Avaliação do suporte à acessibilidade visual em iOS Nativo conforme diretrizes do SiDI

Pedro Henrique Pinheiro Costa[1], **Windson Viana de Carvalho**[1,3]

[1]Instituto Universidade Virtual (UFC Virtual) – Universidade Federal do Ceará (UFC)
CEP 60.320-275 – Fortaleza, CE – Brasil

[2]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Mestrado e Doutorado em Ciências da Computação (MDCC)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

henriquephpcosta@gmail.com, windson@virtual.ufc.br

Abstract. *Smartphones have become so present in the daily life of the population to the point of composing half of the digital devices in Brazil. Despite this, some studies indicate that many mobile applications still fail to provide accessibility features so that the entire population can make full use of the diversity of available applications. In this context, this research analyzes accessibility support in the development of a Native iOS application following SiDI guidelines. The application was evaluated by an automated tool and two experts, before and after implementing the support, which produced an application with Voice Over support, but some accessibility issues.*

Resumo. *Smartphones têm se tornado tão presentes no cotidiano da população ao ponto de comporem metade dos aparelhos digitais no Brasil. Apesar disso, alguns estudos indicam que várias aplicações móveis ainda falham em suprir recursos de acessibilidade para que toda a população possa exercer pleno uso da diversidade de aplicativos disponíveis. Nesse contexto, esta pesquisa analisa o suporte à acessibilidade no desenvolvimento de uma aplicação em iOS Nativo seguindo diretrizes do SiDI. A aplicação foi avaliada por uma ferramenta automatizada e dois especialistas, antes e após a implementação do suporte, produzindo uma aplicação com suporte para Voice Over, mas alguns problemas de acessibilidade.*

1. Introdução

Desde seu surgimento, a presença dos *smartphones* na sociedade tem se intensificado. De acordo com Meirelles, essa presença é tão forte no Brasil que mais da metade dos dispositivos digitais no país são *smartphones*: cerca de 234 milhões de celulares inteligentes dentre um total de 424 milhões de dispositivos como tablets, notebooks e desktops [Meirelles 2020]. Dito isto, o total de *smartphones* é maior do que a população Brasileira, de 211,8 milhões de habitantes no mesmo período [IBGE 2020]. Estes dados demonstram a magnitude do mercado de dispositivos móveis e o impacto que suas aplicações podem realizar na sociedade.

Embora este setor apresente tal dimensão e esteja em constante desenvolvimento, ainda existem muitos aspectos que requerem melhorias, como o suporte à acessibilidade.

Segundo um estudo, de um conjunto de 2.270 aplicações, cerca de 89% delas apresentaram problemas severos de acessibilidade [Chen et al. 2021]. Estendendo esse cenário para o universo total de aplicações disponíveis, isso significaria que cerca de 1 em cada 10 aplicativos seria acessível a qualquer pessoa independente de ter qualquer deficiência ou não.

Em um levantamento do Instituto Brasileiro de Geografia e Estatística (IBGE) em 2010, o Brasil apresentava cerca de 6,5 milhões de pessoas com alguma deficiência visual [IBGE 2010]. Visando esse público, *smartphones* contam com serviços de acessibilidade que são capazes de realizar a leitura da descrição de botões e textos além de indicar ações que o usuário pode realizar nos vários aplicativos [Chantre 2015]. Com a importância do desenvolvimento de suporte para acessibilidade, vários estudos e organizações buscaram desenvolver diretrizes para orientar a implementação de funcionalidades com acessibilidade em dispositivos móveis, como o trabalho de Ballantyne *et. al*. [Ballantyne et al. 2018] e o Guia de Acessibilidade do SiDI [SIDI 2015], um instituto de tecnologia brasileiro em parceria com a Samsung.

Serviços de acessibilidade requerem que os desenvolvedores das aplicações e dos componentes de interface implementem o suporte para se integrar a eles, como por exemplo descrições para serem lidas pelos leitores de tela. Entretanto, a acessibilidade dos aplicativos no mercado e o próprio conhecimento dos desenvolvedores mobile sobre como implementar a acessibilidade ainda permanecem imaturos [Leite et al. 2021][Vendome et al. 2019]. Cerca de 46% dos mais de 13 mil aplicativos observados por [Vendome et al. 2019], por exemplo, não adicionaram essas descrições, o que impede que seus usuários tenham a leitura assistida de forma adequada. Além disso, uma pesquisa feita com 857 desenvolvedores brasileiros mostrou o baixo conhecimento e a baixa adesão na implementação de aspectos de acessibilidade nos aplicativos móveis que eles desenvolvem [Leite et al. 2021].

Uma das formas de amenizar tais deficiência é garantir que os componentes de interface padrões das plataformas de desenvolvimento já disponham de elementos acessíveis, assim como, que a plataforma de desenvolvimento ofereça mecanismos simples de implementação dos aspectos de acessibilidade. Pesquisas anteriores de Ribeiro *et. al*. (2019), e Bezerra e Viana (2021) buscaram analisar o suporte de acessibilidade das plataformas de desenvolvimento Ionic, React Native e Android Nativo. Para tal, versões de uma mesma aplicação móvel foram implementadas nas plataformas-alvo e cada uma teve sua acessibilidade checada. Nesse contexto, a presente pesquisa segue os mesmos princípios e busca analisar o desenvolvimento de acessibilidade para uma aplicação móvel em dispositivos iOS. Este trabalho desenvolveu um aplicativo Prova de Conceito (PoC) nos mesmos moldes das pesquisas anteriores e foi baseado nas mesmas diretrizes. Seu objetivo foi identificar como é o nível de acessibilidade básico da plataforma iOS e a viabilidade de atender os requisitos definidos diretrizes de acessibilidade do SiDI [SIDI 2015], visando facilitar o desenvolvimento de acessibilidade para essa plataforma ao esclarecer os recursos e procedimentos necessários para um suporte básico ao Voice Over.

