# Code For All:
# Componentes modulares para criação de aplicações acessíveis em SwiftUI

**Hugo dos Santos Silva**[1], **Kiev Gama**[1*]

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

hss@cin.ufpe.br, kiev@cin.ufpe.br

***Abstract.*** *With the growth in the use of mobile devices and, consequently, the use of apps, it has become important to carefully consider how accessible they are for users with disabilities. Among the reasons why developers do not prioritize accessibility in their applications are lack of awareness, time and resource constraints, and technical complexity. In this way, CodeForAll is proposed, a package with a set of extensions to facilitate the creation of inclusive platforms using SwiftUI.*

***Resumo.*** *Com o crescimento da utilização de dispositivos móveis e, consequentemente, o uso de aplicativos, tornou-se importante considerar cuidadosamente o quão acessíveis eles são para usuários com deficiência. Dentre as razões pelas quais as pessoas desenvolvedoras não priorizam a acessibilidade em seus aplicativos, encontra-se a falta de conscientização, restrições de tempo e recursos, e complexidade técnica. Dessa forma, é proposto CodeForAll, um package com um conjunto de componentes modularizados para facilitar a criação de plataformas inclusivas utilizando SwiftUI.*

[*]orientador

## 1. Introdução

Os aplicativos móveis desempenham um papel essencial em nossa vida cotidiana, sendo cruciais para aproximadamente 15% da população global que vive com alguma forma de deficiência, de acordo com a World Health Organization (WHO)[Organization et al. 2011]. No entanto, muitas vezes, os desenvolvedores negligenciam a acessibilidade ao criar essas aplicações, o que resulta em uma exclusão significativa dessas pessoas[Zhang et al. 2021]. Com o aumento do número de dispositivos móveis em uso, as aplicações móveis se tornaram uma parte fundamental da interação das pessoas com serviços essenciais e o mundo digital, ressaltando a importância crítica de tornar a acessibilidade uma prioridade central durante o processo de desenvolvimento de aplicativos móveis.

A importância de ajudar desenvolvedores a escreverem códigos para criar aplicações mais acessíveis é um tópico crucial e relevante, especialmente no contexto do aumento contínuo das aplicações móveis. A acessibilidade digital é uma questão fundamental de direitos humanos, igualdade e inclusão social. Garantir que as aplicações móveis sejam acessíveis a todos os usuários, independentemente de suas capacidades físicas, cognitivas ou sensoriais, é essencial para promover uma sociedade mais inclusiva e equitativa.

Embora tenha havido um aumento significativo no número de diretrizes, padrões e recomendações para promover a acessibilidade em aplicativos móveis, ainda persistem desafios na garantia efetiva dessa acessibilidade. Por mais que essas diretrizes tenham desempenhado um papel importante na conscientização dos desenvolvedores sobre a importância do design inclusivo, obstáculos no processo de desenvolvimento muitas vezes desencorajam os desenvolvedores de efetivamente implementarem acessibilidade em suas aplicações móveis[de Oliveira and Filgueiras 2018].

Desenvolver componentes acessíveis manualmente em *SwiftUI* pode ser mais demorado e exigir mais esforço dos desenvolvedores devido à necessidade de implementar marcações e testes de acessibilidade, bem como o conhecimento necessário em acessibilidade. Além disso, a escalabilidade pode ser um desafio à medida que a complexidade dos componentes aumenta.

Felizmente, ajudar desenvolvedores a escreverem códigos acessíveis não apenas melhora a experiência do usuário para pessoas com deficiência, mas também oferece benefícios significativos a toda a base de usuários [Schmutz et al. 2016]. Um estudo realizado pela Accenture, em parceria com a Disability:IN e o American Association of People with Disabilities (AAPD), mostrou que empresas que priorizam a acessibilidade em suas aplicações móveis têm um crescimento médio de receita mais elevado em comparação com aquelas que não o fazem [Accenture 2018]. Isso se deve ao fato de que, ao tornar as aplicações mais acessíveis, elas se tornam mais fáceis de usar para todos os usuários, resultando em maior engajamento e lealdade à marca.

Além disso, investir na acessibilidade também contribui para a conformidade com regulamentações e padrões internacionais. Por exemplo, em muitos países, leis como a Americans with Disabilities Act (ADA)[ada 1990] nos Estados Unidos e o Web Content Accessibility Guidelines (WCAG) estabelecem diretrizes para garantir a acessibilidade em serviços digitais, incluindo aplicações móveis[w3C 2018]. O não cumprimento dessas

regulamentações pode levar a processos legais e danos à reputação da empresa, como os previstos na Lei Brasileira de Inclusão da Pessoa com Deficiência, 13.146, de 6 de julho de 2015[Civil 2015].

Outro aspecto importante é que tornar as aplicações mais acessíveis não é uma tarefa extremamente complexa. Muitas vezes, pequenas mudanças no código ou no design podem fazer uma grande diferença. Por exemplo, fornecer alternativas textuais para botões, garantir que os elementos da interface sejam identificáveis pelos leitores de tela e usar cores e contrastes adequados são algumas das práticas recomendadas.

Ao fomentar a criação de aplicações móveis mais acessíveis, promove-se uma cultura de inclusão e respeito à diversidade. Cada pessoa deve ter a oportunidade de utilizar as tecnologias móveis de forma independente e autônoma, e é responsabilidade de desenvolvedores e empresas garantir que esse objetivo seja alcançado.

É indiscutível que auxiliar os desenvolvedores na criação de aplicativos móveis mais acessíveis é uma necessidade crucial, não apenas por razões éticas e legais, mas também por motivos econômicos e sociais. A acessibilidade digital é uma jornada contínua e, à medida que a tecnologia avança, é essencial que se continue investindo em soluções inclusivas para garantir que todos possam desfrutar dos benefícios da revolução móvel. Nesse contexto, o propósito deste trabalho é apresentar um conjunto de componentes em *SwiftUI* que atuem como um suporte para os desenvolvedores na criação de aplicativos *iOS* nativos, com foco na acessibilidade para usuários com deficiência visual.

## 2. Definições técnicas

Esta seção tem como objetivo apresentar conceitos importantes para compreensão e desenvolvimento deste trabalho, os principais conjuntos de diretrizes de acessibilidade para as necessidades básicas que uma aplicação mobile deve seguir, as ferramentas que auxiliam ao uso de softwares mobile e na criação de softwares acessíveis e métodos de testagem de acessibilidade.

### 2.1. Deficiência visual

A deficiência visual é um dos principais desafios enfrentados por uma parcela significativa da população mundial [mec 2018]. Caracterizada pela perda parcial ou total da capacidade de enxergar, essa condição tem implicações profundas nas esferas física, emocional e social dos indivíduos afetados. A compreensão dos aspectos teóricos relacionados à deficiência visual é fundamental para desenvolver políticas, práticas e intervenções que garantam a inclusão e a qualidade de vida dessas pessoas.

### 2.2. Acessibilidade Mobile

Nos dias atuais, os smartphones desempenham um papel central em nossas vidas, oferecendo uma ampla gama de funcionalidades que abrangem desde a comunicação até o entretenimento e a produtividade. No entanto, para as pessoas com baixa visão e cegueira, a utilização desses dispositivos pode apresentar desafios significativos. Graças ao reconhecimento desta necessidade, tanto o sistema operacional *Android* quanto o *iOS* da Apple reconhecem essa necessidade e vêm incorporando ferramentas de acessibilidade que visam melhorar a experiência de uso para esses usuários como o *TalkBack*[goo 2009] e o *VoiceOver*[app 2009b], que são recursos de acessibilidade essenciais em dispositivos

móveis, projetados para melhorar a experiência de uso de pessoas com deficiência visual. Porém, é crucial destacar que a eficácia total dessas ferramentas nos aplicativos está condicionada à forma como o aplicativo foi desenvolvido para integrar e dar suporte a essas funcionalidades de acessibilidade.

- **VoiceOver**: O *VoiceOver* é um recurso de acessibilidade desenvolvido pela Apple para pessoas com deficiência visual. Ele é projetado para auxiliar os usuários a interagir com dispositivos da Apple, como iPhones, iPads, Macs, Apple Watches e Apple TVs, fornecendo informações auditivas sobre elementos de interface e conteúdo na tela.
  O *VoiceOver* transforma informações visuais em informações auditivas, permitindo que os usuários naveguem, interajam e usem seus dispositivos por meio de comandos de voz e gestos específicos. Quando ativado, o *VoiceOver* lê em voz alta os elementos na tela, como botões, ícones, texto, links e muito mais. O VoiceOver também fornece feedback sobre ações, como toques e gestos, tornando possível o uso independente de dispositivos por pessoas com deficiência visual [app 2009a].
- ***Diretrizes de acessibilidade***: Compreender as necessidades dos usuários com deficiência visual é um elemento fundamental em qualquer empreendimento de desenvolvimento de software que aspire atingir um padrão de acessibilidade de alta qualidade para todos os públicos. Atualmente existem diversos estudos que servem como orientações essenciais para dirigir o processo de criação de projetos de software. Para este estudo, a base foi construída seguindo as diretrizes WCAG e Accessible Rich Internet Applications (ARIA).
- ***WCAG***: A WCAG são diretrizes internacionais desenvolvidas pelo World Wide Web Consortium (W3C) para tornar o conteúdo da web mais acessível a pessoas com deficiências [w3c 2018], sendo importante ressaltar que em 2018, o W3C lançou uma atualização, a WCAG 2.1, que introduziu diretrizes e critérios de sucesso adicionais que se concentram especificamente em melhorar a acessibilidade em dispositivos móveis. O objetivo principal das WCAG é garantir que os sites e aplicativos sejam projetados e desenvolvidos de forma a proporcionar uma experiência igualitária para todos os usuários, independentemente de suas habilidades ou deficiências.
  As WCAG definem critérios específicos que os desenvolvedores devem atender para garantir a acessibilidade do conteúdo da web. Esses critérios são organizados em quatro princípios principais, conhecidos como P.O.U.R:
  - **Perceptível (*Perceivable*):** O conteúdo da web deve ser apresentado de maneira que os usuários possam percebê-lo, independentemente de suas limitações sensoriais. Isso envolve garantir que todas as informações importantes sejam apresentadas de maneira visual, auditiva ou tátil.
  - **Operável (*Operable*):** Os usuários devem ser capazes de interagir e navegar pelo conteúdo da web de maneira eficaz, mesmo se estiverem usando tecnologias assistivas. Isso inclui fornecer uma navegação clara, evitando armadilhas de teclado e permitindo que os usuários tenham tempo suficiente para ler e entender o conteúdo.
  - **Compreensível (*Understandable*):** O conteúdo e a navegação devem ser fáceis de entender, independentemente do nível de conhecimento ou ha-

