Native or Web-Hybrid Apps? An Analysis of the Adequacy for Accessibility of Android Interface Components Used with Screen Readers

Lucas Pedroso Carvalho
Departamento de Ciência da Computação
Universidade Federal de Lavras
Lavras, MG, Brasil
lucaspedrosocarvalho@gmail.com

André Pimenta Freire
Departamento de Ciência da Computação
Universidade Federal de Lavras
Lavras, MG, Brasil
apfreire@dcc.ufla.br

ABSTRACT
Creating accessible mobile applications involves several important design decisions in order to accommodate the needs of disabled users, especially people with visual disabilities who use screen readers. The goal of the study presented in this paper was to analyze the adequacy of interface components to implement mobile applications, in order to identify the main accessibility problems that could be encountered by developers when using them, and the main strategies to overcome those issues. We performed an accessibility evaluation of a sample of 30 Android interface components present in 3 prototypes of mobile applications, employing the development techniques of native applications with and withoug Web components and hybrid development using the Apache Cordova framework. The results showed that the prototypes developed using web components were more compatible with accessibility criteria in the Web Content Accessibility Guidelines (WCAG 2.0) and with the screen reader Talkback. The most frequent accessibility problems in such components occurred in tables, headings and multimedia elements. Based on the current challenges for accessibility in mobile applications, we highlight the limitations of some interface components and emphasize that more studies need to be carried out to consolidate accessibility guidelines and good practices for mobile devices.

Author Keywords
Accessibility; Mobile Applications; Screen Readers

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces; Screen Design

INTRODUÇÃO

Proporcionar às pessoas com deficiência a participação em atividades, como o uso de serviços, produtos e informações é essencial para que elas possam ser mais independentes e seguras para exercer seus direitos e sua cidadania. Tal inclusão não deve ser diferente quanto se trata dos recursos tecnológicos. É importante garantir que essas pessoas também consigam realizar suas tarefas e utilizar sistemas da mesma forma com que as pessoas que não possuem deficiência utilizam.

Segundo a Parte 171 da norma internacional ISO 9241 [11], a acessibilidade de sistemas interativos pode ser definida como a "usabilidade de um produto, serviço, ambiente ou facilidade por pessoas com a mais ampla gama de capacidades". Considerando a definição de usabilidade da ISO 9241 - Parte 11 [10], isso significaria que aumentar a acessibilidade das interações humano-computador promove maior eficácia, eficiência e satisfação para os usuários que possuem uma grande variedade de capacidades e preferências, incluindo pessoas com deficiência. A ISO 9241 - Parte 171 [11] ainda reconhece que alguns usuários de *software* necessitarão de tecnologias de apoio para utilizar algum sistema, como um leitor de telas para pessoas cegas ou recursos de adaptação para pessoas com baixa visão.

Para tornar o conteúdo da Web mais acessível, foram criadas diretrizes de desenvolvimento para sistemas com o objetivo de alcançar um maior nível de acessibilidade. O grupo de acessibilidade Web do W3C (World Wide Web Consortium) definiu um vasto conjunto de recomendações para acessibilidade de páginas Web, o WCAG (Web Content Accessibility Guidelines) 2.0 [4]. Esse modelo foi concebido para tentar acompanhar evoluções tecnológicas e inspirou vários padrões no mundo todo, inclusive o eMAG (Modelo de Acessibilidade em Governo Eletrônico) [2], do governo brasileiro.

Com o crescimento e a necessidade do uso de dispositivos móveis para acesso à Internet, a inclusão de pessoas com deficiência no uso de aplicações móveis surge como um desafio a ser enfrentado pela sociedade. O desenvolvimento desses aplicativos deve considerar a forma como os usuários que possuem limitações e deficiências podem interagir com eles. Se os componentes de interface não forem acessíveis ou se os recursos de Tecnologia Assistiva não fornecerem apoio necessário, os usuários irão encontrar problemas de compatibilidade, desempenho e usabilidade [11]. Dessa forma, a acessibilidade depende de importantes decisões de *design* na construção dos aplicativos e a abordagem escolhida para implementação dos

componentes de interface tem implicações importantes para a acessibilidade.

Diferentes abordagens tem sido utilizadas na construção desses aplicativos e diferentes técnicas podem ser utilizadas. Neste trabalho investigamos três formas de desenvolvimento de aplicações móveis: desenvolvimento de aplicativo híbrido (utilizando como base tecnologias Web incorporadas aos aplicativos), desenvolvimento de aplicativo nativo (sem uso de recursos Web) e desenvolvimento de aplicativo nativo com recursos da Web em partes da interface.

Em vista das diferentes técnicas para construção de aplicações móveis, ainda existem poucas informações para desenvolvedores sobre as possibilidades de fornecimento de recursos de acessibilidade para pessoas com deficiência utilizando diferentes técnicas. Os membros da comunidade internacional do W3C têm dedicado esforços para adequar o WCAG 2.0 e os seus princípios, diretrizes e critérios de sucesso para que possam ser aplicados ao conteúdo da Web móvel [16]. Além disso, existem algumas recomendações técnicas de acessibilidade de desenvolvedores de plataformas para dispositivos móveis (como iOS e Android). Entretanto, pouco se sabe sobre o alcance dos recursos de acessibilidade para aplicações móveis desenvolvidas de forma híbrida, nativa ou nativa com incorporação de recursos Web, em uma combinação da natureza dos componentes de interface e dos serviços disponibilizados por recursos de Tecnologia Assistiva.

O objetivo deste trabalho foi de realizar uma análise, a partir da implementação e avaliação técnica de protótipos de aplicativos nas modalidades de desenvolvimento nativa, híbrida e nativa com partes Web, e verificar problemas de acessibilidade e estratégias de acesso disponíveis para contornar os problemas de cada uma. Dessa forma, a partir dos resultados obtidos, espera-se poder contribuir para informar as escolhas nas decisões de design de aplicativos móveis visando acessibilidade.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 são apresentados os principais conceitos para o desenvolvimento de aplicações para dispositivos móveis, seguido de uma revisão da literatura no campo da acessibilidade para essas aplicações. Na Seção 3 é apresentado o método utilizado nas inspeções de acessibilidade. Na Seção 4 são discutidos os resultados encontrados. Finalmente, na Seção 5, é apresentada uma conclusão e propostas de trabalhos futuros.

REFERENCIAL TEÓRICO

Aplicativos para Dispositivos Móveis

O desenvolvimento de aplicativos para dispositivos móveis tem crescido, assim como a popularização desses. Acompanhando o crescente desenvolvimento de aplicativos para a plataforma Android, o uso do sistema operacional também vem crescendo no Brasil, possuindo uma participação do mercado de brasileiro de 82.15% [22].

Diferentes meios podem ser utilizados para criar aplicações para os dispositivos móveis, concebendo diversas formas de aplicativos. Em definições encontradas na literatura, os aplicativos desenvolvidos para *smartphones* e *tablets* podem ser classificados como: aplicativos nativos *(native apps)*, aplicações Web *(web apps)* ou aplicativos híbridos *(hybrid apps)*. Segundo Budiu [3], não existe uma única e melhor solução para o desenvolvimento móvel: cada um deles tem seus pontos fortes e fracos. A escolha de um em relação à outro depende das necessidades únicas de cada organização. Entretanto, cada forma de implementação traz implicações importantes para as decisões de design, particularmente para a acessibilidade.