O restante deste artigo é estruturado da seguinte forma: a Seção 2 apresenta os conceitos de acessibilidade mais importantes para o trabalho, enquanto a Seção 3 descreve a metodologia adotada. A Seção 4 apresenta o desenvolvimento inicial da PoC e a Seção 5 descreve o processo da implementação de elementos de acessibilidade. Na Seção 6 é

demonstrado o resultado das avaliações automatizadas e a Seção 7 apresenta as avaliações de dois especialistas em design e interação. Por fim, a Seção 8 discute os resultados encontrados e a Seção 9 apresenta as considerações finais sobre o trabalho.

2. Acessibilidade

Este estudo usa o conceito de acessibilidade como sendo a possibilidade de que qualquer pessoa, com quaisquer capacidades físico-motoras, tenha acesso aos benefícios de uma vida em sociedade [Nicholl and Boueri Filho 2001]. No meio digital, essa definição se estende à prática inclusiva de programas que possam ser utilizados por todos, com deficiências ou não [ISO Central Secretary 2011]. No contexto de *smartphones*, cujo principal método de interação é sua interface gráfica, as deficiências visuais se tornam particularmente relevantes.

A Organização Mundial da Saúde (OMS), em sua Classificação Internacional da Funcionalidade, Incapacidade e Saúde (CIF), define categorias para a deficiência visual desde a perda leve da visão até ausência total de visão [Wohlin 2014]. A cegueira é caracterizada por acuidade visual menor que 0,1 com a melhor correção no melhor olho ou campo visual abaixo de 20 graus [García 2017]. Outra categoria é a baixa visão, que ocorre quando há comprometimento da visão em ambos os olhos mesmo com correções como óculos, lentes de contato ou cirurgias [Campos et al. 2007].

Visando estabelecer diretrizes para auxiliar na melhoria da acessibilidade em dispositivos móveis, o SiDI apresenta um guia [SIDI 2015] com orientações como:

- Design Minimalista: privilegiar componentes relevantes e com função comprovada, evitando informações desnecessárias.
- Fluxo Natural: ordenar telas sequencialmente e seus componentes em fluxo natural na linguagem ocidental, seguindo da esquerda para a direita e de cima para baixo.
- Coerência Externa: privilegiar a localização de componentes de acordo com convenções estabelecidas.
- Coerência Interna: garantir que a aplicação tenha um padrão de cores, posicionamento, fontes e demais elementos, estabelecendo coerência entre todas as suas telas.
- Evitar Excesso de Informações: muitas informações em uma única tela se torna cansativo e dificulta a interação com leitores de tela.
- Contraste de Cores: utilizar cores que facilitem a identificação de componentes e percepção de alterações na tela.

Em *smartphones*, a principal ferramenta para a acessibilidade visual é o uso de leitores de tela, que são capazes de descrever os elementos na tela do dispositivo e fornecem feedback sonoro para as ações realizadas pelo usuário. Além disso, esses sistemas podem reconhecer gestos específicos na tela para realizar certas ações, como rolagem de tela ou acessar opções extras da aplicação específica. Atualmente, os principais leitores de tela em dispositivos móveis são os disponíveis por padrão de acordo com o sistema: o Talkback[1] para Android e o Voice Over[2] para iOS.

[1] `https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback`

[2] `https://www.apple.com/accessibility/vision/`

Durante o desenvolvimento de acessibilidade para esses sistemas, é possível utilizar inspetores automatizados que verificam alguns elementos da acessibilidade da aplicação: aplicações iOS podem ser avaliadas pelo Apple Accessibility Inspector[3] e projetos Android podem fazer o mesmo com o Google Accessibility Scanner[4]. Ambas as ferramentas destacam componentes sem os devidos identificadores de acessibilidade e utilizam métodos de inspeção de usabilidade com a avaliação de parâmetros dentro de certas heurísticas para verificar aspectos como contraste de cores e posicionamento de elementos da interface.

3. Metodologia

Este trabalho é categorizado como exploratório ao buscar maior familiaridade com o problema, visando torná-lo mais explícito [Gil et al. 2002], com o problema sendo, neste caso, a falta de indícios sobre o suporte ao desenvolvimento de acessibilidade visual em iOS Nativo.

