funcione conforme o esperado em diferentes cenários e ambientes, melhorando sua qualidade, confiabilidade e usabilidade. Portanto, a integração de testes automáticos no ciclo de desenvolvimento do CodeForAll pode ser uma etapa importante para aprimorar ainda mais a qualidade do *package*, garantindo que auxilie desenvolvedores a atenderem às necessidades de um público diverso.

Por fim, espera-se que esse trabalho auxilie, simplifique e impulsione o trabalho dos programadores ao adicionar funcionalidades acessíveis no desenvolvimento de aplicativos. Ao adotar componentes inclusivos como os de CodeForAll em seus projetos, engenheiros de *software* proporcionariam uma experiência de usuário eficiente para os mais diversos públicos. Compreender os desafios enfrentados pelos indivíduos com deficiência visual e tentar solucioná-los deve ser um objetivo contínuo da sociedade, para que todos tenham acesso às mesmas oportunidades.

Referências

[Abuaddous et al. 2016] Abuaddous, H., Jali, M. Z., and Basir, N. (2016). Web accessibility challenges. *International Journal of Advanced Computer Science and Applications*, 7(10):172–181.

[Android Developers] Android Developers. Accessibility Developer Checklist | Android Developers. `https://stuff.mit.edu/afs/sipb/project/android/`

docs/guide/topics/ui/accessibility/checklist.html. Acesso em: 6 de fevereiro de 2024.

[Apple Inc a] Apple Inc. About the VoiceOver rotor on iPhone, iPad, and iPod touch - Apple Support. https://support.apple.com/en-us/111796. Acesso em: 10 de fevereiro de 2024.

[Apple Inc b] Apple Inc. Accessibility | Apple Developer Documentation. https://developer.apple.com/design/human-interface-guidelines/accessibility. Acesso em: 6 de fevereiro de 2024.

[Apple Inc c] Apple Inc. Accessibility Inspector | Apple Developer Documentation. https://developer.apple.com/documentation/accessibility/accessibility-inspector. Acesso em: 10 de fevereiro de 2024.

[Apple Inc d] Apple Inc. Human Interface Guidelines | Apple Developer Documentation. https://developer.apple.com/design/human-interface-guidelines. Acesso em: 6 de fevereiro de 2024.

[Apple Inc e] Apple Inc. Swift Packages | Apple Developer Documentation. https://developer.apple.com/documentation/xcode/swift-packages. Acesso em: 5 de fevereiro de 2024.

[Apple Inc f] Apple Inc. SwiftUI Overview - Xcode - | Apple Developer Documentation. https:https://developer.apple.com/xcode/swiftui/. Acesso em: 5 de fevereiro de 2024.

[Bangor et al. 2009] Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual sus scores mean: adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123.

[Barbosa et al. 2021] Barbosa, S. D. J., da Silva, B. S., Silveira, M. S., Gasparini, I., Darin, T., and Barbosa, G. D. J. (2021). *Interaçãoo Humano-Computador e Experiência do Usuário*
. Autopublicação.

[Brasil 2000] Brasil (2000). Cadernos da tv escola - deficiência visual. http://portal.mec.gov.br/seed/arquivos/pdf/deficienciavisual.pdf. Ministério da Educação - Scretaria de Educação à distância. Acesso em: 5 de fevereiro de 2024".

[Brooke 1995] Brooke, J. (1995). Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189.

[Carvalho et al. 2016] Carvalho, L. P., Peruzza, B. P. M., Santos, F., Ferreira, L. P., and Freire, A. P. (2016). Accessible smart cities?: Inspecting the accessibility of brazilian municipalities' mobile applications. In *Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems*, New York, NY, USA. ACM.

[Ceccarini and Prandi 2019] Ceccarini, C. and Prandi, C. (2019). Tourism for all: a mobile application to assist visually impaired users in enjoying tourist services. In *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6.