O desenvolvimento de aplicativos nativos faz uso de linguagens específicas para determinadas plataformas, como por exemplo, Java para Android, Objective C para iOS e C++ para Windows Phone. Em particular, o desenvolvimento nativo para a plataforma Android também permite o uso de componentes embutidos com a linguagem de marcação HTML (HyperText Markup Language), possibilitando a construção de aplicações com componentes da Web. Nesse tipo de desenvolvimento, o código utilizado em uma plataforma não pode ser reaproveitado para um sistema operacional diferente, sendo necessário reescrever todo o código para alcançar portabilidade.

No desenvolvimento nativo, é possível explorar ao máximo a totalidade dos recursos disponíveis no dispositivo, como sensores, acelerômetro, bússola, recursos de chamadas e interface, processamento gráfico e outros, permitindo a construção de aplicativos com muitos recursos, capazes de processar vídeos e imagens de grandes resoluções. Esses aplicativos podem utilizar as notificações do sistema e serem executados *offline*.

As aplicações Web são páginas na Internet acessadas como qualquer outra através de um navegador que esteja previamente instalado nos dispositivos móveis. Esses *websites* são desenvolvidos utilizando tecnologias amplamente difundidas, como o HTML, CSS (Cascading Style Sheets) e JS (JavaScript), tornando o processo de desenvolvimento mais ágil por não necessitar de códigos diferentes para cada plataforma. Por outro lado, os *web apps* possuem acesso limitado aos recursos de hardware e dados dos dispositivos, sendo dependentes de conexão à Internet para acesso ao conteúdo.

O desenvolvimento híbrido visa combinar as vantagens do desenvolvimento nativo e Web para a criação de aplicações. Para isso, diferentes *frameworks* e ferramentas auxiliam o desenvolvedor, tais como: Appcelerator Titanium[1], Adobe PhoneGap [2], Apache Cordova [3], Ionic [4], dentre outros.

Essas ferramentas permitem que no desenvolvimento sejam utilizadas tecnologias voltadas para o desenvolvimento Web, como o HTML e JavaScript, e ao final do processo o aplicativo seja instalado no dispositivo da mesma forma que um aplicativo nativo, favorecendo a possibilidade de interoperabilidade para diferentes plataformas. Os aplicativos híbridos utilizam um navegador nativo da plataforma que está incorporado na aplicação, no caso do Android, o WebView.

Neste estudo foram investigadas algumas tecnologias para o desenvolvimento de aplicações móveis nativas e híbridas para

[1] Appcelerator Titanium - Disponível em appcelerator.com
[2] Adobe PhoneGap - Disponível em build.phonegap.com
[3] Apache Cordova - Disponível em cordova.apache.org
[4] Ionic - Disponível em ionicframework.com

a plataforma Android, a mais utilizada no Brasil. O ambiente de desenvolvimento integrado Android Studio, da Google, foi utilizado para desenvolver aplicativos nativos e o *framework* Apache Cordova para desenvolver aplicativos híbridos.

O Apache Cordova permite o uso de tecnologias padrões da Web (HTML, CSS e JavaScript) para personalizar a interface de usuário e para conceber as funções lógicas do aplicativo. Os aplicativos executados em cada plataforma dependem de conexões de interface compatíveis com padrões para acessar recursos de cada dispositivo, como sensores, dados e outros.

Acessibilidade para Aplicativos em Dispositivos Móveis

Vários estudos têm dedicado esforços para pesquisar métodos para avaliar a acessibilidade de aplicativos em dispositivos móveis. No entanto, poucos trabalhos estão focados no estudo de como esses aplicativos são construídos e nas implicações da escolha de tipos de componentes de interface na acessibilidade dos aplicativos quando utilizados em dispositivos móveis.

Pesquisas na área de acessibilidade com usuários que utilizam leitores de tela para acessar aplicações e *websites* em seus dispositivos móveis tem crescido bastante nos últimos anos. Um estudo realizado por Chiti e Leporini [6] contou com a participação de 4 usuários com deficiência visual, que avaliaram um protótipo de uma aplicação desenvolvida para a plataforma Android. Os usuários forneceram sugestões úteis, como questões relacionadas aos gestos *touchscreen*, que podem auxiliar os desenvolvedores que trabalham na concepção e no desenvolvimento de tecnologias assistivas para dispositivos móveis.

Em um estudo posterior, Leporini *et al.* [13] avaliaram os problemas de usabilidade e acessibilidade em dispositivos móveis da Apple com o leitor de tela VoiceOver. O trabalho envolveu uma inspeção de usabilidade das interfaces dos dispositivos e *feedbacks* de 55 usuários cegos. Os resultados confirmaram que o VoiceOver é fundamental para que esses usuários possam interagir com aplicações móveis, porém, ainda existem problemas de usabilidade que devem ser sanados, como a falta de clareza de detalhes nos elementos interativos.

De acordo com Siebra *et al.* [21], existem várias iniciativas para o desenvolvimento de recomendações para aplicações móveis acessíveis, porém as abordagens existentes são apenas sugestões e não uma lista concreta de requisitos funcionais. Segundo eles, ainda é necessário realizar uma avaliação dos requisitos de cada tipo de deficiência para entender melhor o impacto que elas têm sobre a usabilidade das aplicações móveis. Entretanto, os autores não consideraram como o tipo de implementação dos componentes de interface podem influenciar no cumprimento dos requisitos selecionados.

Em um estudo realizado por Shitkova *et al.* [20], buscou-se responder quais diretrizes de usabilidade devem ser consideradas para desenvolver um *site* ou aplicativo móvel acessível e até que ponto estas diretrizes são aplicáveis no processo de desenvolvimento. Os autores propuseram um catálogo com diretrizes de usabilidade e aplicaram no desenvolvimento de um aplicativo e de um *website* móvel. Entretanto, não foi possível avaliar a fundo a relevância, utilidade, suficiência e abrangência das diretrizes devido à natureza do método de pesquisa aplicado para criar o catálogo.

Park *et al.* [15] avaliaram como quatro participantes com deficiência visual realizavam determinadas tarefas nos seus dispositivos móveis. Os resultados apontaram para problemas graves de acessibilidade na digitação e em funções do leitor de tela VoiceOver. Como consequência, os autores propuseram um conjunto de 10 heurísticas para o desenvolvimento de aplicações móveis mais acessíveis. Barreiras provenientes do uso do recurso *touchscreen* levaram aos estudos de Kane *et al.* [12] e Piccolo *et al.* [17]. Os autores também definiram um conjunto de diretrizes focado no design de modelos de interações acessíveis para dispositivos móveis

Clegg-Vinell *et al.* [7] utilizaram as diretrizes do WCAG 2.0 como objeto de estudo de um outro artigo. Os autores contaram com a participação de pessoas com diferentes tipos de deficiências que avaliaram uma variedade de *sites* e aplicativos nativos em dispositivos móveis de diferentes plataformas. Os resultados deste estudo apontaram para a necessidade de uma nova abordagem de diretrizes através da combinação de elementos de acessibilidade, usabilidade e experiência do usuário, com o propósito de otimizar sua eficácia e reduzir o tempo gasto pelos desenvolvedores para abordá-las.