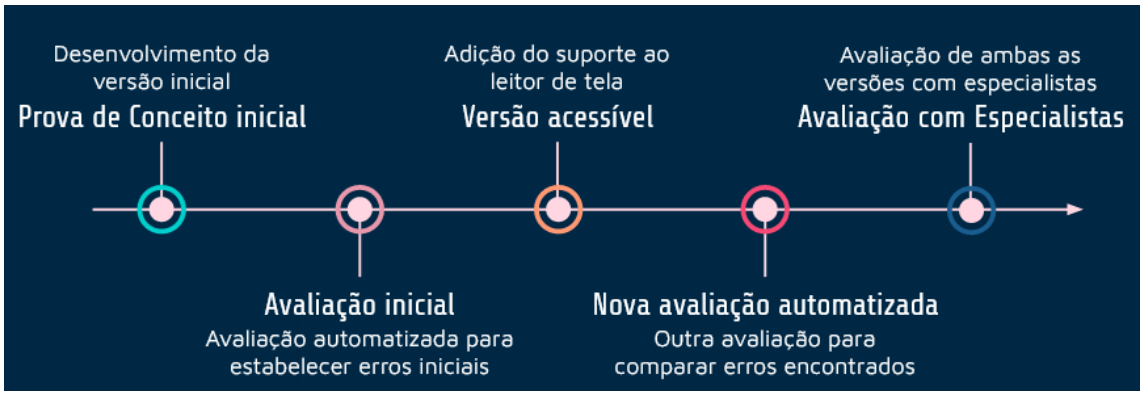

Figura 1. Momentos da Pesquisa

Inicialmente, foi feita uma pesquisa exploratória utilizando a literatura científica encontrada através do Google Scholar[5], utilizando palavras-chave como *"accessibility", "mobile apps", "iphone"* e *"accessibility testing"*. Essa pesquisa teve o intuito de definir melhor o tema ao adquirir mais informações sobre o contexto atual. Então foram pesquisados materiais sobre acessibilidade, dispositivos móveis, e o sistema iOS. Após esse estudo, foi delineado um projeto dando continuidade aos trabalhos de [Ribeiro et al. 2019], e [Bezerra and Viana 2021], os quais foram obtidos a partir do repositório de trabalhos de conclusão da UFC Virtual. Assim optou-se por realizar um estudo semelhante, fazendo uma análise do desenvolvimento em iOS, mas adicionando a avaliação com especialistas, uma vez que diversos estudos indicam como apenas diretrizes de acessibilidade não são suficientes para identificar os problemas de acessibilidade em uma aplicação [Calvo et al. 2016].

O elemento central da pesquisa é uma aplicação Prova de Conceito (PoC) implementada em duas versões: uma versão com apenas os elementos básicos para a aplicação

[3]https://developer.apple.com/library/archive/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXTestingApps.html

[4]https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor

[5]https://scholar.google.com.br/

e uma versão com adição de elementos acessíveis ao leitor de tela. A PoC em questão é a mesma desenvolvida nos trabalhos de [Ribeiro et al. 2019] e [Bezerra and Viana 2021], no intuito de dar continuidade à pesquisa e permitir a comparação futura entre os resultados das plataformas. A escolha do sistema iOS foi baseada no mesmo princípio de continuidade, uma vez que as pesquisas anteriores abordaram apenas o Android, com programação Nativa e *cross-platform*. Além disso, o iOS é a segunda plataforma mais popular para dispositivos móveis no mundo [6].

A PoC é uma aplicação com temática *fitness* que armazena informações das atividades físicas do usuário. A Figura 2 mostra algumas das telas em alta fidelidade definidas por [Ribeiro et al. 2019].

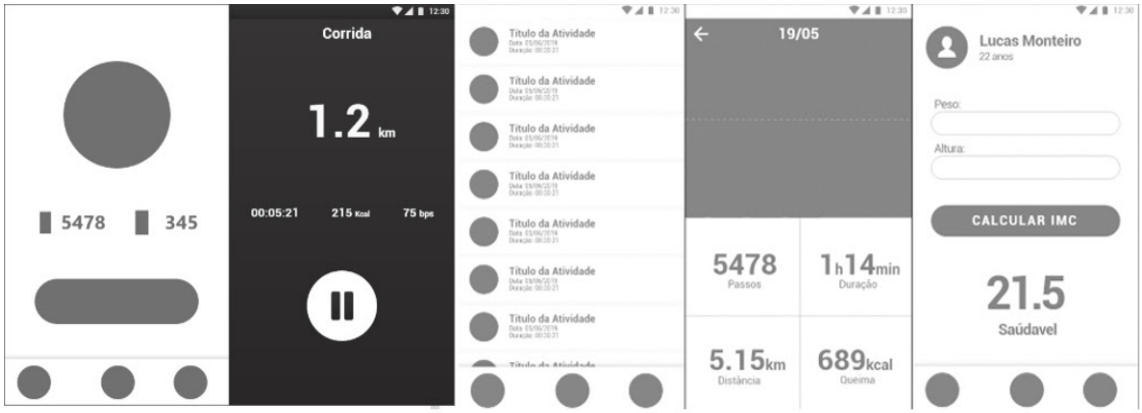

Figura 2. Telas do protótipo em alta fidelidade, por [Ribeiro et al. 2019]

Em sua primeira versão, a PoC foi desenvolvida com recursos padrões do desenvolvimento iOS e então avaliada pelo Apple Accessibility Inspector, o inspetor automatizado de acessibilidade oficial da Apple. O intuito dessa Versão Padrão era estabelecer o desempenho básico da plataforma em termos de acessibilidade, quando o desenvolvedor não busca implementar o suporte voluntariamente.

As ferramentas para cada avaliação foram um iPhone 11 modelo MWLY2BZ/A com a versão 15.2.1 do sistema iOS e o leitor de tela Voice Over nele contido, um Macbook Pro modelo MacBookPro16,1 com a versão 11.6 do sistema macOS e o Apple Accessibility Inspector contido no Xcode versão 13.2.1 (13C100).

Com o inspetor de acessibilidade, foi feita uma verificação em cada tela do aplicativo através de uma simulação de iPhone feita pelo Xcode, gerando uma lista de aspectos a melhorar para ter um suporte adequado de acessibilidade. Então o autor navegou por toda a aplicação utilizando o Voice Over, tomando nota da leitura de tela completa e das descrições de elementos em foco.

Então a aplicação foi refatorada para atender aos mesmos requisitos utilizados por [Ribeiro et al. 2019] e [Bezerra and Viana 2021], os quais são estabelecidos como mandatórios na seções de interação e navegação do Guia de Acessibilidade do SiDI, a saber:

- R31 - O leitor de telas deve informar ao usuário todos os eventos visíveis.