bilidade do usuário. Isso envolve o uso de linguagem clara, organização lógica e instruções compreensíveis.
  - **Robusto (*Robust*):** O conteúdo deve ser desenvolvido de maneira que possa ser interpretado por uma ampla variedade de tecnologias assistivas e navegadores, garantindo que ele permaneça acessível no futuro.

Cada um desses princípios é acompanhado por diretrizes específicas e critérios de sucesso que os desenvolvedores devem atender para alcançar a conformidade com as WCAG. As diretrizes estão organizadas em três níveis de conformidade: A (o mais básico), AA (intermediário) e AAA (mais avançado).

As WCAG desempenham um papel crítico na promoção da inclusão digital, garantindo que a web seja acessível a todos, independentemente de suas capacidades. Organizações e desenvolvedores são incentivados a adotar essas diretrizes para criar uma experiência online mais igualitária e acessível.

É crucial destacar que, mesmo que um projeto alcance uma avaliação positiva nos critérios do WCAG, isso não assegura automaticamente que ele proporcionará uma experiência excepcional para todos os usuários. É de se esperar que, no mínimo, o projeto atenda às necessidades fundamentais para acessibilidade em projetos mobile, para tornar as coisas mais acessíveis para pessoas com diferentes limitações.

No contexto de acessibilidade digital, é relevante mencionar WCAG como uma referência vital. O WCAG oferece um conjunto estabelecido de diretrizes para garantir a acessibilidade de conteúdo na web[w3c 2018], e suas orientações podem ser devidamente aplicadas ao desenvolvimento de componentes acessíveis através do Swift Package Manager (SPM). Ao abraçar o WCAG, os princípios de semântica de conteúdo, contraste de cores, navegação coerente e apoio a tecnologias assistivas podem ser integrados ao processo de criação de componentes, assegurando que eles não apenas cumpram padrões de acessibilidade, mas também ofereçam uma experiência inclusiva e funcional para um amplo espectro de usuários, incluindo aqueles com deficiências. Isso ressalta a importância de alinhar os princípios amplamente reconhecidos do WCAG com a criação de componentes através do SPM, resultando em soluções mais inclusivas e robustas.

- ***ARIA***: ARIA é um conjunto de atributos e papéis que são adicionados a elementos HyperText Markup Language (HTML) para melhorar a acessibilidade de aplicativos da web interativos e ricos em conteúdo. ARIA foi desenvolvido para complementar as capacidades das tecnologias de acessibilidade existentes, como leitores de tela, permitindo que desenvolvedores tornem o conteúdo da web mais acessível a pessoas com deficiências[(WAI) 2022]. O ARIA ajuda a atender às recomendações do WCAG ao melhorar a acessibilidade em aplicativos da web de várias maneiras, incluindo aprimoramento da semântica dos elementos, melhorias na navegação, fornecimento de estados e propriedades, suporte a interações complexas, acessibilidade em aplicativos dinâmicos e identificação de erros em formulários. O ARIA é uma tecnologia voltada principalmente para melhorar a acessibilidade em aplicativos da web, não tendo um equivalente específico para aplicativos móveis. No entanto, as boas práticas de acessibilidade podem e devem ser aplicadas em aplicativos móveis, seguindo as diretrizes de acessibilidade das plataformas móveis e adotando princípios gerais de acessibilidade.

  As principais finalidades do ARIA incluem:

- Adicionar semântica acessível: Às vezes, elementos HTML padrão não conseguem transmitir informações suficientes sobre a funcionalidade ou o propósito de elementos interativos. ARIA fornece atributos adicionais que ajudam a descrever adequadamente elementos como botões, menus, caixas de diálogo e mais, permitindo que leitores de tela e outras tecnologias assistivas forneçam informações relevantes aos usuários.
- Melhorar a interatividade: ARIA permite que elementos interativos, como *widgets* personalizados, sejam percebidos corretamente por tecnologias assistivas. Isso ajuda os usuários a entenderem como interagir com esses elementos e a receber *feedback* apropriado.
- Gerenciar conteúdo dinâmico: Com a proliferação de conteúdo dinâmico e aplicações da web de página única (SPA), ARIA ajuda a garantir que as mudanças no conteúdo sejam comunicadas corretamente aos usuários de tecnologias assistivas, sem a necessidade de recarregar a página.
- Melhorar a navegabilidade: ARIA pode ser usado para criar *landmarks* (marcos de navegação) que ajudam os usuários a navegar em uma página da web com mais eficiência.
- Descrever estados e propriedades: Elementos ARIA podem descrever o estado atual de um elemento interativo, como um botão que está ativado ou desativado, ou uma barra de progresso que mostra o progresso atual.
- Personalização da experiência: ARIA permite que os desenvolvedores personalizem como as informações são comunicadas aos usuários de tecnologias assistivas, para atender às necessidades individuais.

É importante ressaltar que, embora o ARIA seja uma ferramenta poderosa para aprimorar a acessibilidade na web, seu uso deve ser criterioso. O ARIA é principalmente destinado a melhorar a acessibilidade em aplicativos da web e não possui um equivalente específico para aplicativos móveis. No entanto, as boas práticas de acessibilidade podem e devem ser aplicadas em aplicativos móveis, seguindo as diretrizes de acessibilidade das plataformas móveis e adotando princípios gerais de acessibilidade para garantir uma experiência acessível para todos os usuários. Portanto, os princípios de design acessível devem ser a base, e o ARIA deve ser usado quando os elementos HTML padrão não são suficientes para transmitir informações e funcionalidades adequadamente. O ARIA não substitui a necessidade de seguir as diretrizes de acessibilidade, como as WCAG, mas sim complementa ao ajudar a atender às recomendações do WCAG de forma a melhorar a acessibilidade em aplicativos da web de várias maneiras, incluindo aprimoramento da semântica dos elementos, melhorias na navegação, fornecimento de estados e propriedades, suporte a interações complexas e acessibilidade em aplicativos dinâmicos.

## 2.3. Desenvolvimento para iOS

### 2.3.1. Xcode

O *Xcode* é um ambiente de desenvolvimento da Apple para criar aplicativos em sistemas operacionais como *iOS* e *macOS*. O *Xcode* oferece ferramentas para escrever código, projetar interfaces de usuário, depurar, testar e lançar aplicativos. Também inclui simu-

ladores, gerenciamento de versão e recursos para otimização de desempenho, tornando-se uma ferramenta essencial para desenvolvedores de aplicativos da Apple[app e].

### 2.3.2. XCTest

O *XCTest* é um *framework* de teste automatizado integrado ao *Xcode* da Apple, essencial para desenvolvedores de aplicativos *iOS*, *macOS*, *watchOS* e *tvOS*. O *XCTest* oferece suporte para testes de unidade e testes de interface do usuário, com uma sintaxe clara e uma *API* rica para escrever, executar e depurar testes diretamente no ambiente de desenvolvimento[app f].

### 2.3.3. Accessibility Inspector

O *Accessibility Inspector* é uma ferramenta integrada no *Xcode*, desenvolvida pela Apple, para testar e melhorar a acessibilidade de interfaces de usuário em aplicativos para diferentes plataformas da empresa. Ele oferece informações em tempo real sobre a estrutura e acessibilidade dos elementos na tela, detecta problemas de acessibilidade, permite simular deficiências e oferece detalhes sobre elementos de interface. É uma ferramenta essencial para garantir que os aplicativos sejam usáveis por pessoas com deficiências, promovendo a inclusão e a qualidade da experiência do usuário[app a].

### 2.3.4. SwiftUI

O *SwiftUI* é um *framework* da Apple para criar interfaces de usuário em aplicativos *iOS*, *macOS*, *watchOS* e *tvOS*. Ele usa uma abordagem declarativa e reativa, permitindo que os desenvolvedores descrevam a interface usando código. Isso simplifica o desenvolvimento, pois as interfaces se adaptam automaticamente às mudanças nos dados. O *SwiftUI* é multiplataforma, integra-se bem com a linguagem Swift e oferece pré-visualização ao vivo das alterações na interface durante o desenvolvimento [app d].