Em trabalhos recentes realizados pelo mesmo grupo de pesquisa dos autores deste estudo, Serra *et al.* [19] e Carvalho *et al.* [5] avaliaram a acessibilidade de diferentes aplicações governamentais do Brasil para dispositivos móveis. Os principais pontos dessas pesquisas, como as adaptações realizadas no WCAG 2.0 e os problemas de acessibilidade encontrados nos componentes de interface foram muito importantes para o desenvolvimento deste estudo.

METODOLOGIA

Este artigo descreve um estudo baseado na avaliação de uma amostra de componentes de interface Web em diferentes protótipos de aplicações móveis, de forma a coletar os problemas de acessibilidade encontrados nas inspeções de usabilidade das aplicações, utilizando uma adaptação dos critérios de sucesso do WCAG 2.0 para o contexto das aplicações móveis.

Amostra dos Componentes de Interface

Uma amostra de componentes de interface para Web foi selecionada a partir de uma categorização realizada por Freire [9] e Power *et al.* [18] em estudos anteriores. Freire categorizou os problemas encontrados pelos usuários com deficiência ao utilizarem *websites* em seis níveis: *content*, *delivery media*, *web page structure*, *website navigation*, *information architecture* e *underlying system characteristics.* Cada nível apresenta subcategorias que descrevem a natureza do problema encontrado pelos usuários. Em particular, neste trabalho, utilizamos dois níveis da classificação: *delivery media* (mídia) e *web page structure* (estrutura da página Web). Esses níveis se referem à camada da estrutura de uma página Web, apresentando 8 categorias de componentes de interface da Web equivalentes em aplicativos nativos, tais como textos, imagens, áudios, vídeos, multimídia e outros, permitindo que sejam implementados e testados.

Visando a realização das inspeções de acessibilidade, 30 componentes de interface Web foram selecionados para a amostra a partir das 8 categorias apresentadas na classificação anterior.

Os 30 componentes de interface foram selecionados a partir da uma investigação em documentações dos componentes padrões do HTML e do Android. Todos os 30 componentes da amostra são listados a seguir:

Mídia

- **Textos:** texto em parágrafo, texto em lista ordenada, texto em lista desordenada, texto em colunas e texto em citação.
- **Imagens:** imagem informativa, imagem decorativa, imagem funcional, imagem de texto, imagem complexa, grupo de imagens e mapas de imagem.
- **Áudio, vídeo e multimídia:** player de vídeo e de áudio.
- **Outros tipos de mídia:** notação matemática e fórmula química.

Estrutura da Página Web

- **Cabeçalhos:** título de seções.
- **Links:** link local, link externo e link em imagem.
- **Tabelas:** tabela com um cabeçalho, tabela com dois cabeçalhos, tabela com cabeçalhos irregulares, tabela com cabeçalhos de vários níveis e tabela com legenda e resumo.
- **Controles, formulários e funcionalidades:** caixa de seleção, botão de opção, botão, lista suspensa e campo de texto.

Implementação dos Componentes de Interface

Com o intuito de realizar as avaliações, foram criados protótipos de aplicações móveis contendo os componentes anteriormente mencionados em sua forma padrão. Esses aplicativos representam exemplos de aplicações com tais componentes e permitem que testes sejam realizados.

Cada um dos 30 componentes de interface foi implementado em três protótipos de aplicativos móveis. O primeiro protótipo móvel utilizou componentes padrões do Android Studio no desenvolvimento nativo. No segundo protótipo, foram incluídos componentes HTML foi empregada no Android Studio, gerando uma aplicação nativa com recursos Web. Finalmente, o último protótipo utilizou o *framework* Apache Cordova 6.3.1 para implementar os componentes na forma híbrida.

Durante a implementação do protótipo utilizando o desenvolvimento nativo sem o uso componentes Web, alguns componentes da amostra que possuem marcações apenas para a Web foram adaptados. Os componentes de "texto em lista ordenada" e "texto em lista desordenada" foram substituídos por um componente equivalente e padrão do Android Studio, o ListView; os diferentes tipos de marcações do componente "título de seções" não estão disponíveis no Android Studio e foram substituídos por representações visuais do componente TextView; os componentes "link local" e "link externo" também foram adaptados, pois o Android Studio não possui a mesma forma de marcação dos links como no HTML, que utiliza *tags* específicas; todas os componentes de tabelas foram desenvolvidos no Android Stuido através do componente GridView, diferentemente do HTML, que utiliza *tags* específicas para a marcação dos elementos do corpo da tabela; alguns componentes de imagens específicos para o HTML também foram ajustados para a implementação no Android Studio.

Na concepção do protótipo utilizando o desenvolvimento nativo com o uso de componentes Web não foram realizadas adaptações, pois neste tipo de desenvolvimento todos os componentes Web são implementados da mesma forma como no HTML e renderizados através de uma WebView. Todos os protótipos foram desenvolvidos para o sistema operacional Android 6.0.

Procedimento para Inspeção de Acessibilidade

Para realizar as inspeções de acessibilidade nos componentes de interface, optamos pelo método de revisão de diretrizes nos protótipos desenvolvidos.

Como mencionado anteriormente, não existem diretrizes consolidadas para a avaliação de aplicações móveis. Os guias de programação de acessibilidade da Apple para iOS e Google para Android fornecerem recomendações mais técnicas relacionadas à programação de componentes de interface específicos e não incluem problemas mais abrangentes presentes nos guias de acessibilidade amplamente aceitos, como as diretrizes de acessibilidade para a Web. Deste modo, diferentes diretrizes de acessibilidade foram analisadas, como a ISO 9241-171 [11] para a acessibilidade de *software* e a WCAG 2.0 [4] para a acessibilidade na Web. Tais opções não apresentaram recomendações específicas para a acessibilidade móvel.

Segundo a Web Accessibility Initiative (WAI) [23] que publicou o WCAG 2.0, as principais tecnologias presentes nos trabalhos do W3C suportam a acessibilidade, incluindo aquelas que são essenciais para a Web em dispositivos móveis. Além disso, a grande maioria dos padrões de interface do usuário de sistemas *desktops* são igualmente aplicáveis no celular, como textos, *hiperlinks*, tabelas, botões, menus e outros. Dito isto, optamos pela escolha do WCAG 2.0 para a inspeção de acessibilidade, após realizarmos adaptações para o contexto das aplicações para dispositivos móveis.

Realizamos uma análise de todos os 61 critérios de sucesso disponíveis no WCAG 2.0 e verificamos quais critérios não eram diretamente aplicáveis no contexto da acessibilidade das aplicações em dispositivos móveis a fim de realizar algumas adaptações para realizar as inspeções de acessibilidade. Os critérios de sucesso adaptados correspondiam à navegação por teclado, linguagens de marcação da Web, consistência do *layout*, redimensionamento do texto e outros.

