UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

AALT: um *framework* com soluções práticas para melhorar a acessibilidade em aplicativos Android

Anderson Canale Garcia

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Anderson Canale Garcia

AALT: um *framework* com soluções práticas para melhorar a acessibilidade em aplicativos Android

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Renata Pontin de Mattos Fortes
Co-Orientadora: Profa. Dra. Kamila Rios da Hora Rodrigues

USP – São Carlos
Outubro de 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Garcia, Anderson Canale
G216a AALT: um *framework* com soluções práticas para melhorar a acessibilidade em aplicativos Android / Anderson Canale Garcia; orientadora Renata Pontin de Mattos Fortes; coorientadora Kamila Rios da Hora Rodrigues. -- São Carlos, 2023.
178 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Acessibilidade. 2. Desenvolvimento Mobile. 3. Testes automatizados. 4. AALT. 5. AATK. I. Fortes, Renata Pontin de Mattos, orient. II. Rodrigues, Kamila Rios da Hora, coorient. III. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Anderson Canale Garcia

AALT: a framework with practical solutions to enhance accessibility in Android applications

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Renata Pontin de Mattos Fortes
Co-advisor: Profa. Dra. Kamila Rios da Hora Rodrigues

USP – São Carlos
October 2023

À Professora Renata, por sua vida e por seu trabalho especialmente dedicados à acessibilidade digital.

AGRADECIMENTOS

Agradeço primeiramente a Deus, princípio de tudo e razão de todas as coisas.

À querida Professora Renata, por não ter desistido de mim ao longo desses 18 anos. Ninguém, como ela, conhece tão bem o que vivi por este mestrado. Queria poder ser para você agora, Re, o suporte que foi para mim nos momentos mais difíceis.

À Professora Kamila, por assumir a responsabilidade e, principalmente, por me fazer enxergar o valor da minha pesquisa, nos momentos de insegurança.

À minha esposa, meu amor, Giselle, que ressignificou tudo. Também por me ensinar o poder da oração ao Espírito Santo pelo dom da sabedoria, o que tanto me ajudou. E claro, por encher de vida a minha vida, com nossos 2 + 1 + 6 tão amados filhos, humanos, canino e felinos.

À minha família, amparo emocional para todas as horas.

A três pessoas que incentivaram especialmente a retomar esse sonho: Professora Silvana, a Sil, Professor Chu e Silvana Wick.

Às Professoras Graça e Solange, pela presteza, solidariedade e incentivo.

À Professora Cibele e ao Diego, pela amizade e ajuda com a análise de dados.

Ao Professor Delamaro, por gentilmente ceder espaço em suas turmas e colaborar fundamentalmente para os experimentos deste trabalho.

Aos colegas de pós, especialmente a Sandra, a Flávia e a Lianna, pelo suporte na reta final.

Ao Edmar, pela compreensão e flexibilizações.

E a tantas outras pessoas que eu poderia e deveria nomear, mas certamente não seria capaz de fazê-lo sem cometer injustiças. Em uma história de tantos anos, com tantas mudanças, foram muitas pessoas que, de algum modo, incentivaram, acompanharam ou fizeram concessões. Guardo todos vocês com carinho. Eu não esqueço.

“Amar e mudar as coisas me interessa mais”
(Belchior)

RESUMO

GARCIA, A. C. **AALT: um *framework* com soluções práticas para melhorar a acessibilidade em aplicativos Android**. 2023. 178 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