[6] `https://gs.statcounter.com/os-market-share/mobile/worldwide`

- R32 - O leitor de telas deve informar o conteúdo de um componente assim que tocado, interrompendo qualquer leitura em andamento.
- R33 - A aplicação deve fornecer feedback sonoro sobre todas as ações executadas pelo usuário.
- R34 - A aplicação deve fornecer feedback visual sobre todas as ações executadas pelo usuário.
- R35 - As telas da aplicação, exceto popups (pequenas janelas que se abrem por cima da tela sendo visualizada), devem disponibilizar link para a tela principal do aplicativo.
- R36 - As telas da aplicação devem disponibilizar o botão ”Voltar”para a tela anteriormente acessada pelo usuário.
- R39 - A aplicação deve suportar a navegação baseada em foco.
- R40 - A aplicação deve informar possíveis erros de interação ao usuário.

Em seguida, a nova versão foi analisada novamente pelo inspetor automatizado a fim de comparar os erros remanescentes mesmo após a implementação do suporte para acessibilidade. Por fim, dois especialistas avaliaram a aplicação em relação aos mesmos critérios estabelecidos pelo SiDI.

4. Prova de Conceito

A Prova de Conceito (PoC) é denominada FitApp. Ela tanto armazena informações das atividades físicas do usuário como gera relatórios para acompanhamento do progresso dos exercícios feitos. Além disso, é possível monitorar o exercício em tempo real e calcular o Índice de Massa Corporal (IMC) ao preencher um breve formulário. Conforme mencionado anteriormente, a versão Android Nativo, cujas telas são apresentadas na Figura 3, foi implementada originalmente por [Ribeiro et al. 2019], sendo usada como referência por [Bezerra and Viana 2021] e para a implementação da versão em iOS Nativo neste trabalho.

Figura 3. Telas da aplicação desenvolvida para Android Nativo, por [Ribeiro et al. 2019]

4.1. Versão iOS Nativo

A aplicação em iOS Nativo foi desenvolvida utilizando a linguagem de programação Swift[7]. As três telas principais (Atividade, Histórico e Perfil) são *UI View Controllers*

[7]https://www.apple.com/br/swift/

inseridas em uma *UI Tab Bar Controller*, que permite a navegação na barra inferior. As telas de Corrida e Gráficos também são *UI View Controllers*, mas inseridas a partir das suas respectivas telas anteriores (a tela de Atividade é anterior à de Corrida e a de Histórico anterior à de Gráficos). Todas as telas são apresentadas na Figura 5.

Também é importante ressaltar a necessidade de confirmação da adição do idioma Português na seção de Localização do projeto, conforme a Figura 4. Sem sua adição, certas leituras podem ser feitas incorretamente pelo Voice Over, como no uso do teclado numérico, que lê a tecla novamente após ser pressionada para confirmar o comando.

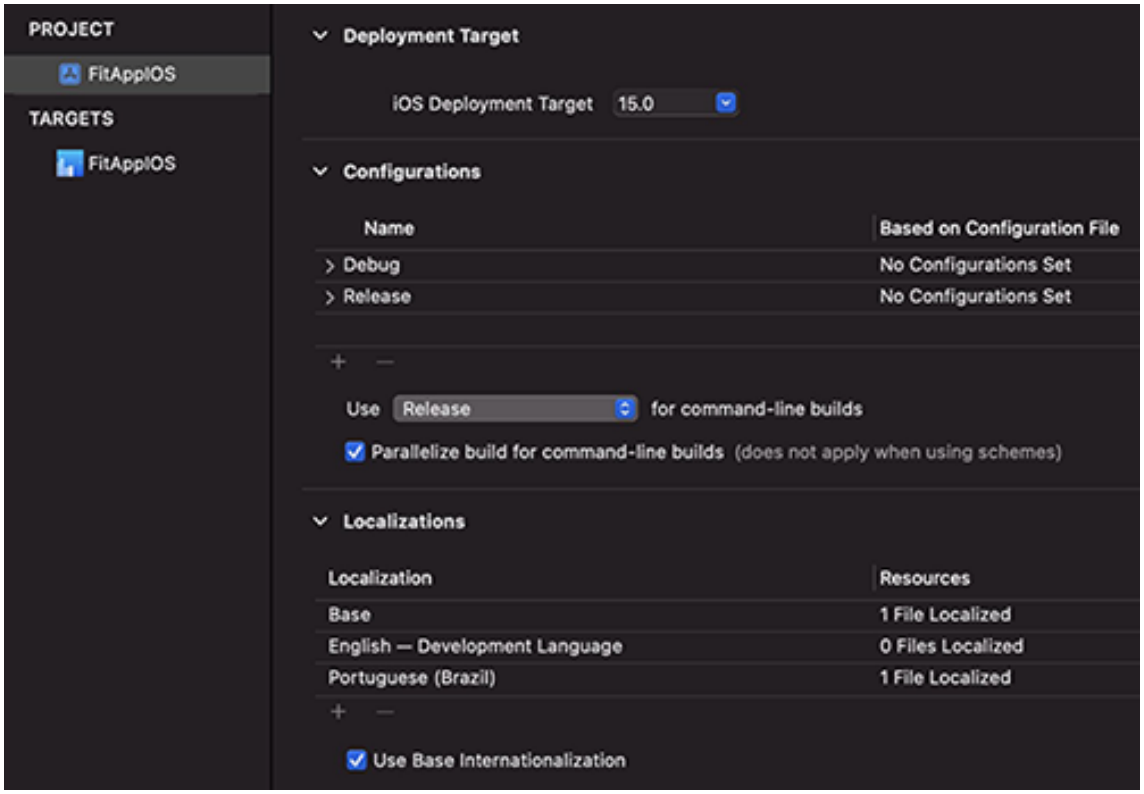

Figura 4. Configuração da Localização para Português

Os elementos visuais foram criados utilizando os SF Symbols[8], ícones padrões do sistema, e formas básicas, além de duas imagens personalizadas para os gráficos nas telas de Atividade e Gráficos, as quais foram feitas por [Ribeiro et al. 2019]. Ao final, o projeto continha 52 arquivos, gerando um arquivo de 444 KB, enquanto a aplicação instalada ocupava 377 KB.