Uma adaptação foi realizada nos critérios de sucesso 2.1.1 e 2.1.3, que dizem respeito à "Teclado: Todas as funcionalidades do conteúdo são operáveis através de uma interface de teclado sem a necessidade de qualquer espaço de tempo entre cada digitação individual", respectivamente com exceções a funções fundamentais e sem exceção. Ao utilizar recursos de Tecnologia Assistiva, como um leitor de tela, em aplicativos móveis, o usuário manipula os ícones, botões e outros itens através de gestos, tais como "deslizar para a direita" para mover para o próximo item na tela, "deslizar para a esquerda" para mover para o item anterior na tela e "tocar duas vezes" para selecionar o item em foco. Esses gestos simulam a tecla "tab" utilizada na navegação pelo teclado por usuários de leitores de tela em sistemas *desktops*.

Outros critérios de sucesso que consideram a utilização de tecnologias baseadas na Web para a implementação dos componentes, como HTML, também foram adaptados. Entre esses critérios, podemos citar o 4.1.1, que aborda questões de análise e validação do HTML. Para inspecionar os protótipos desenvolvidos, o avaliador deve considerar outras tecnologias, como o Java, utilizada no Android Studio para o desenvolvimento de aplicações para a plataforma Android, para verificar falhas de programação nos componentes de interface que podem causar problemas aos usuários de tecnologias assistivas.

Logo após a adaptação dos critérios de sucesso para o contexto dos dispositivos móveis, um avaliador especialista utilizou todos os 61 critérios de sucesso do WCAG 2.0 para auditar os 30 componentes de interface da amostra presentes nos 3 protótipos desenvolvidos. Para cada componente de interface inspecionado, o avaliador simulou o uso dos gestos básicos do TalkBack utilizados por pessoas com deficiência visual. Foram registrados os principais problemas encontrados e o número de violações de cada critério de sucesso.

As inspeções manuais foram realizadas utilizando o leitor de tela TalkBalk 5.1.0.12 em um *smartphone* Moto G 2ª Geração com o sistema operacional Android 6.0.

RESULTADOS E DISCUSSÃO

Detalhamento das Avaliações dos Componentes

Nesta seção apresentamos os resultados detalhados da avaliação de acessibilidade da amostra dos componentes de interface no TalkBack, considerando os 61 critérios de sucesso do WCAG 2.0. Os critérios foram classificados como "P" (Passou), nos casos que o componente cumpria todos os requisitos do critério de sucesso e era acessível através do leitor de tela, ou "F" (Falhou), quando não cumpria as recomendações do critério de sucesso.

Como vários critérios de sucesso se aplicam apenas a determinados componentes de interface, nas tabelas presentes nas seções a seguir apresentamos uma seleção dos principais critérios de sucesso para cada componente.

Textos

Os componentes textuais são estruturas importantes durante o desenvolvimento de uma aplicação, pois permitem a utilização de textos em parágrafos, listas, colunas e citações, e transmitem grande parte da informação para o usuário. Durante as inspeções de acessibilidade nos componentes da categoria "textos" na aplicação nativa (com uso de recursos Web) e híbrida, apenas o critério de sucesso "2.4.7 Foco Visível" foi violado para o componente "texto em colunas", de acordo com a Tabela 1. Neste caso, o TalkBack não determinou visualmente o componente que estava em foco, dificultando a interação de usuários que dependem de recursos de Tecnologia Assistiva, tais como pessoas com baixa visão que usam sua visão residual em conjunto com leitores de tela.

Na aplicação nativa (sem uso de recursos Web) os componentes de textos em listas ("texto em lista ordenada" e "texto em lista desordenada") apresentaram problemas com o TalkBack. Como mencionado anteriormente, o componente ListView, padrão do Android Studio, foi utilizado como uma alternativa para a implementação das listas no aplicativo nativo (sem uso de recursos Web). Diferentemente dos componentes de textos em listas no HTML, o TalkBack não relacionou os itens da lista com uma marca ou numeração, violando o critério de sucesso "1.3.1 Informações e Relações".

Segundo o WCAG 2.0, é importante facilitar aos usuários a separação de informações do primeiro plano com o plano de fundo, tornando a apresentação do conteúdo em sua forma padrão e facilitando a percepção de pessoas com deficiência. Nesse contexto, o redimensionamento de textos e a alteração da percepção visual de blocos de textos permitem personalizar o conteúdo de acordo com os requisitos do usuário. O Android fornece ferramentas como gestos de ampliação, texto grande, texto em alto contraste, inversão de cores e correção de cor que permitem apresentação das informações disponíveis em um formato alternativo. O componente "texto em parágrafo", em específico, cumpre os critérios de sucesso "1.4.4 Redimensionar Texto" e "1.4.8 Apresentação Visual".

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Texto em parágrafo	1.4.4 - P 1.4.8 - P 2.4.7 - P 3.1.4 - F	1.4.4 - P 1.4.8 - P 2.4.7 - P 3.1.4 - P	1.4.4 - P 1.4.8 - P 2.4.7 - P 3.1.4 - P
Texto em lista ordenada	1.3.1 - F	1.3.1 - P	1.3.1 - P
Texto em lista desordenada	1.3.1 - F	1.3.1 - P	1.3.1 - P
Texto em colunas	2.4.7 - P	2.4.7 - F	2.4.7 - F
Texto em citação	1.4.4 - P 1.4.8 - P	1.4.4 - P 1.4.8 - P	1.4.4 - P 1.4.8 - P

1.3.1 Informações e Relações, 1.4.4 Redimensionar Texto, 1.4.8 Apresentação Visual, 2.4.7 Foco Visível, 3.1.4 Abreviaturas

Tabela 1. Inspeção de acessibilidade nos componentes de textos

Em suma, apesar de algumas violações nos critérios de sucesso, os componentes de texto apresentaram ser mais acessíveis no desenvolvimento nativo (com uso de recursos Web) e híbrido, pois o TalkBack fornece informações e relações mais completas dos componentes. Os componentes de textos em listas mencionados nessa seção podem ser vistos na Figura 1.

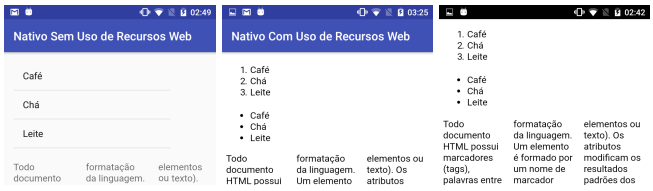

Figura 1. Componentes "texto em lista ordenada", "texto em lista desordenada" e "texto em colunas" implementados na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

Imagens

Imagens devem possuir alternativas de texto que descrevam sua informação ou função. Isso garante que as imagens podem ser utilizados por pessoas com várias deficiências. Os componentes de imagens presentes nos protótipos, apesar de apresentarem um resultado satisfatório na inspeção de acessibilidade, violaram alguns critérios de sucesso específicos. De acordo com a Tabela 2, o componente "grupo de imagens"

não passou no critério de sucesso “1.3.1 Informações e Relações” em nenhum dos 3 protótipos desenvolvidos. O TalkBack falhou em identificar o relacionamento das imagens do grupo.

Para o critério de sucesso “1.1.1 Conteúdo Não Textual”, foi verificada a existência de recursos em todos os componentes de imagens nas 3 aplicações pois toda alternativa textual das imagens foi informada pelo leitor de tela. O componente “imagem de texto”, por exemplo, que apresenta um texto em forma de imagem, conforme a Figura 2, oferece possibilidade de apresentação de texto alternativo para transmitir as informações. É importante que os desenvolvedores se atentem às marcações dos conteúdos não textuais para torná-los acessíveis pelos leitores de tela.