À medida que aplicativos móveis assumem um papel cada vez mais significativo na vida das pessoas, assegurar que esses aplicativos sejam acessíveis a todos, incluindo pessoas com deficiência, é uma questão primordial para a inclusão digital. Apesar da existência de recomendações e diretrizes de acessibilidade bem estabelecidas, desenvolvedores ainda encontram desafios para considerar a acessibilidade durante o desenvolvimento de aplicativos para *mobile*, e muitos dos aplicativos mais populares ainda apresentam barreiras de acesso para pessoas com deficiência. Muitas dessas barreiras de acessibilidade poderiam ser identificadas por ferramentas automatizadas de teste, mas essas ferramentas não são amplamente utilizadas ou conhecidas. O objetivo desta pesquisa é identificar e reunir soluções práticas que possam auxiliar desenvolvedores a detectar e resolver problemas de acessibilidade em aplicativos Android nativos. Para isso, são realizadas uma revisão da literatura sobre os problemas de acessibilidade mais recorrentes, especialmente para pessoas com deficiência visual, e um mapeamento sistemático da literatura, que explora técnicas de desenvolvimento que considerem a inclusão de requisitos de acessibilidade em aplicativos móveis durante o processo de desenvolvimento. Também são revisadas as ferramentas de teste de acessibilidade disponíveis para Android. A principal contribuição desta pesquisa é o *Android Accessibility Learning and Testing* (AALT) - um *framework* composto por treinamentos, recursos e ferramentas para promover a acessibilidade em aplicativos Android nativos. Entre os artefatos produzidos se destacam: um conjunto de requisitos de acessibilidade, definidos a partir das recomendações da literatura, e reescritos como declarações testáveis; o *Automated Accessibility Testing Kit* (AATK) - um *kit* de testes de acessibilidade automatizados para aplicativos Android nativos; e uma coleção de treinamentos, em formato de *codelabs*, para capacitar desenvolvedores e disseminar informações sobre a acessibilidade digital em aplicativos móveis. Avaliações de usabilidade do AATK foram realizadas com estudantes de graduação de disciplinas de Teste de Software. Os resultados indicam uma maior aceitação entre os estudantes nos semestres mais avançados. A pesquisa ressalta a necessidade de atribuição de responsabilidades, formação e conscientização sobre acessibilidade digital, tanto em ambientes acadêmicos quanto industriais. O AALT e seus artefatos contribuem para esta agenda, auxiliando desenvolvedores de aplicativos Android a aprimorar a acessibilidade, impulsionando a inclusão digital.

Palavras-chave: Acessibilidade, Desenvolvimento *Mobile*, Testes automatizados, AALT, AATK.

ABSTRACT

GARCIA, A. C. **AALT: a framework with practical solutions to enhance accessibility in Android applications**. 2023. 178 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

As mobile apps increasingly become essential in people's lives, it is crucial for digital inclusion to make sure these apps are accessible to everyone, including those with disabilities. Despite the existence of well-established accessibility guidelines and recommendations, developers continue to face challenges in considering accessibility during mobile app development, and many popular apps still present accessibility barriers. Many of these accessibility barriers could be identified by automated testing tools, but these tools are not widely used or known. This research aimed to identify and compile practical solutions to assist developers in identifying and addressing accessibility issues in native Android apps. To achieve this, three key research activities were conducted: a literature review on the most recurrent accessibility issues, especially for visually impaired individuals; a systematic mapping study to explore development techniques that incorporate accessibility requirements into mobile apps during the development process; and a review of the existing Android accessibility testing tools. The main contribution of this research was the Android Accessibility Learning and Testing (AALT) - a framework composed of training, resources, and tools to promote accessibility in native Android apps. The produced artifacts include: a set of accessibility requirements, derived from literature recommendations, and rewritten as testable statements; the Automated Accessibility Testing Kit (AATK) - a testing kit for native Android apps; and a collection of training sessions in codelab format, designed to empower developers and spread information about digital accessibility in mobile apps. The initial usability evaluations of AATK were carried out with undergraduate students enrolled in Software Testing courses. The results indicate greater acceptance among students in the final semesters. The research highlights the need for responsibility assignment, training, and awareness about digital accessibility in both academic and industrial environments. The AALT and its artifacts contribute to this agenda, aiding Android app developers in enhancing accessibility, thereby driving digital inclusion.

Keywords: Accessibility, Mobile Development, Automated Tests, AALT, AATK.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE CÓDIGOS-FONTE

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AALT	*Android Accessibility Learning and Testing*
AATK	*Automated Accessibility Testing Kit*
AOP	*Aspect-Oriented Programming*
API	*Application Programming Interface*
ASD	*Agile Software Development*
ATF	*Accessibility Test Framework*
ATTD	*Accessibility Testing Tool for Designers*
BBC	*British Broadcasting Corporation*
BCC	Bacharelado em Ciências da Computação
BSI	Bacharelado em Sistemas de Informação
CBD	*Component-based Development*
CD	Entrega Contínua
CEP	Comitê de Ética em Pesquisa em Seres Humanos
CI	Integração Contínua
CSS	*Cascading Style Sheets*
CSV	*Comma-separated values*
HTML	*HyperText Markup Language*
IEM	*Inline error message*
IHC	Interação Humano-Computador
JVM	*Java Virtual Machine*
maR	*mobile accessibility recommendations*
MDD	*Model-driven Development*
MWABP	*Mobile Web Application Best Practices*
MWBP	*Mobile Web Best Practices*
N.D.	Não Definido
N.I.	Não Informada
RAD	*Rapid Application Development*
RUP	*Rational Unified Process*
SIDI	Samsung Instituto de Desenvolvimento para Informática
T.A.	Tecnologia Assistiva
TCLE	Termo de Consentimento Livre e Esclarecido