O aplicativo pode ser testado conectando um iPhone a um sistema macOS com o projeto[9] aberto no Xcode[10], o ambiente de desenvolvimento oficial Apple, ou com um sistema macOS emulando um dispositivo iOS. Por um período limitado, para permitir a avaliação dos especialistas, a aplicação também pôde ser testada via Testflight[11], a plataforma Apple para testes de aplicações em desenvolvimento.

[8] https://developer.apple.com/sf-symbols/
[9] https://github.com/Silmunia/Fit-App-iOS/
[10] https://developer.apple.com/xcode/
[11] https://developer.apple.com/testflight/

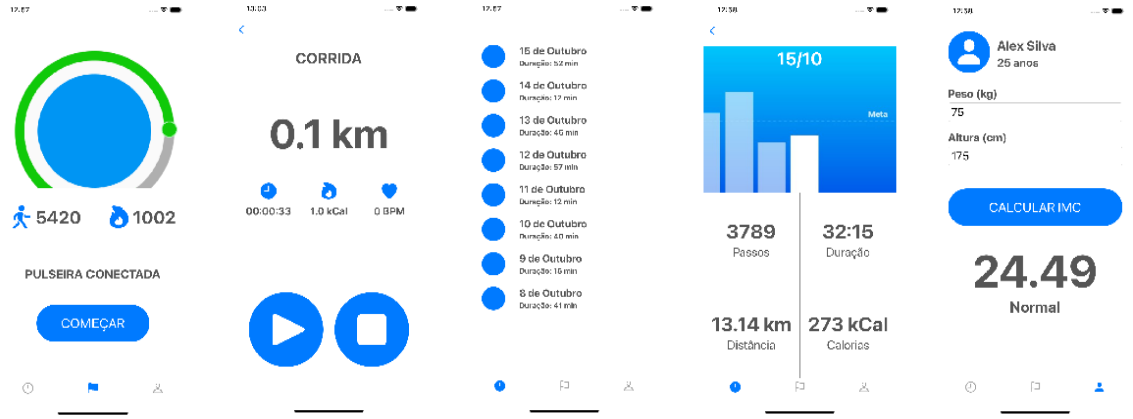

Figura 5. Telas da aplicação desenvolvida para iPhone

5. Desenvolvimento da Acessibilidade

O desenvolvimento da acessibilidade foi orientado pelo Apple Accessibility Inspector e pelo uso do Voice Over pelo autor, assim como os requisitos do Guia de Acessibilidade Móvel do SiDI, mencionados anteriormente.

O elemento mais relevante para a implementação do suporte ao Voice Over foi a propriedade *accessibilityLabel* em todos os elementos textuais. Com funcionalidade semelhante ao *contentDescript* no sistema Android, adicionar o texto que informe ao Voice Over como descrever o elemento foi suficiente para ter o suporte adequado em quase todos os elementos. Por exemplo, a Figura 6 mostra o código antes e depois de usar essa propriedade no componente que exibe o tempo de corrida. Outra propriedade interessante nos casos de botões e outros elementos que mudavam o estado da aplicação foi *accessibilityHint*, que permite ao Voice Over descrever brevemente o resultado da interação com o elemento.

Os únicos casos onde *accessibilityLabel* não bastou para um suporte adequado foram com elementos personalizados, como as informações na tela de Gráficos. Uma vez que o título da informação e os dados em si ("Passos" e "3789", por exemplo) são apresentados em fontes com tamanhos diferentes, foi necessário que fossem separados em duas *UI Labels* diferentes. Porém, graças à ordem de leitura do Voice Over, isso resultava na tela sendo descrita como "3789. 32 minutos e 15 segundos. Passos. Duração" em vez de "3789 Passos. Duração de 32 minutos e 15 segundos".

O problema foi resolvido ao agrupar pares de *UI Labels* em um elemento *UI View*, atribuir a esse elemento uma *accessibilityLabel* com a descrição desejada e então usar a propriedade *isAccessibilityElement* para que o leitor de tela identifique o componente personalizado como um elemento importante, conforme pode ser observado na Figura 7. Essa propriedade também pôde ser usada para que o Voice Over ignore as imagens de gráficos, uma vez que elas não possuem funcionalidade na PoC.