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Imagem informativa	1.1.1 - P	1.1.1 - P	1.1.1 - P
Imagem decorativa	1.1.1 - P	1.1.1 - P	1.1.1 - P
Imagem funcional	1.1.1 - P	1.1.1 - P	1.1.1 - P
Imagem de texto	1.1.1 - P	1.1.1 - P	1.1.1 - P
	1.4.5 - P	1.4.5 - P	1.4.5 - P
	1.4.9 - P	1.4.9 - P	1.4.9 - P
Imagem complexa	1.1.1 - P	1.1.1 - P	1.1.1 - P
	1.3.1 - F	1.3.1 - P	1.3.1 - P
Grupo de imagens	1.1.1 - P	1.1.1 - P	1.1.1 - P
	1.3.1 - F	1.3.1 - P	1.3.1 - P
Mapas de imagem	1.1.1 - P	1.1.1 - P	1.1.1 - P
	1.3.1 - F	1.3.1 - P	1.3.1 - P

1.1.1 Conteúdo Não Textual, 1.3.1 Informações e Relações, 1.4.5 Imagens de Texto, 1.4.9 Imagens de Texto (Sem Exceção)

Tabela 2. Inspeção de acessibilidade nos componentes de imagens

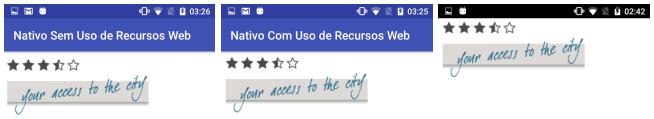

Figura 2. Componentes “grupo de imagens” e “imagem de texto” implementados na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

Áudio, Vídeo e Multimídia

O HTML5 introduziu duas novas *tags* de multimídia para exibir áudio e vídeo nas páginas Web, proporcionando acesso a diferentes formatos de arquivo para diferentes navegadores. Os componentes multimídia incluem imagens, músicas, sons, vídeos, registros, filmes e animações que podem ser incorporados nos diversos tipos de aplicativos móveis.

Figura 3. Componente “player de vídeo” implementado na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

Os componentes “player de vídeo” (Figura 3) e “player de áudio” presentes na amostra apresentaram algumas violações semelhantes nos critérios de sucesso do WCAG 2.0 para os 3 protótipos desenvolvidos. Dentre essas violações, podemos citar a falta de descrição no botão padrão para reproduzir o vídeo, problemas de exibição da legenda do vídeo e falta de recomendações consolidadas para inserir audiodescrição, legendas e língua de sinais nos componentes de áudio e vídeo em dispositivos móveis.

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Player de vídeo	1.2.1 - F	1.2.1 - F	1.2.1 - F
	1.2.2 - F	1.2.2 - F	1.2.2 - F
	1.2.3 - F	1.2.3 - P	1.2.3 - P
	4.1.2 - F	4.1.2 - P	4.1.2 - P
Player de áudio	1.2.1 - F	1.2.1 - F	1.2.1 - F
	1.2.2 - F	1.2.2 - F	1.2.2 - F
	1.2.6 - P	1.2.6 - P	1.2.6 - P

1.2.1 Apenas Áudio e Apenas Vídeo (Pré-gravado), 1.2.2 Legendas (Pré-gravadas), 1.2.3 Audiodescrição ou Mídia Alternativa (Pré-gravada), 1.2.6 Língua de Sinais (Pré-gravada), 4.1.2 Nome, Função, Valor

Tabela 3. Inspeção de acessibilidade nos componentes de áudio, vídeo e multimídia.

Conforme os resultados apresentados na Tabela 3, os componentes de interface dessa seção ainda não oferecem suporte ideal para o leitor de telas TalkBack. Os desenvolvedores que desejam utilizar componentes de áudio, vídeo e multimídia devem estar atentos que alguns usuários não terão acesso às informações transmitidas.

Outros Tipos de Mídia

Neste trabalho também investigamos a acessibilidade de outros tipos de mídia em aplicativos para dispositivos móveis, como “notação matemática” e “fórmula química”. Os meios atuais para o desenvolvimento de aplicações móveis, além dos leitores de tela, como o TalkBack, não fornecem apoio para fórmulas e equações mais complexas. Os mecanismos nativos da Web e do Android Studio não fornecem recursos para exibir visualmente as fórmulas e equações, como apresentado na Figura 4 a seguir.

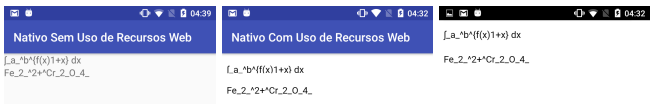

Figura 4. Componentes “notação matemática” e “fórmula química” implementados na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

A Tabela 4 mostra que os componentes de interface desta seção presentes nos 3 protótipos violaram o critério de sucesso “4.1.2 Nome, Função, Valor”. Esses componentes se mostraram totalmente incompatíveis com o leitor de tela TalkBack, impedindo que várias informações, como função, estado e valor, fossem sinterizadas em voz para os usuários cegos.

Neste caso, o fornecimento de um texto alternativo para gráficos de notações matemáticas e fórmulas químicas pode ser utilizado. Enquanto isso faz cumprir a norma de acessibilidade para o acesso aos conteúdos não textuais, esta prática não resulta em um nível de acesso que é verdadeiramente comparável ao de uma pessoa sem deficiência, que pode ver e interpretar

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Notação matemática	4.1.2 - F	4.1.2 - F	4.1.2 - F
Fórmula química	4.1.2 - F	4.1.2 - F	4.1.2 - F

4.1.2 Nome, Função, Valor

Tabela 4. Inspeção de acessibilidade nos componentes de outros tipos de mídia.

seu próprio significado. Diferentes aplicações podem ser utilizadas para contornar esse problema e tornar as equações mais acessíveis pelos leitores de tela. O uso da aplicações em XML incorporadas em aplicações, como o MathML (Mathematical Markup Language), permite que leitores de tela descrevam notações matemáticas e informem sua estrutura e conteúdo. Ainda não há suporte para leitura desse tipo de conteúdo por leitores de tela de dispositivos móveis em Português.

Desenvolvedores que desejam utilizar os componentes mencionados em suas aplicações devem buscar outras ferramentas que se integram com suas aplicações, tornando o conteúdo corretamente apresentável visualmente e compatíveis com os principais leitores de tela do mercado.

Cabeçalhos

Na linguagem de marcação HTML pode-se utilizar até 6 níveis de cabeçalhos para a definição dos títulos e subtítulos dos documentos. Um cabeçalho é definido pela delimitação do texto pela *tag* correspondente, aplicando o estilo negrito e um tamanho específico à fonte de acordo com o nível do tópico selecionado. O Android Studio não oferece marcações que identifiquem os níveis de cabeçalho, se limitando apenas na formatação visual dos títulos das seções, conforme a Figura 5. A ausência dessas *tags* nos aplicativos nativos que não utilizam recursos Web implica diretamente a capacidade dos leitores de tela de fornecer as informações das seções que ajudam o usuário na compreensão do conteúdo. Além disso, o TalkBack possui uma funcionalidade que permite a navegação entre os cabeçalhos de uma página, permitindo que os usuários tenham uma visão geral do que é exibido na tela e acelerem a navegação. Essa funcionalidade está disponível apenas para aplicações que utilizam componentes de interface Web.