### 2.3.5. Swift Package Manager

O SPM é uma ferramenta da Apple para gerenciar dependências e distribuir código *Swift*. Integrada ao sistema de compilação *Swift*, o SPM automatiza o processo de obtenção, compilação e vinculação de dependências externas. Isso simplifica o gerenciamento de bibliotecas, promove a coerência entre versões e incentiva boas práticas de desenvolvimento. O SPM também melhora a eficiência, a organização e a confiabilidade no desenvolvimento *Swift* [app b].

## 3. Trabalhos Relacionados

Dentro do contexto da acessibilidade em projetos de software, são conduzidos estudos cruciais para compreender as complexidades envolvidas e para aprimorar a experiência do usuário considerando as limitações individuais. As diretrizes acessíveis são exemplificadas pelo próprio WCAG, que é uma referência fundamental nesse estudo. Além disso,

há uma ênfase na criação de ferramentas que não apenas facilitem, mas também motivem os desenvolvedores a construir aplicações mais acessíveis. Neste trabalho, propõe-se uma ferramenta de apoio ao desenvolvimento de novas aplicações de forma modularizada e acessível.

### 3.1. Bibliotecas focadas em acessibilidade

*Chakra UI* [Chakra ] é uma biblioteca proeminente de componentes para construção de aplicativos *React*, destacando-se por sua abordagem simples, modular e acessível. Ela oferece uma variedade abrangente de blocos de construção prontos para uso, permitindo que os desenvolvedores criem interfaces de usuário coerentes e visualmente atrativas com facilidade. O que a torna verdadeiramente notável é o foco dedicado à acessibilidade, integrando práticas de design inclusivo e padrões de acessibilidade em sua estrutura. Isso resulta em componentes não apenas esteticamente agradáveis, mas também funcionalmente utilizáveis por uma ampla gama de usuários, inclusive aqueles com deficiências. A *Chakra UI* adere a princípios como semântica HTML precisa, contraste de cores adequado, navegação intuitiva e suporte a tecnologias assistivas, assegurando que as interfaces criadas sejam acessíveis desde o início. Sua abordagem modular flexível e documentação abrangente facilitam a adoção, agilizando o processo de desenvolvimento e contribuindo para a criação de experiências de usuário inclusivas e de alta qualidade.

O *UIKit*[app 2007] é uma biblioteca essencial no desenvolvimento de aplicativos nativos para *iOS*, proporcionando uma base sólida para a criação de interfaces de usuário acessíveis. O *UIKit* destaca-se por seu suporte à acessibilidade, permitindo que os desenvolvedores tornem seus aplicativos inclusivos por meio da definição de etiquetas descritivas e atributos acessíveis. Além disso, o *UIKit* oferece flexibilidade ao permitir a criação de componentes personalizados estendendo as classes existentes, como *UIView* e *UIControl*, facilitando a adaptação às necessidades específicas do aplicativo. Além disso, o *framework* oferece suporte a recursos avançados, como animações e gestos, permitindo a criação de experiências de usuário interativas em dispositivos *iOS*, contribuindo para a qualidade e eficácia dos aplicativos nativos da Apple.

### 3.2. Artigos focados em acessibilidade

Um estudo focado na criação de aplicativos de transporte acessíveis[Krainz et al. 2016] trata da criação de aplicativos de transporte acessíveis para pessoas com deficiência visual, usando uma abordagem de Desenvolvimento orientado por modelos (Model Driven Development -MDD). Este artigo aborda a interseção de MDD, desenvolvimento de aplicativos móveis e acessibilidade, reconhecendo a falta de foco na acessibilidade em abordagens anteriores de MDD e no desenvolvimento de aplicativos móveis. O trabalho também menciona iniciativas relacionadas à acessibilidade e assistência a pessoas com deficiência visual em dispositivos móveis. O objetivo é criar um modelo que permita desenvolver aplicativos de transporte acessíveis desde o início do processo de desenvolvimento, melhorando a experiência para pessoas com deficiência visual.

Um estudo voltado para o modelo de avaliação de usabilidade de aplicativos móveis para deficientes visuais[Hussain and Omar 2020] concentra-se na avaliação da usabilidade de aplicativos móveis desenvolvidos para pessoas com deficiência visual. Por meio de uma revisão sistemática da literatura, o estudo identifica as diversas dimensões de usabilidade que são cruciais para ajudar desenvolvedores e avaliadores de aplicativos

a avaliar adequadamente essas aplicações, particularmente aquelas voltadas para usuários com deficiência visual moderada e grave. A pesquisa ressalta a importância de tornar essas tecnologias acessíveis e amigáveis para esse grupo de usuários, abordando métodos de teste, apresentando resultados relevantes e oferecendo recomendações práticas para melhorar a qualidade e a acessibilidade desses aplicativos. Com isso, o trabalho contribui substancialmente para o campo da acessibilidade digital, fornecendo diretrizes práticas para o desenvolvimento de aplicativos móveis verdadeiramente inclusivos e atentos às necessidades das pessoas com deficiência visual.

A solução proposta nessa pesquisa reúne alguns dos pontos associados à acessibilidade mobile desenvolvendo componentes modulares para criação de aplicações móveis de maneira ágil e acessível.

## 4. Code4All: componentes modulares para criação de aplicações acessíveis em *SwiftUI*.

Nesta seção, será descrita detalhadamente a abordagem metodológica utilizada para o desenvolvimento do pacote de componentes modulares voltados para a criação de aplicações acessíveis em *SwiftUI*. O objetivo desta pesquisa foi criar uma solução que permitisse aos desenvolvedores criar aplicativos *iOS* mais acessíveis de forma eficiente, facilitando a implementação de práticas de acessibilidade.

O desenvolvimento do pacote de componentes seguiu um processo linear que envolveu várias etapas bem definidas. A metodologia de desenvolvimento adotada foi baseada nas 7 etapas seguintes: análise e planejamento inicial, design dos componentes acessíveis, definição das ferramentas, desenvolvimento do pacote de componentes, testes de acessibilidade, Metodologia de avaliação. As seções serão detalhadas a seguir.

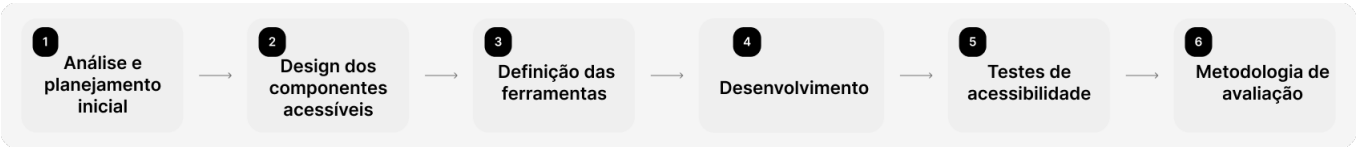


**Figure 1. Etapas da metodologia.**

### 4.1. Análise e planejamento inicial

Nesta fase, foi realizada uma revisão detalhada da documentação oficial do *SwiftUI* e das diretrizes de acessibilidade da Apple. Foram identificados os principais desafios relacionados à acessibilidade que os desenvolvedores enfrentam ao criar interfaces de usuário em aplicações mobile. Um estudo dedicado à avaliação do estado atual da acessibilidade em aplicativos móveis [Yan and Ramachandran 2019] investigou a acessibilidade de 479 aplicativos *Android* em 23 categorias de negócios na Google Play, analisando suas estruturas de interface gráfica e conformidade com diretrizes de acessibilidade. Utilizando a ferramenta IBM Mobile Accessibility Checker (MAC), o estudo identificou problemas de acessibilidade categorizados como violações, potenciais violações ou avisos. Os resultados indicaram que uma proporção significativa dos aplicativos apresentava problemas de acessibilidade, com cinco categorias de *widgets* responsáveis pela maioria dos elementos da Graphical User Interface (GUI) e pela maioria dos problemas detectados, dentre as

categorias de widget testadas estavam *TextView*, *ImageView*, *View*, *Button* e *ImageButton*. Esses problemas incluíam falta de foco em elementos, descrições ausentes, baixo contraste de cores de texto, espaçamento insuficiente entre elementos e tamanhos de fontes e elementos menores do que o recomendado. Essas questões predominantes contribuíram para a falta de acessibilidade dos aplicativos móveis avaliados. Os resultados do estudo que avaliou a acessibilidade de aplicativos *Android* podem ser aplicados aos aplicativos *iOS*, destacando a necessidade de abordar desafios semelhantes na plataforma *iOS*. Embora o foco tenha sido em aplicativos *Android*, as conclusões revelam problemas universais de acessibilidade, como falta de foco em elementos, descrições ausentes, contraste inadequado de cores de texto e espaçamento insuficiente, que também afetam a experiência do usuário em dispositivos *iOS*. Dessa forma, os objetivos do projeto e os critérios de sucesso foram definidos, que incluíam a conformidade com as diretrizes de acessibilidade da Apple, as WCAG e as práticas de ARIA, bem como a garantia da facilidade de uso por parte dos desenvolvedores.