6. Avaliações Automatizadas

Ao final do desenvolvimento de cada versão, foi feita uma avaliação utilizando o Apple Accessibility Inspector e Voice Over. A primeira avaliação teve como objetivo verificar a acessibilidade da aplicação quando o desenvolvedor não busca adicionar os elementos

```
lazy var timeLabel: UILabel = {
    let label = UILabel()
    label.translatesAutoresizingMaskIntoConstraints = false
    label.font = .systemFont(ofSize: 22, weight: .bold)
    label.textColor = .darkGray
    label.text = String(format: "%02d:%02d:%02d", timeCounter[0], timeCounter[1],
        timeCounter[2])
    label.textAlignment = .center
    self.view.addSubview(label)
    return label
}()
```

```
lazy var timeLabel: UILabel = {
    let label = UILabel()
    label.translatesAutoresizingMaskIntoConstraints = false
    label.font = .systemFont(ofSize: 22, weight: .bold)
    label.textColor = .darkGray
    label.text = String(format: "%02d:%02d:%02d", timeCounter[0], timeCounter[1],
        timeCounter[2])
    label.accessibilityLabel = "\(timeCounter[0]) horas, \(timeCounter[1]) minutos e
        \(timeCounter[2]) segundos"
    label.textAlignment = .center
    self.view.addSubview(label)
    return label
}()
```

Figura 6. Código do texto do cronômetro na Versão Padrão (acima) e Versão Acessível (abaixo), fazendo uso da *accessibilityLabel*

de suporte para o Voice Over, enquanto a segunda avaliação oferece uma comparação de resultados quando o aplicativo conta com uma implementação de acessibilidade baseada nas diretrizes escolhidas do SiDI.

6.1. Resultados da Versão Padrão

O Voice Over apresentou os elementos conforme seu tipo corretamente, porém alguns elementos não apresentaram descrição adequada ou suficiente para que sua funcionalidade fosse clara, conforme a Tabela 1. Botões contendo texto tiveram problemas de agrupamento onde texto e botão foram descritos como entidades distintas, e, conforme descrito anteriormente na seção de desenvolvimento, elementos da Tela de Gráficos não foram agrupados adequadamente.

Tabela 1. Quantidade de erros contabilizados com o Voice Over

Tela da Aplicação	Elementos sem marcadores ou marcadores errados	Elementos de fundo lidos	Erros de feedback após ação	Erros de agrupamento de conteúdo
Atividade	6	0	2	1
Histórico	3	0	3	0
Perfil	8	0	2	1
Gráficos	4	0	3	4
Corrida	4	0	0	3

A Tabela 2 a seguir demonstra os avisos encontrados em cada tela com o Apple Accessibility Inspector. A maioria foi sobre a responsividade das fontes e outros refe-

```
lazy var durationElement: DataViewMultilabel = {
    let element = DataViewMultilabel()
    element.translatesAutoresizingMaskIntoConstraints = false
    element.setLabels(titleString: "Duração", valueString: "32:15", acessibilityString:
        "Duração de 32 minutos e 15 segundos")
    self.view.addSubview(element)
    return element
}()
```

```
public func setLabels(titleString: String, valueString: String, acessibilityString: String) {
    elementTitle.text = titleString
    elementValue.text = valueString
    self.isAccessibilityElement = true
    self.accessibilityLabel = acessibilityString
}
```

Figura 7. Código do componente personalizado que permitiu a leitura adequada do texto na Tela de Gráficos da Versão Acessível

rentes aos ícones, além de alguns avisos de contraste pela escolha de cores da Prova de Conceito.

Tabela 2. Quantidade de erros contabilizados com Apple Accessibility Inspector

Tela da Aplicação	Falha de Contraste	Contraste quase aceitável	Texto potencialmente inacessível	Elemento sem descrição	Sem suporte para mudança de tamanho da fonte
Atividade	1	2	3	1	4
Histórico	0	0	0	0	16
Perfil	0	1	1	3	9
Gráficos	1	0	0	0	9
Corrida	0	5	5	2	5

6.2. Resultados da Versão Acessível

Com a implementação de suporte adequado ao Voice Over, o autor concluiu a navegação da aplicação com sucesso e nenhum comportamento inadequado ou indesejado foi encontrado. Enquanto isso, ao analisar através do inspetor de acessibilidade, os alertas de elementos sem descrição foram totalmente eliminados, mas outros alertas, como contraste de cores e fontes sem responsividade, não foram afetados, uma vez que o desenvolvimento de acessibilidade focou nos elementos mínimos para o suporte ao Voice Over. A Tabela 3 exibe os resultados encontrados nesta versão da Prova de Conceito.

Tabela 3. Erros contabilizados com Apple Accessibility Inspector após implementação de suporte à acessibilidade

Tela da Aplicação	Falha de Contraste	Contraste quase aceitável	Texto potencialmente inacessível	Sem suporte para mudança de tamanho da fonte
Atividade	1	2	3	4
Histórico	0	0	0	16
Perfil	0	1	1	9
Gráficos	1	0	0	9
Corrida	0	5	5	5

6.3. Comparação dos Resultados

Ao somar os erros encontrados em cada versão, tanto os erros apontados pelo Apple Accessibility Inspector quanto os encontrados pelo autor navegando pela aplicação com o Voice Over, cujos vídeos estão disponíveis em um repositório[12], é possível ver a diferença de erros através da Tabela 5: apesar de todas as telas ainda apresentarem erros, a Versão Acessível demonstra uma redução de cerca de 42% em relação à Versão Padrão.

Tabela 4. Total de erros por tela em cada versão da aplicação

Telas da Aplicação	Erros na Versão Padrão	Erros na Versão Acessível
Atividade	20	10
Histórico	22	16
Perfil	25	11
Gráficos	17	10
Corrida	24	15
Total	108	62

7. Avaliação com Especialistas

Feitas as avaliações com a ferramenta automatizada e o levantamento de erros no Voice Over feito pelo autor, também foram realizadas avaliações por dois especialistas em design e interação, uma vez que a implementação de suporte à acessibilidade orientada apenas por diretrizes de acessibilidade não é capaz de identificar tantos problemas quanto profissionais especializados, como demonstrado por [Calvo et al. 2016] com especialistas avaliando aplicações com base na WCAG 2.0 (Web Content Accessibility Guidelines) e ainda identificando problemas que as orientações não cobriram. Desta forma, o intuito das avaliações foi fornecer perspectivas detalhadas dos possíveis problemas no uso do Voice Over.