[Colorado State University] Colorado State University. Headings. https://www.chhs.colostate.edu/accessibility/best-practices-how-tos/headings/. Acesso em: 10 de fevereiro de 2024.

[Darvishy et al. 2020] Darvishy, A., Hutter, H.-P., and Frei, J. (2020). Making mobile map applications accessible for visually impaired people. In Ahram, T., Taiar, R., Colson, S., and Choplin, A., editors, *Human Interaction and Emerging Technologies*, pages 396–400, Cham. Springer International Publishing.

[de Almeida and Gama 2021] de Almeida, V. L. and Gama, K. (2021). Mobile accessibility guidelines adoption under the perspective of developers and designers. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 127–128.

[de Oliveira and Filgueiras 2018] de Oliveira, A. F. B. A. and Filgueiras, L. V. L. (2018). Developer assistance tools for creating native mobile applications accessible to visually impaired people: A systematic review. IHC '18, New York, NY, USA. Association for Computing Machinery.

[Departamento de Governo Eletrônico 2014] Departamento de Governo Eletrônico (2014). eMAG - Modelo de Acessibilidade em Governo Eletrônico. https://emag.governoeletronico.gov.br. Acesso em: 6 de fevereiro de 2024.

[Di Gregorio et al. 2022] Di Gregorio, M., Di Nucci, D., Palomba, F., and Vitiello, G. (2022). The making of accessible android applications: an empirical study on the state of the practice. *Empir. Softw. Eng.*, 27(6):145.

[Ferreira et al. 2012] Ferreira, S. B. L., Nunes, R. R., and da Silveira, D. S. (2012). Aligning usability requirements with the accessibility guidelines focusing on the visually-impaired. *Procedia Computer Science*, 14:263–273. Proceedings of the 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2012).

[Gamma et al. 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 edition.

[Ghidini et al. 2016] Ghidini, E., Almeida, W. D. L., Manssour, I. H., and Silveira, M. S. (2016). Developing apps for visually impaired people: Lessons learned from practice. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 5691–5700.

[Huq et al. 2023] Huq, S. F., Alshayban, A., He, Z., and Malek, S. (2023). #A11yDev: Understanding contemporary software accessibility practices from twitter conversations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA. ACM.

[Khan and Khusro 2021] Khan, A. and Khusro, S. (2021). An insight into smartphone-based assistive solutions for visually impaired and blind people: issues, challenges and opportunities. *Univers. Access Inf. Soc.*, 20(2):265–298.

[Kim et al. 2016] Kim, H. K., Han, S. H., Park, J., and Park, J. (2016). The interaction experiences of visually impaired people with assistive technology: A case study of smartphones. *International Journal of Industrial Ergonomics*, 55:22–33.

[Likert 1932] Likert, R. (1985 - 1932). *A technique for the measurement of attitudes / by Rensis Likert.* Archives of psychology ; no. 140. [s.n.], New York.

[Matausch et al. 2012] Matausch, K., Peböck, B., and Pühretmair, F. (2012). Accessible content generation an integral part of accessible web design. *Procedia Computer Science*, 14:274–282. Proceedings of the 4th International Conference on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2012).

[Mayordomo-Martínez et al. 2019] Mayordomo-Martínez, D., Sánchez-Aarnoutse, J.-C., Carrillo-de Gea, J. M., García-Berná, J. A., Fernández-Alemán, J. L., and García-Mateos, G. (2019). Design and development of a mobile app for accessible beach tourism information for people with disabilities. *Int. J. Environ. Res. Public Health*, 16(12):2131.

[Milne et al. 2014] Milne, L. R., Bennett, C. L., and Ladner, R. E. (2014). The accessibility of mobile health sensors for blind users. *Journal on Technology and Persons with Disabilities*.

[Organização das Nações Unidas 2006] Organização das Nações Unidas (2006). Convenção sobre os direitos das pessoas com deficiência.