### 4.2. Design de componentes acessíveis

A fase de design desempenha um papel crucial na criação de componentes acessíveis para aplicações em *SwiftUI*. Inicialmente implementou-se padrões de design de interação que favorecessem a usabilidade por parte de pessoas com deficiências visuais. Essa abordagem não apenas tornaria os componentes mais inclusivos, mas também enriqueceria a experiência do usuário para todos os públicos. Utilizou-se técnicas de design centrado no usuário para criar componentes modulares que atendessem às necessidades de acessibilidade, através da utilização rótulos descritivos significativos para todos os elementos interativos. Isso incluiu a associação de rótulos a elementos como botões e *textfields*, garantindo que os usuários com deficiências visuais recebessem informações claras sobre a função e o propósito de cada componente. Considerando que o público-alvo são desenvolvedores, a flexibilidade e personalização dos componentes foram consideradas para acomodar uma variedade de casos de uso.

### 4.3. Definição das ferramentas

Uma vez que o Design de componentes acessíveis que serão implementados está definido, a etapa seguinte é a definição das ferramentas para a construção, buscando simplificar o processo de desenvolvimento, testes, documentação:

- **SwiftUI 5.8**: Como mencionado anteriormente o *SwiftUI* é um *framework* desenvolvido pela Apple para a criação de interfaces de usuário em aplicativos móveis para iOS, macOS, watchOS e tvOS. Se destacando pela sua facilidade de uso e abordagem declarativa para a criação de interfaces [app d].
  O *SwiftUI* foi utilizado neste projeto devido à sua abordagem declarativa, que simplifica a criação de interfaces, aumenta a produtividade, promove a reutilização de código, oferece suporte multiplataforma, estimula a reutilização de componentes de interface de usuário, permitindo que os desenvolvedores criem componentes personalizados e os compartilhem entre diferentes partes de seus aplicativos. Isso economiza tempo e esforço na criação de interfaces consistentes em todo o aplicativo.
- **Swift Package Manager (SPM)**: A modularização em *SwiftUI* refere-se à prática de dividir o código do aplicativo em módulos independentes, cada um com uma

responsabilidade e funcionalidade específicas. Isso melhora a organização do código, facilitando a localização e a manutenção de partes específicas do aplicativo. Além disso, a modularização promove a reutilização de código ao permitir que componentes sejam compartilhados em diferentes partes do aplicativo ou até mesmo entre projetos, facilita a testabilidade, pois módulos independentes podem ser testados individualmente, contribuindo para a detecção e correção de erros, isola as dependências entre os diferentes módulos, tornando o código mais robusto e escalável, o que permite adicionar ou remover funcionalidades sem afetar outras partes do código. Neste projeto, a modularização em *SwiftUI* foi uma prática pensada para facilitar a integração dos componentes criados à aplicativos terceiros, uma vez que isola funcionalidades, define interfaces claras, melhora a testabilidade, oferece documentação focada, promove a reutilização de código e facilita a integração por meio de ferramentas de gerenciamento de dependências.

- ***GitHub***: O *GitHub* é uma plataforma líder de hospedagem de código-fonte e colaboração amplamente adotada por desenvolvedores em todo o mundo [git ]. Baseado no sistema de controle de versão *Git*, o *GitHub* oferece um conjunto abrangente de recursos que inclui desde a hospedagem de repositórios e controle de versão até colaboração em projetos, gerenciamento de tarefas, automação de integração contínua e garantia de segurança. Esta plataforma desempenha um papel fundamental no contexto do SPM neste projeto específico, pois possibilita a hospedagem, distribuição e colaboração eficaz dos pacotes desenvolvidos, sendo facilmente acessados por meio do repositório: Code4All. Com o *GitHub*, os *packages* podem ser publicados no formato adequado para o SPM, tornando-os acessíveis a outros desenvolvedores. Além disso, o *GitHub* oferece recursos cruciais de controle de versão, colaboração, rastreamento de problemas, integração contínua e gerenciamento de dependências, tornando-o uma escolha ideal para projetos de código aberto como este.

## 4.4. Desenvolvimento do pacote de componentes

Ao definir as ferramentas a serem adotadas no desenvolvimento do projeto e ao explorar quase 30 componentes básicos disponíveis no *SwiftUI*, juntamente com as necessidades dos usuários para criar interfaces, considerando especialmente os principais *widgets* que requerem acessibilidade, conforme indicado pelo estudo [Yan and Ramachandran 2019] mencionado anteriormente, propõe-se a criação dos seguintes componentes para a plataforma:

### 4.4.1. Componentes básicos

- **Text4All**: Usado para exibir texto na interface do usuário, já personalizado, solicitando apenas o texto que será apresentado e a descrição do texto que facilitará a leitura por meio do leitor de tela. Possibilitando que o desenvolvedor possa personalizar atributos como fonte, tamanho, cor, etc. É comumente usado para mostrar rótulos, títulos e informações de texto simples. O *Text* no *SwiftUI* possui recursos integrados que podem facilitar a implementação de acessibilidade por parte dos desenvolvedores, como etiquetas descritivas integradas: O *Text* permite adicionar etiquetas descritivas diretamente usando o atributo *.accessibility(label:)*,

como visto na figura 2, de forma que facilite a atribuição de descrições a elementos de texto, como rótulos ou cabeçalhos, tornando-os mais compreensíveis para leitores de tela [app c].

```
Text("Título da Página")
    .accessibilityLabel("Título da Página")
```

**Figure 2. SwfitUI Text implementation**

Dessa forma, o componente personalizado *Text4All* foi projetado para ser flexível e reutilizável, levando em consideração a acessibilidade desde sua implementação. O *Text4All* recebe o texto que será apresentado como argumento e também exige uma etiqueta de acessibilidade *.accessibilityTextInfo*, como pode ser visto na figura 3.

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

**Figure 3. Text4All implementation**

Com o *Text4All*, se torna necessária a utilização da descrição do texto por parte do desenvolvedor de maneira que já se tenha um direcionamento para entregar o mínimo de acessibilidade. Ao incluir explicitamente o argumento *.accessibility-TextInfo* durante o uso do componente personalizado, é uma forma de incentivo aos desenvolvedores a considerar a acessibilidade desde o início, o que pode levar a um aplicativo mais inclusivo e consciente de acessibilidade.
Além disso, ao utilizar o componente *Text4All*, os desenvolvedores também podem personaliza-lo de acordo com suas necessidades, incluindo cor, tamanho, fonte e outros atributos visuais.

- **Textfield4All**: Elemento de entrada de texto que permite aos usuários digitar informações, onde através da modularização foi criado baseado no ”*Text*” para propor a descrição do placeholder.
  Assim como o *Text*, o *TextField* permite que você forneça uma etiqueta descritiva diretamente usando o atributo *.accessibility(label:)*, como pode ser visto na figura 4. Isso ajuda a fornecer uma descrição clara e concisa do campo de entrada de texto para leitores de tela [app c].

```
TextField("Email", text: $inputText)
```

**Figure 4. SwfitUI Textfield implementation**

Assim, o *TextField4All* foi criado reutilizando a acessibilidade já adicionada no *Text4All*, como pode ser visto na figura 5 , de forma que o *textHint* presente no

*textfield* já conte com uma etiqueta de acessibilidade. Utilizando o *Text4All* integrado ao *TextField4All* permite com que o *textfield* se torne mais acessível através da descrição e ainda permite que o pacote fique mais modularizado através da reutilização dos seus próprios componentes.

```
TextField4All(textHint: Text4All, receivedText:
    $inputText)
```

**Figure 5. TextField4All implementation**

- **Button4All**: Usado para criar botões interativos na interface do usuário, possibilitando a adição de ações que são executadas quando o botão é tocado. Levando em consideração botões que contém texto, o componente criado para o *Text* também foi reutilizado para propor a descrição do botão.

  Visando trazer uma melhor descrição para os componentes e consequentemente para aplicações como um todo, levando em consideração que botões são componentes bastante presentes nas aplicações o *Button4All* foi projetado para ser mais flexível e reutilizável, sendo construído com o *Text4All*, de forma que também já tenha uma etiqueta de acessibilidade. Na criação de botões etiquetas de acessibilidade se tornam ainda mais necessárias, pois muitas vezes botões são compostos por elementos que não são auto descritivos o suficiente para que o *VoiceOver* possa interpreta-lo de maneira clara, elementos como: ícones, imagens e textos que não oferecem informações sobre o que o botão faz, como exemplificado na figura 6, que traz a implementação de botões que são para enviar um *email* de forma que apenas o texto ”enviar” possa ficar vago.

```
Button("Enviar", action: {sendEmail()})


Button4All(action: {sendEmail()}, textObject: Text4All(text:
    "Enviar", accessibilityTextInfo: "Enviar email"))
```

**Figure 6. Button implementation comparison**

Além disso, o *Button4All* oferece a flexibilidade de personalização e a facilidade de reutilização, tornando-o ideal para ser incorporado em várias partes do aplicativo sem a necessidade de duplicação de código de acessibilidade. Tudo o que você precisa fazer é fornecer o *Text4All*, que já inclui uma etiqueta de acessibilidade, ao utilizar esse componente. Apesar de poder ser personalizado, assim como os outros componentes citados anteriormente, o *Button4All* conta com uma personalização *default*, mais amigavél e intuitiva, como mostrado no ponto 1 da figura 7, em relação ao *Button* sem personalização representado no ponto 2.