7.1. Sobre os Especialistas

Um especialista é egresso da graduação em Sistemas e Mídias Digitais na Universidade Federal do Ceará (UFC), com 8 anos de experiência profissional em design digital e mestrando em Ciência da Computação na UFC com foco na Interação Humano-Computador (IHC) e Acessibilidade. O outro especialista também é egresso da graduação em Sistemas e Mídias Digitais na UFC, com 4 anos de experiência em design digital e cerca de 3 anos orientando design voltado para iOS e outros dispositivos Apple na Apple Developer Academy IFCE.

7.2. Procedimento

Primeiro os especialistas foram informados sobre a natureza da aplicação desenvolvida, explicando seu intuito e funcionalidades, além da natureza da pesquisa. Então o autor entregou a cada especialista um documento contendo uma tabela para cada tela da aplicação, apresentando os requisitos escolhidos em formato de perguntas, conforme a lista a seguir.

[12] `https://drive.google.com/drive/folders/1m3Up7BzVZb2Xfg2pjrM9d4UWneKD4U9Y?usp=sharing`

Tabela 5. Qualificações dos Especialistas

Qualificações	Especialista A	Especialista B
Formação	Bacharelado em Sistemas e Mídias Digitais	
Experiência com Design	8 anos	4 anos
Outras	Mestrando em Ciência da Computação com foco em IHC e Acessibilidade	3 anos orientando design para iOS na Apple Developer Academy IFCE

- R31 - O leitor de telas informa ao usuário todos os eventos visíveis?
- R32 - O leitor de telas informa o conteúdo de um componente assim que tocado, interrompendo qualquer leitura em andamento?
- R33 - A aplicação fornece feedback sonoro sobre todas as ações executadas pelo usuário?
- R34 - A aplicação fornece feedback visual sobre todas as ações executadas pelo usuário?
- R35 - As telas da aplicação, exceto popups (pequenas janelas que se abrem por cima da tela sendo visualizada), disponibilizam link para a tela principal do aplicativo?
- R36 - As telas da aplicação disponibilizam o botão ”Voltar”para a tela anteriormente acessada pelo usuário?
- R39 - A aplicação suporta a navegação baseada em foco?
- R40 - A aplicação informa possíveis erros de interação ao usuário?

O intuito foi que cada avaliador navegasse por cada versão da aplicação utilizando o Voice Over e então contabilizasse os erros que encontrasse referente a cada requisito. Por fim, também era possível fazer observações específicas para requisitos em cada versão do aplicativo e ao final do documento também havia um espaço para observações gerais.

Ambos os especialistas começaram com a Versão Padrão da aplicação e depois avaliaram a Versão Acessível.

7.3. Análise dos Resultados

Os especialistas encontraram múltiplos erros do Voice Over na Versão Padrão da aplicação. Além disso, ambos os especialistas encontraram outros problemas na aplicação, como nomenclaturas de botões e resultados de certas interações, sem relação direta com a leitura do Voice Over.

Enquanto isso, as avaliações da Versão Acessível não apresentaram erros de leitura do Voice Over, mas ainda identificaram problemas de acessibilidade, como falta de feedback sonoro ou visual em algumas telas. A Tabela 6 apresenta um resumo da contagem de erros feita pelos especialistas em cada versão da aplicação desenvolvida.

Os documentos com as avaliações íntegras podem ser encontrados no mesmo repositório[13] que contém as versões da aplicação desenvolvida.

[13] `https://github.com/Silmunia/Fit-App-iOS/tree/main/FitAppIOS/Evaluation-Docs-(BR)`

Tabela 6. Problemas apontados pelos especialistas em cada versão

Contexto	Problemas na Versão Padrão	Problemas na Versão Acessível
Tela de Atividade	7	3
Tela de Histórico	6	6
Tela de Perfil	8	7
Tela de Gráficos	8	5
Tela de Corrida	14	5
Outras Observações	11	8
Total	54	34

8. Discussão

Comparando os resultados obtidos, foi possível perceber como cada verificação revelou uma nova camada de problemas. Enquanto o Accessibility Inspector foi capaz de revelar a maioria dos problemas que precisavam ser resolvidos para o uso adequado do Voice Over, ao navegar pela aplicação, o autor pôde encontrar erros de falta de informação suficiente e ordem de leitura errada que a ferramenta automatizada não foi capaz de identificar. Por fim, os especialistas revelaram problemas mais sutis, como os feedbacks visuais e sonoros de certas telas serem insuficientes, apesar de estarem presentes.

Na Versão Padrão, foram encontrados múltiplos erros que impossibilitaram o uso adequado do leitor de tela. Embora elementos padrões do sistema fossem identificados adequadamente, o leitor de tela não foi capaz de tratar os casos de elementos personalizados. Enquanto que na Versão Acessível, o autor não encontrou problemas com o Voice Over, mas o inspetor automatizado e os especialistas ainda apontaram elementos que reduzem a acessibilidade da aplicação, como a falta de feedback visual ou sonoro suficientes em vários momentos, apesar do leitor de tela funcionar.

Dito isto, é interessante ressaltar como adicionar o suporte para o Voice Over foi particularmente simples para os elementos com design mais próximo do design padrão utilizado por aplicações oficiais Apple, bastando adicionar uma propriedade contendo a descrição desejada, sem modificar significativamente o fluxo de desenvolvimento. No entanto, um elemento da Tela de Gráficos cuja personalização se resumia a uma ordem de leitura ligeiramente diferente foi o bastante para surgir um erro de leitura no Voice Over, exigindo então o uso de propriedades complementares para um funcionamento adequado.