[Organização Pan-Americana da Saúde] Organização Pan-Americana da Saúde. Deficiência. `https://www.paho.org/pt/topicos/deficiencia`. Acesso em: 2 de fevereiro de 2024.

[Ross et al. 2018] Ross, A. S., Zhang, X., Fogarty, J., and Wobbrock, J. O. (2018). Examining image-based button labeling for accessibility in android apps through large-scale analysis. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, New York, NY, USA. ACM.

[Ross et al. 2020] Ross, A. S., Zhang, X., Fogarty, J., and Wobbrock, J. O. (2020). An epidemiology-inspired large-scale analysis of android app accessibility. *ACM Trans. Access. Comput.*, 13(1):1–36.

[Sanchez-Gordon and Luján-Mora 2013] Sanchez-Gordon, S. and Luján-Mora, S. (2013). Web accessibility of moocs for elderly students. In *2013 12th International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–6.

[Secretaria de Defesa Social de Pernambuco] Secretaria de Defesa Social de Pernambuco. Acessibilidade. `https://www.sds.pe.gov.br/acessibilidade/17-acessibilidade/9-acessibilidade`. Acesso em: 2 de fevereiro de 2024.

[Siebra et al. 2015] Siebra, C., Gouveia, T., Macedo, J., Correia, W., Penha, M., Silva, F., Santos, A., Anjos, M., and Florentin, F. (2015). Usability requirements for mobile accessibility. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, New York, NY, USA. ACM.

[Silva 2023] Silva, H. d. S. (2023). Componentes modulares para criação de aplicações acessíveis em SwiftUI. `https://repositorio.ufpe.br/handle/123456789/52715`. Trabalho de graduação, Universidade Federal de Pernambuco.

[University of California San Francisco] University of California San Francisco. Accessible Images Best Practices | UCSF IT. https://it.ucsf.edu/how-to/accessible-images-best-practices. Acesso em: 10 de fevereiro de 2024.

[U.S. General Services Administration 2017] U.S. General Services Administration (2017). Section 508. https://www.section508.gov. Acesso em: 6 de fevereiro de 2024.

[W3C 2008] W3C (2008). Mobile Web Best Practices 1.0. https://www.w3.org/TR/mobile-bp/. Acesso em: 6 de fevereiro de 2024.

[W3C 2009] W3C (2009). Relationship between Mobile Web Best Practices (MWBP) and Web Content Accessibility Guidelines (WCAG). https://www.w3.org/TR/2009/NOTE-mwbp-wcag-20090709/. Acesso em: 6 de fevereiro de 2024.

[W3C 2018] W3C (2018). Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 2.1. https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/. Acesso em: 6 de fevereiro de 2024.

[W3C Web Accessibility Initiative 2006] W3C Web Accessibility Initiative (2006). WAI-ARIA Overview. https://www.w3.org/WAI/standards-guidelines/aria/. Acesso em: 6 de fevereiro de 2024.

[W3C Web Accessibility Initiative 2022] W3C Web Accessibility Initiative (2022). Images Tutorial. https://www.w3.org/WAI/tutorials/images/. Acesso em: 10 de fevereiro de 2024.

[WebAIM 2017] WebAIM (2017). Screen reader user survey 7 results. https://webaim.org/projects/screenreadersurvey7/. Acesso em: 5 de fevereiro de 2024.

[World Health Organization] World Health Organization. Disability. https://www.who.int/news-room/fact-sheets/detail/disability-and-health#:~:text=Key%20facts,earlier%20than%20those%20without%20disabilities. Acesso em: 2 de fevereiro de 2024.

[World Health Organization 2023] World Health Organization (2023). Blindness and vision impairment. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment#:~:text=Globally%2C%20at%20least%202.2%20billion,is%20yet%20to%20be%20addressed. Acesso em: 2 de fevereiro de 2024.

[Yan and Ramachandran 2019] Yan, S. and Ramachandran, P. G. (2019). The current status of accessibility in mobile apps. *ACM Trans. Access. Comput.*, 12(1).