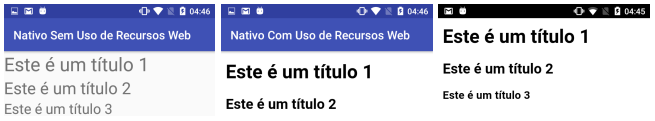

Figura 5. Componente "título das seções" implementado na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Título de seções	1.3.1 - F 2.4.2 - P 2.4.6 - P 2.4.10 - P 4.1.2 - F	1.3.1 - P 2.4.2 - F 2.4.6 - P 2.4.10 - P 4.1.2 - P	1.3.1 - P 2.4.2 - F 2.4.6 - P 2.4.10 - P 4.1.2 - P

1.3.1 Informações e Relações, 2.4.2 Página com Título, 2.4.6 Cabeçalhos e Rótulos, 2.4.10 Cabeçalhos da Sessão, 4.1.2 Nome, Função, Valor

Tabela 5. Inspeção de acessibilidade nos componentes de cabeçalhos

De acordo com os principais resultados das avaliações de acessibilidade dos "títulos das seções" exibidos na Tabela 5 anterior, as aplicações que utilizam recursos Web apresentaram um resultado superior ao aplicativo nativo que não utiliza recursos Web. Nesse cenário, os desenvolvedores que desejam criar aplicações que apresentam o conteúdo dividido em várias seções devem optar pelo uso de aplicações nativas (com uso de recursos Web) ou híbridas.

Links

Um link é uma conexão de um recurso da Web para outro. Em aplicações móveis, o link pode ser utilizado para acessar, por exemplo, uma imagem, um clipe de vídeo, uma trecho de áudio, um programa, uma nova página da aplicação, notificações, um documento HTML e outros.

O WCAG 2.0 possui dois critérios de sucesso específicos que tratam da finalidade dos links em contexto e apenas o link: 2.4.4 e 2.4.9. De acordo com esses critérios de sucesso, um mecanismo deve estar disponível para permitir que a finalidade de cada link seja identificada a partir apenas do texto do link (Nível AAA) ou a partir do texto do link em conjunto com seu respectivo contexto (Nível A). O TalkBack não encontrou problemas para identificar os "links locais", "links externos" e "links em imagens", proporcionando que pessoas com deficiência visual possam determinar o propósito de um link, explorando seu contexto. Um exemplo do componente "link externo" desenvolvido para os 3 protótipos de aplicações móveis é apresentado na Figura 6.

Figura 6. Componente "link externo" implementado na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

O critério de sucesso "4.1.2 Nome, Função, Valor" também possui recomendações específicas para links em HTML, sendo aplicáveis nos aplicativos nativo (com uso de recursos Web) e híbrido, conforme a Tabela 6. O critério de sucesso recomenda o uso de elementos de link com os atributos e valores adequados para que tecnologias assistivas consigam operar esses componentes. Existem diferentes meios para o desenvolvimento de links em aplicações nativas (sem o uso de recursos Web) no Android Studio, entretanto, várias alternativas não são acessíveis com o leitor de tela, implicando em links sem nomes, descrições e valores.

Tabelas

Tabelas são utilizadas para organizar dados através de uma relação lógica em grades. Para torná-las acessíveis é essencial utilizar *tags* que indiquem células de cabeçalhos e células de dados, e definam seu relacionamento. Leitores de tela utilizam essas informações para fornecer contextos aos usuários. A Figura 7 a seguir exemplifica o resultado de uma tabela com dois cabeçalhos utilizando as marcações adequadas.

O Android Studio não fornece componentes com *tags* para a marcação de células de cabeçalhos e células de dados, causando problemas para usuários que utilizam tecnologias assistivas, como um leitor de telas. Neste caso, o TalkBack encontrou

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Link local	2.4.4 - P	2.4.4 - P	2.4.4 - P
	2.4.9 - P	2.4.9 - P	2.4.9 - P
	4.1.2 - F	4.1.2 - P	4.1.2 - P
Link externo	2.4.4 - P	2.4.4 - P	2.4.4 - P
	2.4.9 - P	2.4.9 - P	2.4.9 - P
	4.1.2 - F	4.1.2 - P	4.1.2 - P
Link em imagem	2.4.4 - P	2.4.4 - P	2.4.4 - P
	2.4.9 - P	2.4.9 - P	2.4.9 - P
	4.1.2 - F	4.1.2 - P	4.1.2 - P

2.4.4 Finalidade do Link (Em Contexto), 2.4.9 Finalidade do Link (Apenas o Link), 4.1.2 Nome, Função, Valor

Tabela 6. Inspeção de acessibilidade nos componentes de links

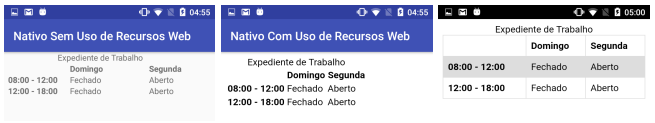

Figura 7. Componente "tabela com dois cabeçalhos" implementado na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

dificuldades para determinar qual coluna e qual linha estão associadas a cada uma das células da tabela. É importante que os desenvolvedores se preocupem em encontrar alternativas para determinar programaticamente as informações e os relacionamentos de tabelas para que sejam acessíveis a todos. No caso de tecnologias que não suportam relacionamentos programáticos, as relações podem ser apresentadas em texto. A Tabela 7 apresenta os 2 critérios de sucesso que falharam para os componentes de tabela na aplicação nativa que não utiliza recursos Web.

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Tabela com um cabeçalho	1.3.1 - F 1.3.2 - F	1.3.1 - P 1.3.2 - P	1.3.1 - P 1.3.2 - P
Tabela com dois cabeçalhos	1.3.1 - F 1.3.2 - F	1.3.1 - P 1.3.2 - P	1.3.1 - P 1.3.2 - P
Tabela com cabeçalhos irregulares	1.3.1 - F 1.3.2 - F	1.3.1 - F 1.3.2 - F	1.3.1 - P 1.3.2 - P
Tabela com cabeçalhos de vários níveis	1.3.1 - F 1.3.2 - F	1.3.1 - F 1.3.2 - F	1.3.1 - F 1.3.2 - F
Tabela com legenda e resumo	1.3.1 - F 1.3.2 - F	1.3.1 - P 1.3.2 - P	1.3.1 - P 1.3.2 - P

1.3.1 Informações e Relações, 1.3.2 Sequência com Significado

Tabela 7. Inspeção de acessibilidade nos componentes de tabelas

O TalkBack também encontrou barreiras para identificar as relações de tabelas mais complexas nas aplicações que utilizam componentes em HTML, como "tabela com cabeçalhos irregulares" e "tabela com cabeçalhos de vários níveis".