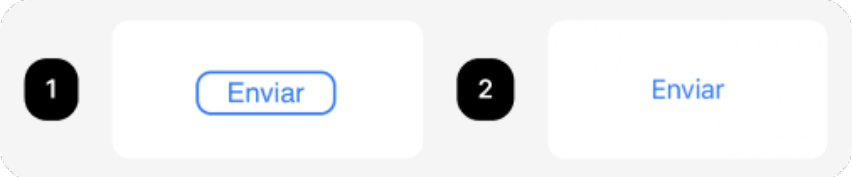


**Figure 7. Button visual comparison**

- **Toggle4All**: Elemento que permite ao usuário alternar entre dois estados, como *"enable"* e *"disable"*, dessa forma o componente foi criado com uma melhor descrição para o valor que é lido pelo *VoiceOver* de acordo com o seu estado atual. O componente *Toggle* presente em *SwiftUI* é adicionado acompanhado de uma *label* ao qual este elemento faz referência [app c]. Dessa forma a criação do *Toggle4All* foi pensada na mensagem a qual acompanha o *toggle* e também no valor que é lido quando o *toggle* está habilitado ou desabilitado, onde através dos valores adicionados nos *enableMessage* e *disableMessage*, torna-se possível, aliado ao *.accessibilityValue* que se encontra presente de forma abstraída para o desenvolvedor no componente *Toggle4All*, transmitir um estado melhor descrito para o usuário ao invés de "0" e "1", como pode ser visto na figura 8 , que mostra a implementação do *toggle* por meio do componente proposto pela pesquisa e o seguinte que seria o *toggle* nativo ilustrado no ponto 1. Através de uma análise feita através do *Accessibility Inspector*, verificando ambas implementações, pode-se visualizar no ponto 2 da figura 8, que é a inspeção do componente referente ao qual foi implementado utilizando o *package* de componentes proposto, o campo *"value"* ao qual faz menção ao estado atual do *toggle*, transmite um valor mais descritivo enquanto no ponto 3 da figura 8 o valor referente ao estado do *toggle* é apenas "0".

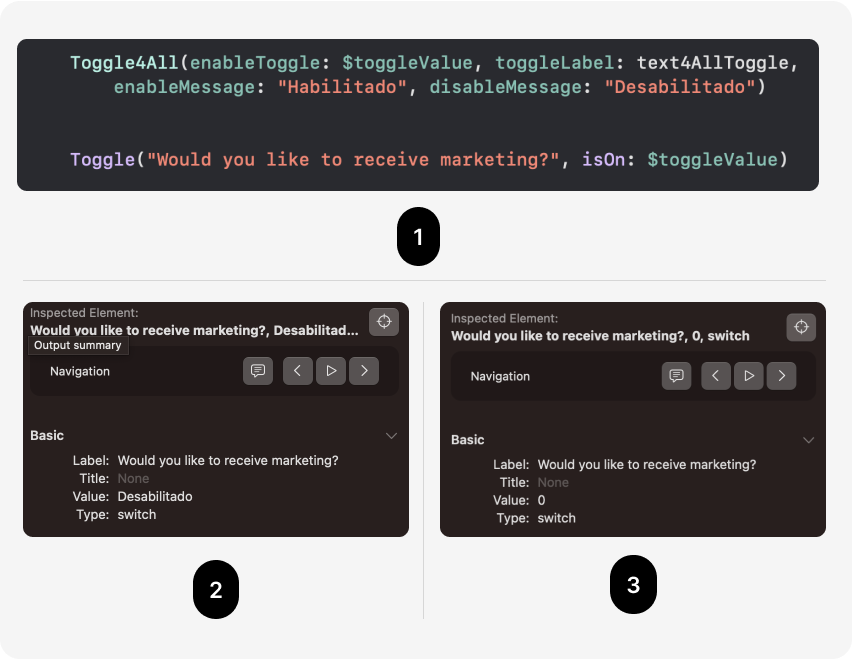


**Figure 8. Análise comparativa entre o componente de *toggle* nativo e o proposto pelo trabalho**

### 4.4.2. Componentes avançados

Após a definição dos componentes mais simples, torna-se possível criar componentes mais complexos e modulares, combinando esses blocos fundamentais. Isso é especialmente relevante ao criar um pacote de componentes SPM, onde a composição de elementos básicos desempenha um papel fundamental. Componentes como:

- **AgeSelectorComponentView**: Componente que permite aos usuários adicionar sua idade em um formulário por meio de uma *label* e dois botões para incrementar ou decrementar a idade, o *AgeSelectorComponentView* é criado através da combinação do *Button* e *Text* criados de maneira acessível para facilitar a descrição dos elementos presentes na tela, além disso levando em consideração a ação não trivial de avisar ao usuário através do *VoiceOver* que a *label* foi alterada, fornecendo uma mensagem para o usuário, anunciada quando o elemento de texto for incrementado ou decrementado. O componente *AgeSelectorComponentView* aborda a acessibilidade de forma integral, desde o texto até os botões. Cada elemento é acompanhado por etiquetas de acessibilidade claras e informativas, tornando-o acessível para usuários que dependem de tecnologias assistivas, como leitores de tela. Através desse componente pode-se exemplificar a abstração de funcionalidades complexas em um componente reutilizável. Ao encapsular a lógica de um seletor de idade completo em um único componente, se é promovida a reutilização de código e a simplificação do desenvolvimento, reduzindo a duplicação de código.

```
let ageSelectorText = Text4All(text: "Idade:",
    accessibilityTextInfo: "Idade: \(ageValue) anos")
let ageSelectorAnnouncement = "Idade atualizada para:"

AgeSelectorComponentView(count: ageValue, ageSelectorText:
    ageSelectorText, ageSelectorAnnouncementTextOnChange:
    ageSelectorAnnouncement)
```

**Figure 9. AgeSelectorComponent code import**

O uso do método *onChange* é utilizado para monitorar alterações no valor da idade para em seguida anunciar essas mudanças usando *postAccessibility* que será preenchido com a mensagem passada através do *ageSelectorAnnouncementTextOnChange*, de forma a proporcionar uma experiência mais rica aos usuários, pois eles recebem feedback imediato sobre o valor atualizado da idade sempre que o incrementam ou decrementam.

- **ToggleComponentView**: Elemento que permite ao usuário alternar entre dois estados, como "*enable*" e "*disable*". Baseado no *Toggle4All*, o *ToggleComponentView* também reutilizou componentes para propor a descrição da mensagem e fornece um aviso ao usuário, anunciado quando o *toggle* for chaveado. Assim como o *AgeSelectorComponentView*, o *ToggleComponentView* aborda a acessibilidade de forma integral, onde cada elemento é acompanhado por etiquetas de acessibilidade claras e informativas, e também faz uso do método *onChange*, utilizado para monitorar alterações no estado do *toggle* para em seguida anunciar es-

sas mudanças usando *postAccessibility* que será preenchido com a mensagem passada através do *toggleAccessibilityTextOnChange*, como pode ser visto na figura 10, fazendo com que os usuários recebam um *feedback* imediato sobre o valor atualizado do *toggle* sempre que alterado.

```
ToggleComponentView(toggleText: "Would you like to receive marketing?",
    toggleAccessibilityTextOnChange: "I would like to receive marketing")
```

**Figure 10. ToggleComponentView code import**

### 4.5. Testes automatizados

A criação de testes autônomos é uma prática essencial no desenvolvimento de pacotes *Swift* usando o SPM. Os testes unitários desempenham um papel crucial na garantia da qualidade do código, permitindo com que seja verificado se cada módulo de código funciona conforme o esperado. Neste contexto, a estrutura *XCTest* fornecida pela Apple é uma ferramenta valiosa para a criação e execução desses testes. O *XCTest* fornece uma série de classes e *APIs* que permitem a criação, organização e execução de testes de maneira estruturada.

A construção dos *XCTest* para testar o código implementado foi pensada e definida de maneira a abranger amplamente os componentes presentes no *package*. Os testes foram projetados para garantir que cada componente funcione corretamente e atenda às expectativas de funcionamento.

- **Seleção dos Componentes a serem Testados:** O primeiro passo na criação dos *XCTest* foi a seleção dos componentes que precisavam ser testados. Foram selecionados os quatro principais componentes: *Text4All*, *TextField4All*, *Toggle4All* e *Button4All*. De forma que para cada um desses componentes, foi planejado um conjunto de testes específicos.