TCLE	Termo de Consentimento Livre e Esclarecido
TDD	*Test-Driven Development*
TDD	*Test-driven Development*
TI	Tecnologia da Informação
UCD	*User-Centered Design*
UI	*User Interface*
UIDPs	*User Interface Design Patterns*
USP	Universidade de São Paulo
UX	*User Experience*
UX	*User eXperience*
W3C	*World Wide Web Consortium*
WAI	*Web Accessibility Initiative*
WCAG	*Web Content Accessibility Guidelines*
XML	*E**x**tensible **M**arkup **L**anguage*
XP	*Extreme Programming*

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Todas as pessoas têm o direito de serem incluídas nas diversas atividades e espaços da sociedade, independentemente de deficiências, localização geográfica, barreiras de linguagem, ou outros fatores, conforme destacado na "Lei Brasileira de Inclusão", Lei nº 13.146 (BRASIL, 2015).

Com este propósito, a acessibilidade se apresenta como um requisito essencial para garantir acesso a todas as pessoas, eliminando barreiras ao acesso e reduzindo a exclusão. Pesquisas têm explorado implementações de sistemas computacionais para melhorar a vida das pessoas com deficiência (CARVALHO *et al.*, 2016; GOMES *et al.*, 2018; VENDOME *et al.*, 2019; ALSHAYBAN; AHMED; MALEK, 2020; RIEGER *et al.*, 2020). Proporcionar a essas pessoas o acesso aos recursos tecnológicos é muito importante para garantir que elas também possam executar suas tarefas e usar os sistemas da mesma maneira que os demais usuários (MANKOFF *et al.*, 2019). No desenvolvimento de software, a acessibilidade é um aspecto vital para alcançar uma grande base de usuários para um produto de software e, em alguns países, é também uma exigência legal (PAIVA; FREIRE; FORTES, 2021).

No contexto da Web, a acessibilidade visa possibilitar que qualquer pessoa possa entender e interagir com os conteúdos disponíveis, independente de deficiências ou outros fatores que possam restringir o acesso (PETRIE; SAVVA; POWER, 2015). Para Tim Berners Lee, inventor da Web e diretor do *World Wide Web Consortium* (W3C), "o poder da Web está em sua universalidade. Ser acessada por todos, independente de deficiência, é um aspecto essencial". Com o crescimento do uso de dispositivos móveis para o acesso à Web, é muito importante certificar-se de que os aplicativos destes dispositivos estejam acessíveis a usuários com deficiência (CLEGG-VINELL; BAILEY; GKATZIDOU, 2014; HENRY; ABOU-ZAHRA; BREWER, 2014).

Em dispositivos móveis, os aplicativos nativos, ou seja, aqueles desenvolvidos para uma plataforma específica, apresentam melhor desempenho (BOSNIC; PAPP; NOVAK, 2017) e possibilitam melhor uso de recursos do dispositivo. Porém, o desenvolvimento deste tipo de

aplicativo apresenta desafios específicos (AHMAD *et al.*, 2018; BI *et al.*, 2022). No que se refere à acessibilidade, vários componentes de interface nativos apresentam problemas comuns e demandam meios alternativos para permitir que usuários com deficiência tenham acesso a determinado conteúdo (CARVALHO; FREIRE, 2017).

Nesse sentido, os desenvolvedores de software desempenham um papel crucial na promoção da acessibilidade digital, mas muitos deles enfrentam desafios ao criar aplicativos acessíveis. Entre os argumentos citados para o não emprego da acessibilidade se destacam, de acordo com Leite *et al.* (2021) e Gomes, Rios e Rodrigues (2020): a falta de conhecimento sobre o tema, tempo do projeto, ausência de treinamento, não definição de requisitos claros e o desinteresse das diferentes partes interessadas. Além disso, segundo Bi *et al.* (2022), recursos limitados ou inadequados, incluindo métodos de desenvolvimento e ferramentas de suporte, bem como a escassez de especialistas, também podem afetar a implementação da acessibilidade digital.

Boa parte dos esforços da literatura tem sido direcionada para a elaboração e organização de *guidelines* (diretrizes) e recomendações de acessibilidade, como as *guidelines* de fabricantes de dispositivos móveis, por exemplo: Google,[1] Apple[2] e Samsung.[3] Além disso, *Frameworks* e modelos de arquitetura podem guiar e acelerar o desenvolvimento de aplicativos acessíveis (OLIVEIRA *et al.*, 2019). Ainda assim, problemas de acessibilidade são encontrados entre os aplicativos mais populares disponíveis em lojas de aplicativos *mobile* (YAN; RAMACHANDRAN, 2019; OLIVEIRA *et al.*, 2023).