Finalmente, os resultados da pesquisa apresentam as seguintes limitações:

- As avaliações foram conduzidas apenas em sistemas iOS, o que é insuficiente para uma conclusão adequada sobre as diretrizes uma vez que sistemas Android representam uma porção significativamente maior dos dispositivos atuais
- Apenas as diretrizes mandatórias de interação e navegação do SiDI foram seguidas, sem considerar outros requisitos do seu Guia de Acessibilidade
- As avaliações foram feitas por apenas dois especialistas, o que limita a diversidade de perspectivas sobre a acessibilidade da aplicação

9. Considerações Finais

O trabalho desenvolveu duas versões de uma mesma aplicação móvel para o sistema operacional iOS, uma versão desenvolvida visando as funcionalidades básicas da Prova

de Conceito (PoC) e outra desenvolvida visando fornecer suporte para o leitor de tela Voice Over, com o desenvolvimento orientado pelas diretrizes selecionadas do SiDI. Desta maneira, a pesquisa dá continuidade aos trabalhos de [Ribeiro et al. 2019] e [Bezerra and Viana 2021] desenvolvendo a mesma PoC, buscando a comparação entre os resultados das plataformas utilizadas.

Essas aplicações foram avaliadas por uma ferramenta automatizada, que apontou elementos sem suporte para o Voice Over, e então utilizadas pelo autor com o leitor de tela, revelando outros elementos com suporte problemático. Por fim, dois especialistas em design e interação avaliaram ambas as versões com base nas diretrizes do SiDI. Através dessas avaliações, foi possível produzir uma aplicação com funcionamento adequado do Voice Over, mas ainda apresentando problemas de acessibilidade, assim demonstrando os recursos e processos necessários para desenvolver um suporte básico para o leitor de telas do sistema iOS.

Para trabalhos futuros, recomenda-se a realização de um estudo comparativo incluindo outras plataformas, como Android Nativo e cross-platform como o Flutter, para comparar a acessibilidade desenvolvida nestes com o iOS Nativo. Além disso, é aconselhável buscar a participação de mais especialistas e usuários reais que façam uso cotidiano de leitores de tela, obtendo opiniões profissionais mais diversas assim como a perspectiva de usuários finais sobre a aplicação.

Referências

Ballantyne, M., Jha, A., Jacobsen, A., Hawker, J. S., and El-Glaly, Y. N. (2018). Study of accessibility guidelines of mobile applications. In *Proceedings of the 17th international conference on mobile and ubiquitous multimedia*, pages 305–315.

Bezerra, F. S. R. and Viana, W. (2021). Desenvolvimento nativo vs ionic vs react native: uma análise comparativa do suporte à acessibilidade em android.

Calvo, R., Seyedarabi, F., and Savva, A. (2016). Beyond web content accessibility guidelines: Expert accessibility reviews. In *Proceedings of the 7th international conference on software development and technologies for enhancing accessibility and fighting info-exclusion*, pages 77–84.

Campos, I., SÁ, E., and SILVA, M. (2007). Formação continuada a distância de professores para o atendimento educacional especializado. *Governo Federal. Brasília*.

Chantre, J. R. M. (2015). *Testes automáticos de acessibilidade em aplicações móveis*. PhD thesis, Universidade da Beira Interior, Covilhã, Portugal.

Chen, S., Chen, C., Fan, L., Fan, M., Zhan, X., and Liu, Y. (2021). Accessible or not an empirical investigation of android app accessibility. *IEEE Transactions on Software Engineering*.

García, J. (2017). Livro branco da tecnologia assistiva no brasil. *São Paulo: Instituto de Tecnologia Social–ITS BRASIL*.

Gil, A. C. et al. (2002). *Como elaborar projetos de pesquisa*, volume 4. Atlas São Paulo.

IBGE, I. (2010). Censo demográfico 2010. *IBGE: Insituto Brasileiro de Geografia e Estatística*.

IBGE, I. (2020). Estimativa da população dos municípios para 2020. *IBGE: Insituto Brasileiro de Geografia e Estatística*.

ISO Central Secretary (2011). ISO/IEC 25010:2011, systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models. Standard ISO/IEC TR 25010:2011, International Organization for Standardization, Geneva, CH.

Leite, M. V. R., Scatalon, L. P., Freire, A. P., and Eler, M. M. (2021). Accessibility in the mobile development industry in brazil: Awareness, knowledge, adoption, motivations and barriers. *Journal of Systems and Software*, 177:110942.

Meirelles, F. S. (2020). Pesquisa anual do fgvcia. *Uso da TI–Tecnologia da Informação nas Empresas. Fundação Getúlio Vargas*.

Nicholl, A. R. J. and Boueri Filho, J. J. (2001). O ambiente que promove a inclusão: conceitos de acessibilidade e usabilidade. *Revista Assentamentos Humanos*, 3(2):49–60.

Ribeiro, L. M., Façanha, A. R., and Viana, W. (2019). Desenvolvimento nativo vs ionic: uma análise comparativa do suporte à acessibilidade em android. *Anais do XII Simpósio Brasileiro de Computação Ubíqua e Pervasiva*.

SIDI (2015). Guia para o desenvolvimento de aplicações móveis acessíveis.

Vendome, C., Solano, D., Liñán, S., and Linares-Vásquez, M. (2019). Can everyone use my app? an empirical study on accessibility in android apps. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 41–52. IEEE.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pages 1–10.