  – Text4All testes:

| Cenário de Teste | Entrada | Status do Teste |
|---|---|---|
| Teste de Texto | Texto: ”Hello” | Passou |
| Teste de Cor | Cor: .blue | Passou |
| Teste de Fonte | Fonte: ”Helvetica” | Passou |
| Teste de Tamanho | Tamanho: 18 | Passou |
| Teste de Acessibilidade | Acessibilidade: ”Accessibility Info” | Passou |

**Table 1. Text4All cenário de testes**

  – TextField4All testes:

| Cenário de Teste | Entrada | Status do Teste |
|---|---|---|
| Teste do texto Hint | Text Hint: ”Username”,<br>Received Text: ”John” | Passou |
| Teste de Cor | Cor: .gray, IsSecureField: false | Passou |
| Teste de Campo Seguro | Text Hint: ”Password”,<br>Received Text: ”1234”,<br>IsSecureField: true | Passou |
| Teste de Tamanho | Tamanho: 18 | Passou |
| Teste de Acessibilidade | Acessibilidade: ”Accessibility Info” | Passou |

**Table 2. TextField4All cenário de testes**

– Toggle4All testes:

| Cenário de Teste | Entrada | Status do Teste |
|---|---|---|
| Teste de Habilitado | Habilitado: true,<br>Texto do Toggle: ”Ativar” | Passou |
| Teste de Desabilitado | Habilitado: false,<br>Texto do Toggle: ”Desativar” | Passou |

**Table 3. Toggle4All cenário de testes**

– Button4All testes:

| Cenário de Teste | Entrada | Status do Teste |
|---|---|---|
| Teste de Ação do Botão | Ação: Exibir mensagem,<br>Texto do Botão: ”Clique aqui” | Passou |
| Teste de Preenchimento | Preenchido: true,<br>Cor de Preenchimento: .green | Passou |
| Teste de Padrão de Estilo | Texto do Botão: ”Enviar”,<br>Estilo: Padrão | Passou |
| Teste de Tamanho | Tamanho: 18 | Passou |
| Teste de Acessibilidade | Acessibilidade: ”Accessibility Info” | Passou |

**Table 4. Button4All cenário de testes**

- **Especificação dos Testes:** Após a identificação dos componentes a serem testados, foram especificados os cenários de teste relevantes para cada classe. Para a classe *Text4All*, por exemplo, foram definidos testes para verificar se as propriedades *text*, *color*, *font*, *size* e *accessibilityTextInfo* eram atribuídas corretamente. Para a classe *Button4All*, os testes incluíram a verificação do comportamento do botão quando acionado, bem como a validação das configurações de estilo, como *padding*, *cornerRadius* e *borderWidth*.
- **Implementação:** Com os cenários de teste definidos, os testes foram implementados usando a estrutura *XCTest*. Cada teste foi encapsulado em um método de teste autônomo, começando com a palavra-chave ”*test*”. Dentro de cada método de teste, foram utilizadas as funções de *assert* da *XCTest*, como *XCTAssertEqual* e *XCTAssertTrue*, para verificar se os resultados esperados correspondiam aos resultados reais do código.

- **Execução e Validação:** Após a implementação, os testes foram executados por meio da funcionalidade de teste integrada ao *Xcode*. Durante a execução, os resultados foram analisados para identificar quais testes passaram e quais falharam, com intuito de garantir que os componentes funcionassem conforme o esperado.

Os *XCTest* foram criados de forma abrangente e estruturada, com o objetivo de validar cada aspecto do código presente no *package*. Os testes criados desempenham um papel crítico na garantia da qualidade do código, facilitando a detecção de erros e regressões à medida que o pacote é desenvolvido e mantido. Além disso, a organização cuidadosa dos testes e a documentação dos cenários de teste contribuem para a compreensão e manutenção contínua do pacote.

### 4.6. Testes de acessibilidade

Os testes de acessibilidade desempenham um papel fundamental no desenvolvimento de software, garantindo que aplicativos e sites sejam acessíveis a todas as pessoas, independentemente de suas habilidades ou deficiências. No desenvolvimento de um *package* de componentes acessíveis, os testes de acessibilidade se tornam essenciais.

Para conduzir os testes de acessibilidade foram utilizados o *Accessibility Inspector* e o *VoiceOver*, ferramentas amplamente utilizadas para conduzir testes de acessibilidade em aplicativos.

O *Accessibility Inspector*, como citado anteriormente, é uma ferramenta de desenvolvedor integrada ao ambiente de desenvolvimento *Xcode* da Apple [app a]. Ele fornece uma maneira detalhada de examinar a acessibilidade de elementos na interface do usuário de um aplicativo ou site. Com o *Accessibility Inspector*, pode-se selecionar elementos específicos, verificar suas propriedades de acessibilidade e identificar problemas, como rótulos ausentes, descrições inadequadas ou marcação semântica incorreta.

O *VoiceOver*, por sua vez, é um leitor de tela integrado nos dispositivos iOS e macOS. O *VoiceOver* permite que os usuários com deficiência visual interajam com aplicativos e sites por meio de gestos e toques, enquanto o dispositivo descreve em voz alta os elementos da interface do usuário [app 2009a]. Os testes com o *VoiceOver* envolveram navegar pelo aplicativo, ouvindo as descrições fornecidas pelo *VoiceOver* e identificando qualquer problema de acessibilidade, como elementos que não são corretamente anunciados ou ordens de leitura confusas. Através da combinação dessas duas ferramentas tornou-se possível criar uma abordagem abrangente para garantir a acessibilidade dos componentes criados. Os seguintes passos foram seguidos para realizar os testes de acessibilidade:

- **Criação de aplicações**: Nesta etapa aplicações foram criadas, utilizando os componentes presentes no *package*, para possibilitar os testes.
- **Ativação das Ferramentas**: O *Accessibility Inspector* é ativado nas preferências do *Xcode*, enquanto o *VoiceOver* é ativado nas configurações de acessibilidade do dispositivo.
- **Inspeção com o *Accessibility Inspector*:** O *Accessibility Inspector* foi usado para selecionar elementos específicos na interface do usuário e verificar suas propriedades de acessibilidade. Através disso foi possível identificar e registrar problemas para as correções necessárias de maneira mais ágil.

- **Navegação com o *VoiceOver*:** Com o *VoiceOver* ativado, testes de navegação pelo aplicativo foram realizados, interagindo com os componentes e ouvindo as descrições de voz para assegurar o funcionamento esperado de forma que qualquer problema de acessibilidade, como falta de rótulos, descrições inadequadas ou marcação incorreta, fossem identificados.
- **Correções e Melhorias:** Com base nos problemas identificados nas etapas anteriores, correções e melhorias foram implementadas no código referente aos componentes para garantir uma melhor acessibilidade.

Dessa forma, ao conduzir esses testes de acessibilidade utilizando o *Accessibility Inspector* e o *VoiceOver*, foi estabelecida uma sólida base para garantir que *package* de componentes seja acessível às pessoas com deficiência visual. Essas ferramentas desempenharam um papel crucial na identificação e correção de problemas de acessibilidade, permitindo o desenvolvimento de componentes mais acessíveis e inclusivos.

## 5. Metodologia de avaliação

Nesta etapa, serão descritos o processo de modelagem da entrevista e questionário de usabilidade e experiência. A análise da experiência dos usuários foram realizadas através de entrevistas com o objetivo de entender como o *package* de componentes poderia auxiliar desenvolvedores a realizar o desenvolvimento de novas aplicações de maneira ágil e acessível. Cada entrevista foi feita individualmente acontecendo de forma presencial ou remota, sendo entrevistados um total de 7 pessoas brasileiras e desenvolvedoras, com seus perfis descritos na Tabela 5.

| Entrevistado | Formação |
|---|---|
| E1 | Graduando em EC |
| E2 | Graduado em EC |
| E3 | Graduando em Musica |
| E4 | Graduando em SI |
| E5 | Graduando em Direito |
| E6 | Graduando em Eng. Biomed. |
| E7 | Graduando EC |

**Table 5. Perfil das pessoas entrevistadas.**

Inicialmente foram realizadas entrevistas estruturadas síncronas com os usuários desenvolvedores, realizando algumas perguntas básicas comuns à maioria das entrevistas e posteriormente foi concedido o acesso ao *package* de componentes para que fossem realizados testes semi-direcionados. Por fim, os entrevistados responderam ao questionário de usabilidade e experiência, com perguntas referentes à utilização do *package* e experiência em relação à acessibilidade no contexto de desenvolvimento.

### 5.1. Roteiro da entrevista estruturada:

O roteiro da entrevista aderiu a um formato previamente estabelecido, incorporando um teste de usabilidade do *package*. A duração estimada para a realização da entrevista variou entre 30 a 40 minutos. No início, foi realizado um conjunto de perguntas fundamentais, as quais foram aplicadas consistentemente em todas as entrevistas:

- Você já utilizou *packages* externos em seus projetos?
- Qual é a sua compreensão de acessibilidade digital?

Após as perguntas, o *package* foi fornecido para que o usuário pudesse baixar e adicioná-lo de forma local à um projeto *SwiftUI* já existente ou um novo projeto criado pelo usuário, seguindo o passo a passo descrito para a utilização de *packages* externos. Uma vez que o usuário importa o *package* ao seu projeto, inicia-se o teste semi-direcionado de usabilidade, seguindo os passos a seguir:

- Adicione um texto à view principal utilizando o componente *text4All*
- Adicione um *textField* à view principal utilizando o componente *textField4All*
- Adicione um botão à view principal utilizando o componente *button4All*
- Adicione um *toggle* à view principal utilizando o componente *toggle4All*
- Personalize os componentes adicionados utilizando seus atributos de estilização
- Adicione o componente *AgeSelectorComponentView* e o componente *ToggleComponentView*
- Rode a aplicação
- Abra o *Accessibility Inspector* e verifique as propriedades de acessibilidade e as descrições dos elementos dispostos na aplicação.

Durante a interação com os usuários, avaliou-se a eficácia do conjunto de componentes no processo de desenvolvimento dos entrevistados. É relevante enfatizar que não houve interferências, com exceção apenas dos momentos em que *feedbacks* eram fornecidos. Ao final da fase de teste semi-direcionado, foram feitas perguntas que se basearam na estrutura do *SCAMPER*, um acrônimo amplamente reconhecido que representa um conjunto de técnicas criativas. Cada letra corresponde a uma ação específica: substituir, combinar, adaptar, modificar, propor, eliminar e reorganizar.[Pontotel ]. Essas perguntas se concentraram especialmente em três aspectos cruciais da inovação e da otimização: a adaptação, com o propósito de preservar elementos eficazes; a eliminação do que é desnecessário para abrir espaço para a combinação de elementos; e a adição de componentes para facilitar a adaptação. As seguintes perguntas são:

- O que deve ser preservado?
- O que deve ser excluído?
- O que deve ser adicionado?