De acordo com os resultados apresentados, tabelas desenvolvidas com o auxílio de componentes Web fornecem vários recursos para o TalkBack que não estão presentes nos componentes de interface padrões do Android Studio, como o GridView. O uso de tabelas em aplicativos nativos que não utilizam recursos Web pode criar barreiras para usuários cegos que utilizam leitores de tela. Os desenvolvedores de aplicativos móveis sem recursos Web podem utilizar componentes alternativos e acessíveis para tabelas, desde que as informações sejam transmitidas da mesma forma para todas as pessoas, incluindo pessoas com deficiência.

Controles, Formulários e Funcionalidades

Controles, formulários e funcionalidades são utilizados para fornecer interações aos usuários em *websites* e aplicações móveis. Os 5 componentes desta categoria apresentaram resultados positivos durante as inspeções de acessibilidade. Eles não violaram nenhum dos 61 critérios de sucesso do WCAG 2.0 e mostraram ter possibilidade de implementação de recursos de acessibilidade para leitores de tela, que podem identificar e entender a funcionalidade dos controles de formulário se forem implementados com acessibilidade, ou seja, com a associação correta de rótulos e outros elementos estruturais.

No geral, é importante que os desenvolvedores se preocupem em rotular todos os controles de entrada. No HTML, por exemplo, os atributos *label* ou *title* devem ser utilizados para que usuários cegos possam ter confiança ao interagir com sua aplicação, seja em busca de informações ou para a inserção de dados pessoais importantes, como cartões de crédito ou números de telefone. A Tabela 8 a seguir lista alguns critérios de sucesso importantes que fornecem recomendações para tornar os componentes de interface de controles, formulários e funcionalidades acessíveis.

Componente de Interface	Nativo (sem Web)	Nativo (com Web)	Híbrido (com Web)
Caixa de seleção	3.3.2 - P	3.3.2 - P	3.3.2 - P
Botão de opção	3.3.2 - P	3.3.2 - P	3.3.2 - P
Botão	4.1.2 - P	4.1.2 - P	4.1.2 - P
Lista suspensa	3.3.2 - P	3.3.2 - P	3.3.2 - P
Campo de texto	3.2.2 - P 3.3.1 - P 3.3.2 - P 4.1.2 - P	3.2.2 - P 3.3.1 - P 3.3.2 - P 4.1.2 - P	3.2.2 - P 3.3.1 - P 3.3.2 - P 4.1.2 - P

3.2.2 Em Entrada, 3.3.1 Identificação do Erro, 3.3.2 Rótulos ou Instruções, 4.1.2 Nome, Função, Valor

Tabela 8. Inspeção de acessibilidade nos componentes de controles, formulários e funcionalidades.

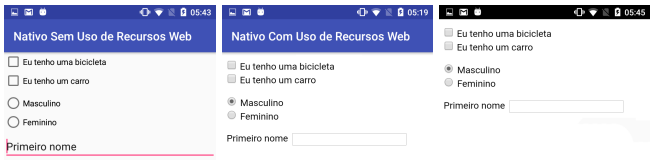

Figura 8. Componentes "botão de opção", "caixa de seleção" e "campo de texto" implementados na aplicação nativa (sem uso de recursos Web), aplicação nativa (com uso de recursos Web) e aplicação híbrida, respectivamente.

A Figura 8 apresenta um exemplo dos principais componentes de interface de controles, formulários e funcionalidades criados com as 3 técnicas de desenvolvimento móvel avaliadas neste trabalho.

Discussão

Os resultados das análises apresentaram pontos positivos para as aplicações desenvolvidas com componentes Web, tanto

aplicativos híbridos desenvolvidos com o *framework* Apache Cordova quanto aplicativos nativos com partes Web embutidas. No geral, os componentes de interface Web se mostraram mais acessíveis às pessoas com deficiência que utilizam o TalkBack. Componentes Web mais complexos, tais como texto em colunas, tabelas com vários cabeçalhos, áudio e vídeo, outros tipos de mídia e outros, não atenderam às funcionalidades esperadas de acordo com alguns critérios de sucesso do WCAG 2.0

No protótipo da aplicação que não utiliza componentes Web, encontramos um número relativamente maior de violações do que nos outros dois protótipos. Os componentes nativos do Android Studio violaram, em sua maioria, os critérios de sucesso "1.3.1 Informações e Relações" e "4.1.2 Nome, Função, Valor". Os problemas de acessibilidade encontrados nos componentes nativos do Android Studio podem implicar na falta de apresentação de suas informações e em uma fraca interpretação do conteúdo por uma ampla variedade de agentes de usuário, incluindo recursos de Tecnologia Assistiva.

Também é importante mencionar que todos os resultados das avaliações nos aplicativos nativos (com uso de recursos Web) e híbridos se mostraram iguais devido ao uso do WebView para renderizar os componentes da Web. Para o usuário, não existem diferenças na forma como o conteúdo dessas duas aplicações são apresentados. O aplicativo nativo fica limitado a poucos recursos da Web, sem nenhuma integração com os serviços nativos do Android Studio, enquanto a aplicação híbrida permite a utilização de todo o potencial do HTML, CSS e JavaScript, além de outras ferramentas, *frameworks* e serviços em nuvem.

Isso mostra que os aplicativos híbridos, além de serem uma das das estratégias de desenvolvimento móveis mais eficazes [14] em termos de portabilidade, também apresentam uma boa acessibilidade em seus componentes de interface para os usuários que utilizam o leitor de tela TalkBack no Android.

A falta de diretrizes bem definidas para a acessibilidade em aplicações móveis, sejam aquelas que utilizam ou não componentes da Web, ainda impõe barreiras para os desenvolvedores devido à falta de padronização dos componentes de interface que possuem suporte para os leitores de tela. Enquanto os leitores de tela ainda não fornecerem suporte para alguns componentes, como as tabelas em uma aplicação nativa sem o uso de recursos Web, o desenvolvedor deve buscar estratégias alternativas para que tecnologias assistivas consigam passar a informação para pessoas com deficiência.

CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou um estudo sobre a acessibilidade de 30 componentes de interface presentes em 3 formas de implementação de aplicativos móveis, por meio de 3 protótipos de aplicativos móveis que empregaram diferentes técnicas de desenvolvimento móvel.

Os resultados mostraram que os componentes de interface concebidas através aplicações nativas que utilizam recursos Web e híbridas são mais compatíveis com os critérios de sucesso do WCAG 2.0. O uso do *framwork* Apache Cordova no desenvolvimento de aplicações híbridas apresentou resultados positivos para a acessibilidade com o leitor de tela TalkBack. Além disso, o uso de método de desenvolvimento nativo com a utilização de recursos da Web, apesar de cumprir vários critérios de sucesso, deve ser evitado caso deseja-se desenvolver aplicações mais complexas.

No entanto, os desenvolvedores também podem optar pelo desenvolvimento de aplicações móveis nativas que empregam componentes de interface nativos do Android Studio. Durante as inspeções de acessibilidade, vários desses componentes apresentaram problemas comuns para a identificação de relações em tabelas, cabeçalhos e multimídias. Isso mostra que os desenvolvedores preocupados com a acessibilidade devem estar cientes das limitações atuais dessas aplicações e devem propor meios alternativos para evitar que usuários sejam impedidos de acessarem determinado conteúdo.