As perguntas visam atuar na identificação dos elementos da interface que estão funcionando bem e contribuindo de maneira positiva para a experiência do usuário, de forma a manter esses elementos para assegurar a continuidade de uma boa usabilidade, detectar elementos na interface que se revelam redundantes, confusos ou que não acrescentam significativamente à experiência do usuário. E por fim, identificar brechas na interface que poderiam ser preenchidas com novos elementos ou funcionalidades, com o propósito de melhorar a usabilidade.

### 5.1.1. Procedimento de condução das entrevistas

As entrevistas ocorreram num período de duas semanas, onde todos os entrevistados possuem experiência anterior no desenvolvimento para aplicações *iOS* e algum contato prévio com *SwiftUI*. Após as entrevistas, foram conduzidas análises das anotações registradas

durante esses encontros. A avaliação qualitativa desses candidatos desempenha um papel crucial na compreensão de sua percepção sobre a extensão e seus sentimentos ao utilizá-la.

### 5.2. Elaboração do questionário de usabilidade e experiência

Nesta seção, serão elencadas as perguntas utilizadas na construção do questionário para ter uma visão abrangente da experiência e perspectivas das pessoas entrevistadas em relação à acessibilidade no contexto do desenvolvimento de produtos e aplicações. Para isso foi utilizado um questionário com três opções de respostas pré-definidas, tornando o processo de resposta mais ágil. Além disso, foi incluída uma opção em aberto para os entrevistados que não se sentissem totalmente representados pelas opções pré-definidas. Inicialmente foram realizadas perguntas para entender mais sobre a visão e a presença de acessibilidade no meio dos entrevistados:

- **Pergunta 1**: Durante a sua formação, o tema da acessibilidade foi abordado? Como você o classificaria?
- **Pergunta 2**: Você já desenvolveu projetos pessoais? Em que medida a acessibilidade foi considerada?
- **Pergunta 3**: Nas experiências de trabalho que você teve, o tema da acessibilidade foi abordado? Como você o classificaria?
- **Pergunta 4**: Ao discutir acessibilidade digital, quais são as fontes de informação que você utiliza?
- **Pergunta 5**: No contexto do seu trabalho, como o desenvolvimento com foco em acessibilidade é percebido?
- **Pergunta 6**: Quais são os principais obstáculos que você enfrenta ao desenvolver aplicações acessíveis em seu trabalho?
- **Pergunta 7**: Na sua opinião, a responsabilidade pelo desenvolvimento de aplicações acessíveis deve partir das empresas ou dos desenvolvedores?
- **Pergunta 8**: Do seu ponto de vista, quais são as ações que podem ser adotadas para aprimorar a situação da acessibilidade no desenvolvimento de produtos?

Através destas perguntas iniciais tem-se o objetivo de explorar a visão e experiência do entrevistado em relação à acessibilidade no desenvolvimento de produtos e projetos. Elas abrangem temas como formação, experiência pessoal e profissional, fontes de informação, desafios enfrentados e responsabilidade.

Em seguida, uma segunda seção do questionário foi utilizada para coletar informações sobre a usabilidade do *package* e para avaliar a disposição dos entrevistados em adotá-la, além de coletar informações sobre a eficácia da ferramenta em questão, sua utilidade educacional e seu potencial para influenciar positivamente as práticas de desenvolvimento.

- **Pergunta 1**: Você acredita que o uso deste *package* facilita a implementação de componentes acessíveis?
- **Pergunta 2**: Você acredita que o uso do *package* ajudou ou pode vir ajudar outras pessoas a entender mais sobre acessibilidade?
- **Pergunta 3**: Você acredita que, ao usar um pacote externo com recursos acessíveis, você se sentiria mais inclinado a desenvolver os demais componentes de maneira acessível?

# 6. Resultados

## 6.1. Análise quantitativa e qualitativa da usabilidade do *package*

Nesta etapa, serão descritos os resultados obtidos mediante aos testes e analises da usabilidade do *package* coletados com usuários reais. Por meio da análise da experiencia dos usuários através das entrevistas realizadas pode-se entender como o *package* de componentes auxiliou desenvolvedores a realizarem o desenvolvimento de novas aplicações de maneira ágil e acessível.

A análise qualitativa se deu através da percepção do uso do *package* pelos entrevistados e *feedbacks* relatados durante o processo de utilização do *package*.

Na análise qualitativa foram abordados os principais pontos de *feedback* que emergiram consistentemente durante as entrevistas, bem como outros aspectos relevantes que foram identificados e registrados durante o teste semi-direcionado referente ao *package* de componentes podendo reunir os principais pontos de análise divididos em usabilidade e integração do package, preocupação sobre acessibilidade, facilidade de uso e personalização.

### 6.1.1. Usabilidade e integração do *package*

- Em algumas das entrevistas realizadas, a integração do package de maneira local apresentou alguns problemas, devido ao *Xcode* não encontrar os módulos adicionados, sendo necessário realizar a adição do módulo de maneira manual.
- Um dos entrevistados expressou preocupações sobre quais campos são obrigatórios para tornar uma aplicação acessível.
- Outro entrevistado elogiou a iniciativa e as possibilidades oferecidas pelo pacote para desenvolver componentes acessíveis, enfatizando a importância da personalização e da utilização de atributos descritivos. Em suas palavras: ”Gostei muito da iniciativa e das possibilidades que o *package* proporciona para o desenvolvimento de componentes e aplicações acessíveis como um todo”.
- Um dos entrevistados observou que o pacote pode ser uma solução valiosa para desenvolvedores com recursos limitados e menos conhecimento sobre acessibilidade.

### 6.1.2. Facilidade de uso e personalização

- A importância da customização e da personalização dos componentes foi enfatizada por um dos entrevistados, destacando a necessidade de nomes mais descritivos para os atributos dos componentes, como largura e altura.
- Durante uma das entrevistas, foi elogiada a facilidade e rapidez de inserção de componentes no código. No entanto, foi sugerido que fosse estabelecido um padrão consistente na ordem dos atributos, de modo a refletir a obrigatoriedade, colocando os atributos obrigatórios antes dos opcionais.
- Outro *feedback* importante sugere a criação de arquivos separados para cada componente no *package*, com o objetivo de aprimorar a organização e facilitar a identificação das funcionalidades individuais. Essa abordagem melhoraria a

eficiência no uso e na localização de componentes, resultando em uma experiência de desenvolvimento mais organizada e produtiva. Nas palavras do entrevistado: ”Poderia haver um arquivo para cada componente, facilitando para o desenvolvedor identificar rapidamente quais estão disponíveis na biblioteca.”

A análise quantitativa dos testes foi conduzida com base nas respostas coletadas por meio do questionário, visando alcançar diversos objetivos, tais como:

- Avaliar a eficácia do package.
- Identificar tendências ou padrões nas respostas.
- Medir a satisfação dos usuários com relação à funcionalidade.
- Identificar áreas de melhoria com base em dados quantitativos.

Com base nas respostas obtidas na primeira seção do questionário, foi observado que mais de 85% dos entrevistados relataram que o tema da acessibilidade não foi abordado em suas formações ou foi tratado de forma bastante limitada. Dentro desse percentual, a distribuição dos números foi uniforme, com exceção do entrevistado E3, que está cursando graduação em Música e relatou que a acessibilidade foi amplamente abordada em sua formação.

Aos trabalhos pessoais desenvolvidos, a acessibilidade não foi considerada prioridade desde o inicio do desenvolvimento e para mais de 42% dos entrevistados a acessibilidade não foi levada em consideração durante o desenvolvimento de suas aplicações. Nas experiências de trabalho dos entrevistados, o tema da acessibilidade não foi abordado ou foi abordado de forma limitada para mais de 85% dos entrevistados.

Em relação as fontes de informação quando se trata de acessibilidade, mais de 42% dos usuários buscam a recorrer à opiniões de colegas e *feedbacks* dos usuários, e quase 29% não costumam buscar informações sobre acessibilidade, número semelhante aos que buscam informações através de diretrizes conhecidas como as da Apple e WCAG.

Voltando ao contexto de trabalho, pouco mais de 14% dos entrevistados relatam que o desenvolvimento com foco em acessibilidade é considerado essencial para os projetos e mais de 85% dos entrevistados relatam que a acessibilidade no desenvolvimento pode ser considerada quando solicitado ou raramente/nunca é considerado uma prioridade. Entre os principais obstáculos, que mais de 71% dos entrevistados relatam enfrentar ao desenvolver aplicações acessíveis, estão a falta de conhecimento técnico e restrições de tempo e recursos.

Com relação a responsabilidade pelo desenvolvimento de aplicações acessíveis, mais de 57% dos entrevistados relatam que as empresas devem estabelecer diretrizes claras e incorporarem o desenvolvimento de aplicações acessíveis no escopo do projeto e quase 43% acreditam que a responsabilidade deve ser de ambas as partes.

Ao final, mais de 71% dos entrevistados acreditam que, para melhorar a situação da acessibilidade no desenvolvimento de produtos, a integração de diretrizes de acessibilidade devem ser realizadas desde o início do desenvolvimento. Com base nas respostas obtidas na segunda seção do questionário, notou-se que 100% dos entrevistados expressaram que o *package* facilita a implementação de componentes acessíveis. Destes, 28% destacaram a necessidade de mais funcionalidades para uma utilização mais eficaz. Além disso, todos os entrevistados acreditam que o uso do *package* ajudou ou tem o potencial

de auxiliar outros desenvolvedores a compreender mais sobre acessibilidade. Mais de 85% também afirmaram que, ao utilizar um *package* externo com recursos acessíveis, se sentiriam mais inclinados a desenvolver os demais componentes de forma acessível, devido à experiência positiva com a acessibilidade. Os demais entrevistados manifestaram que sua inclinação dependeria da facilidade de uso do *package*. Esses resultados sugerem uma recepção positiva em relação ao *package*, com algumas sugestões de melhorias.

### 6.1.3. Conhecimento e preocupação sobre acessibilidade

As entrevistas revelaram uma carência significativa de conscientização e práticas relacionadas à acessibilidade na formação, projetos pessoais e experiências de trabalho da maioria dos entrevistados. Tornou-se evidente que a acessibilidade não foi considerada uma prioridade em muitos desses contextos algo semelhante encontrado num estudo focado na perspectiva de desenvolvedores e designers sobre a implementação de diretrizes de acessibilidade móvel[de Almeida and Gama 2021], que conclui que os desenvolvedores frequentemente percebem a acessibilidade como uma preocupação secundária na tomada de decisões relacionadas à criação de aplicativos móveis.

Embora mais de 71% dos entrevistados acreditem que a integração de diretrizes de acessibilidade desde o início do desenvolvimento seja crucial para melhorar a situação da acessibilidade em produtos, observa-se que, em geral, a acessibilidade não tem sido uma prioridade.

Além disso, mais de 85% dos entrevistados relatou que a acessibilidade não foi abordada de maneira significativa em seus projetos de desenvolvimento pessoal ou em seus ambientes de trabalho. Essa observação levanta preocupações sobre a conscientização e o compromisso, tanto individualmente quanto por parte das empresas, em relação à acessibilidade.

## 7. Conclusão e trabalhos futuros

Com base nas entrevistas realizadas, ficou evidente que na usabilidade do *package* de componentes acessíveis encontra-se valores positivos para os usuários, mesmo que existam oportunidades para melhoria. No contexto de desenvolvimento de um pacote de componentes acessíveis destinado ao usuário final, é essencial que ele seja fácil de integrar ao código e com uso intuitivo, proporcionando uma implementação ágil para os desenvolvedores. Isso é fundamental para que os desenvolvedores se sintam confiantes de que a aplicação que estão construindo é acessível.

Para possíveis trabalhos futuros, há várias oportunidades de desenvolvimento a serem consideradas. Isso inclui a criação de novos componentes para expandir a variedade de recursos acessíveis oferecidos pelo pacote. Mais ainda, é importante adicionar descrições detalhadas aos atributos dos componentes, tornando-os mais fáceis de entender e utilizar.

Outra área de melhoria seria aprimorar a personalização oferecida pelo *SwiftUI*, permitindo que ele gerencie automaticamente a responsividade e a adaptação da interface do usuário a diferentes tamanhos de tela e dispositivos. Isso proporcionaria uma experiência mais consistente aos usuários finais.

Por fim, é fundamental analisar quais componentes são mais amplamente utilizados em aplicações e identificar aqueles que frequentemente carecem de atenção em termos de acessibilidade. Isso direcionaria os esforços para melhorar a acessibilidade onde ela é mais necessária, contribuindo para um desenvolvimento mais inclusivo e acessível de aplicativos.

## References


[app a] Accessibility Programming Guide for OS X: Testing for Accessibility on OS X — developer.apple.com. https://developer.apple.com/library/archive/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXTestingApps.html. [Accessed 11-09-2023].

[git ] GitHub: Let's build from here — github.com. https://github.com/. [Accessed 11-09-2023].

[app b] Swift packages — Apple Developer Documentation — developer.apple.com. https://developer.apple.com/documentation/xcode/swift-packages. [Accessed 11-09-2023].

[app c] SwiftUI — Apple Developer Documentation — developer.apple.com. https://developer.apple.com/documentation/swiftui/. [Accessed 11-09-2023].

[app d] SwiftUI Overview - Xcode - Apple Developer — developer.apple.com. https://developer.apple.com/xcode/swiftui/. [Accessed 11-09-2023].

[app e] Xcode 15 - Apple Developer — developer.apple.com. https://developer.apple.com/xcode/. [Accessed 11-09-2023].

[app f] XCTest — Apple Developer Documentation — developer.apple.com. https://developer.apple.com/documentation/xctest. [Accessed 14-09-2023].

[ada 1990] (1990). The Americans with Disabilities Act — ada.gov. https://www.ada.gov/. [Accessed 15-09-2023].

[app 2007] (2007). UIKit — Apple Developer Documentation — developer.apple.com. https://developer.apple.com/documentation/uikit. [Accessed 17-09-2023].

[app 2009a] (2009a). Acessibilidade - Visão — apple.com. https://www.apple.com/br/accessibility/vision/. [Accessed 11-09-2023].

[app 2009b] (2009b). Ative e treine o VoiceOver no iPhone — support.apple.com. https://support.apple.com/pt-br/guide/iphone/iph3e2e415f/ios. [Accessed 15-09-2023].

[goo 2009] (2009). Primeiros passos no Android com o TalkBack - Ajuda do Acessibilidade no Android — support.google.com. https://support.google.com/accessibility/android/answer/6283677?hl=pt-BR#:~:text=O%20TalkBack%20%C3%A9%20o%20leitor,e%20da%20vers%C3%A3o%20do%20TalkBack. [Accessed 15-09-2023].

[w3c 2018] (2018). Diretrizes de Acessibilidade para ContexFA;do Web (WCAG) 2.1 - PortuguxEA;s — w3c.br. https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/. [Accessed 11-09-2023].

[mec 2018] (2018). MinistxE9;rio da EducaxE7;xE3;o - MinistxE9;rio da EducaxE7;xE3;o — portal.mec.gov.br. http://portal.mec.gov.br/component/tags/tag/deficiencia-visual. [Accessed 11-09-2023].

[w3C 2018] (2018). Web Content Accessibility Guidelines (WCAG) 2.1 — w3.org. https://www.w3.org/TR/WCAG21/#:~:text=WCAG%202.1%20is%20developed%20through,%2C%20organizations%2C%20and%20governments%20internationally. [Accessed 15-09-2023].

[Accenture 2018] Accenture (2018). Getting to equal: The disability inclusion advantage.

[Chakra ] Chakra, U. A simple, modular and accessible component library that gives you the building blocks you need to build your react applications. *Chakra UI: Simple, Modular and Accessible UI Components for your React Applications. URl: https://chakra-ui.com (visited on 02/27/2023)*.

[Civil 2015] Civil, C. (2015). Lei nº 13.146, de 6 de julho 2015. *Institui a lei brasileira de inclusão da pessoa com deficiência (estatuto da pessoa com deficiência). Brasília*.

[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. IEEE.

[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. In *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*, pages 1–9.

[Hussain and Omar 2020] Hussain, A. and Omar, A. M. (2020). Usability evaluation model for mobile visually impaired applications.

[Krainz et al. 2016] Krainz, E., Feiner, J., and Fruhmann, M. (2016). Accelerated development for accessible apps–model driven development of transportation apps for visually impaired people. In *Human-Centered and Error-Resilient Systems Development: IFIP WG 13.2/13.5 Joint Working Conference, 6th International Conference on Human-Centered Software Engineering, HCSE 2016, and 8th International Conference on Human Error, Safety, and System Development, HESSD 2016, Stockholm, Sweden, August 29-31, 2016, Proceedings 8*, pages 374–381. Springer.

[Organization et al. 2011] Organization, W. H. et al. (2011). *World report on disability 2011*. World Health Organization.

[Pontotel ] Pontotel, R. Scamper: entenda o que é esse método e como ele pode melhorar os resultados na sua empresa — pontotel.com.br. https://www.pontotel.com.br/scamper/#:~:text=continue%20a%20leitura!-,O%20que%20%C3%A9%20o%20Scamper%3F,de%20novos%20projetos%2C%20por%20exemplo. [Accessed 11-09-2023].

[Schmutz et al. 2016] Schmutz, S., Sonderegger, A., and Sauer, J. (2016). Implementing recommendations from web accessibility guidelines: would they also provide benefits to nondisabled users. *Human factors*, 58(4):611–629.

[(WAI) 2022] (WAI), W. W. A. I. (2022). WAI-ARIA Overview — w3.org. https://www.w3.org/WAI/standards-guidelines/aria/. [Accessed 11-09-2023].

[Yan and Ramachandran 2019] Yan, S. and Ramachandran, P. (2019). The current status of accessibility in mobile apps. *ACM Transactions on Accessible Computing (TACCESS)*, 12(1):1–31.

[Zhang et al. 2021] Zhang, X., de Greef, L., Swearngin, A., White, S., Murray, K. I., Yu, L., Shan, Q., Nichols, J., Wu, J., Fleizach, C., Everitt, A., and Bigham, J. P. (2021). Screen recognition: Creating accessibility metadata for mobile applications from pixels. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.