Outras características, como os pontos positivos e negativos do desenvolvimento das aplicações híbridas em comparação com os aplicativos nativos, não devem ser descartadas. Como apontado por Bosnic *et al*. [1], o desenvolvimento de aplicações móveis híbridas e nativas possuem vantagens particulares. Enquanto o desenvolvimento híbrido permite gerar aplicativos para múltiplas plataformas a partir de uma única base de código, os aplicativos nativos são mais complexos e possuem desempenho superior.

Embora existam várias propostas de recomendações para tornar o conteúdo das aplicações móveis acessíveis, ainda é muito importante definir e consolidar diretrizes de acessibilidade para dispositivos móveis. Essas diretrizes precisam se capazes de levar em conta as capacidades específicas de cada tipo de usuário e as diferentes tecnologias utilizadas no desenvolvimento. Ainda que vários estudos utilizem critérios de sucesso adaptados, Costa *et al.* [8] alerta que avaliações de acessibilidade devem considerar técnicas focadas no objeto de estudo, pois os resultados podem ser incoerentes quando as mesmas diretrizes são aplicadas em plataformas diferentes.

Como trabalho futuro, pretendemos selecionar uma amostra de aplicativos móveis reais que empreguem as técnicas de desenvolvimento abordadas e realizar testes de usabilidade com usuários com diferentes tipos de deficiências. Em seguida, identificaremos os principais problemas que esses usuários encontram com aplicações móveis nativas e híbridas para fornecer recomendações de design baseadas em padrões de uso e problemas comumente encontrados pelos usuários, contribuindo para a consolidação de diretrizes e técnicas para tornar aplicações móveis mais acessíveis.

AGRADECIMENTOS

Agradecemos ao CNPq (proc. 448521/2014-8) e à FAPEMIG pelo apoio financeiro.

REFERENCES

1. Stefan Bosnic, Ištvan Papp, and Sebastian Novak. 2016. The Development of Hybrid Mobile Applications with Apache Cordova. In *Telecommunications Forum (TELFOR), 2016 24th*. IEEE, 1–4.

2. Brasil. 2014. eMAG - Modelo de Acessibilidade em Governo Eletrônico - Versão 3.1. (2014). Disponível

online em http://emag.governoeletronico.gov.br/. Último acesso em 5 de maio de 2017.

3. Raluca Budiu. 2013. Mobile: Native Apps, Web Apps, and Hybrid Apps. (2013). Disponível online em https://www.nngroup.com/articles/mobile-native-apps/. Último acesso em 5 de maio de 2017. NN Group.
4. Ben Caldwell, Michael Cooper, Loretta Guarino Reid, and Gregg Vanderheiden. 2008. Web Content Accessibility Guidelines (WCAG) 2.0. (2008). Disponível online em https://www.w3.org/TR/WCAG20/. Último acesso em 5 de maio de 2017. World Wide Web Consortium (W3C).
5. Lucas Pedroso Carvalho, Bruno Piovesan Melchiori Peruzza, Flávia Santos, Lucas Pereira Ferreira, and André Pimenta Freire. 2016. Accessible smart cities? Inspecting the accessibility of Brazilian municipalities' mobile applications. In *Proc. of the XV Brazilian Symposium on Human Factors in Computer Systems*.
6. Sarah Chiti and Barbara Leporini. 2012. Accessibility of Android-Based Mobile Devices: A Prototype to Investigate Interaction with Blind Users. In *Proce. of the 13th International Conference on Computers Helping People with Special Needs - Volume Part II (ICCHP'12)*. Springer-Verlag, Heidelberg, 607–614.
7. Raphael Clegg-Vinell, Christopher Bailey, and Voula Gkatzidou. 2014. Investigating the Appropriateness and Relevance of Mobile Web Accessibility Guidelines. In *Proc. of the 11th Web for All Conference (W4A '14)*. Article 38, 4 pages.
8. Daniel Costa, Luís Carriço, and Carlos Duarte. 2015. The Differences in Accessibility of TV and Desktop Web Applications From the Perspective of Automated Evaluation. *Procedia Computer Science* 67 (2015), 388 – 396.
9. André Pimenta Freire. 2012. *Disabled People and the Web: User-Based Measurement of Accessibility*. Ph.D. Dissertation. University of York, Department of Computer Science.
10. ISO. 1998. ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) - Part 11: Guidance on Usability. (1998).
11. ISO. 2008. ISO 9241-171: Ergonomics of Human-System Interaction - Part 171: Guidance on Software Accessibility. (2008).
12. Shaun K. Kane, Jacob O. Wobbrock, and Richard E. Ladner. 2011. Usable Gestures for Blind People: Understanding Preference and Performance. In *Proce. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 413–422.
13. Barbara Leporini, Maria Claudia Buzzi, and Marina Buzzi. 2012. Interacting with Mobile Devices via VoiceOver: Usability and Accessibility Issues. In *Proc. of the 24th Australian Computer-Human Interaction Conference (OzCHI '12)*. 339–348.
14. Ivano Malavolta, Stefano Ruberto, Tommaso Soru, and Valerio Terragni. 2015. Hybrid Mobile Apps in the Google Play Store: An Exploratory Investigation. In *Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft '15)*. 56–59.
15. Kyudong Park, Taedong Goh, and Hyo-Jeong So. 2014. Toward Accessible Mobile Application Design: Developing Mobile Application Accessibility Guidelines for People with Visual Impairment. In *Proc. of HCI Korea (HCIK '15)*. 31–38.
16. Kim Patch, Jeanne Spellman, and Kathy Wahlbin. 2015. Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile. (2015). Disponível online em https://www.w3.org/TR/mobile-accessibility-mapping/. Último acesso em 5 de maio de 2017.
17. Lara Schibelsky G. Piccolo, Ewerton M. de Menezes, and Bruno de C. Buccolo. 2011. Developing an Accessible Interaction Model for Touch Screen Mobile Devices: Preliminary Results. In *Proc. of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction (IHC+CLIHC '11)*. 222–226.
18. Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines Are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 433–442.
19. Leandro Coelho Serra, Lucas Pedroso Carvalho, Lucas Pereira Ferreira, Jorge Belimar Silva Vaz, and André Pimenta Freire. 2015. Accessibility Evaluation of E-Government Mobile Applications in Brazil. *Procedia Computer Science* 67 (2015), 348 – 357.
20. Maria Shitkova, Justus Holler, Tobias Heide, Nico Clever, and Jörg Becker. 2015. Towards Usability Guidelines for Mobile Websites and Applications. In *Wirtschaftsinformatik*. 1603–1617.
21. Clauirton Siebra, Tatiana Gouveia, Jefte Macedo, Walter Correia, Marcelo Penha, Marcelo Anjos, Fabiana Florentin, Fabio Q. B. Silva, and Andre L. M. Santos. 2016. Observation Based Analysis on the Use of Mobile Applications for Visually Impaired Users. In *Proc. of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '16)*. 807–814.
22. Statista. 2016. Market Share Held by Mobile Operating Systems in Brazil from January 2012 to December 2016. (2016). Disponível online em https://www.statista.com/statistics/262167/market-share-held-by-mobile-operating-systems-in-brazil/. Último acesso em 4 de fevereiro de 2017.
23. W3C. 2008. Web Accessibility Initiative (WAI). (2008). Disponível online em https://www.w3.org/WAI/. Último acesso em 5 de maio de 2017.