Recursos insuficientes e inadequados, tais como métodos de desenvolvimento e ferramentas de apoio (BI *et al.*, 2022), são alguns fatores que podem influenciar negativamente a implementação da acessibilidade digital. Oliveira e França (2019) e Cruz, Abreu e Lo (2019) sugerem que a adoção de práticas ágeis de desenvolvimento pode aumentar o engajamento dos desenvolvedores. Propostas integrando a usabilidade com métodos do desenvolvimento ágil, como em Losada *et al.* (2013), Hess *et al.* (2013) e Salman e Deraman (2022), têm sido consideradas. Wolkerstorfer *et al.* (2008), por exemplo, combinam ciclos curtos de entrega e a definição da usabilidade de aplicativos com desenvolvimento orientado a testes.

Uma forma de permitir testes constantes e contínuos em todo o software é usando testes automatizados (ANICHE, 2014). Testes automatizados beneficiam o desenvolvimento de software, melhorando a codificação, eficiência, qualidade e cobertura, ao mesmo tempo, em que permitem que os desenvolvedores se concentrem em tarefas essenciais (CRISPIN; GREGORY, 2009; DELAMARO; MALDONADO; JINO, 2016).

Para dispositivos Android, que respondem por 70% do mercado *mobile* segundo dados de março de 2023,[4] a plataforma disponibiliza um conjunto de ferramentas que podem ser

[1] <https://developer.android.com/guide/topics/ui/accessibility/>
[2] <https://developer.apple.com/accessibility/ios/>
[3] <http://www.sidi.org.br/guiadeacessibilidade/>
[4] <https://gs.statcounter.com/os-market-share/mobile/worldwide>

utilizadas nas diferentes estratégias de testes de acessibilidade (GOOGLE, 2023c). Além disso, outras ferramentas têm sido propostas para a automatização de testes de acessibilidade nesses dispositivos (ELER *et al.*, 2018; VONTELL, 2019; OLIVEIRA; FRANÇA, 2019), estendendo o conjunto de problemas de acessibilidade que podem ser identificados. Porém, a maioria dos testes realizados por essas ferramentas acontecem principalmente após o desenvolvimento do software, sendo essencialmente testes de Interface do Usuário (UI). Testes de UI requerem a utilização de dispositivos físicos ou emulados para serem executados, o que pode resultar em um *feedback* mais distante para os desenvolvedores. (CRISPIN; GREGORY, 2009; GOOGLE, 2023b).

Estudos recentes da literatura (YAN; RAMACHANDRAN, 2019; ALSHAYBAN; AHMED; MALEK, 2020; OLIVEIRA *et al.*, 2023) indicam que os problemas de acessibilidade mais frequentemente reportados em aplicativos Android concentram-se nos elementos de UI mais usados, e são problemas que poderiam ser identificados pelas ferramentas existentes.

1.1 Motivação

Atualmente, aplicativos Android nativos ainda apresentam barreiras que dificultam a interação dos usuários, o que ressalta a necessidade contínua de melhorar a acessibilidade desses aplicativos. Existem instrumentos disponíveis, como guias, recomendações e ferramentas de testes para auxiliar os desenvolvedores a eliminarem essas barreiras em seus aplicativos. No entanto, observa-se que, apesar da disponibilidade dessas soluções, muitos aplicativos continuam apresentando problemas recorrentes, o que indica que essas ferramentas podem não estar sendo adequadamente aplicadas.

Esse contexto impulsiona estudos que explorem o potencial de abordagens teóricas e práticas para avançar o conhecimento e buscar soluções no desenvolvimento de aplicativos móveis acessíveis, visando aprimorar o estado da arte nessa área.

De modo particular, esta pesquisa buscou alinhar o repertório teórico-científico oriundo do grupo de pesquisa em Interação Humano-Computador (IHC) do ICMC-USP com a perspectiva prática da atuação como desenvolvedor de software deste autor. Espera-se, assim, que este trabalho auxilie à compreensão do panorama atual sobre acessibilidade em dispositivos móveis, destacando o que existe na literatura e informando caminhos para fomentar o desenvolvimento de aplicações acessíveis por parte dos desenvolvedores.

1.2 Objetivo

Extensos e variados conjuntos de diretrizes têm sido disponibilizados para auxiliar no desenvolvimento de aplicativos móveis acessíveis. No entanto, desenvolvedores ainda encontram obstáculos para conhecer e compreender essas orientações, bem como para entender como aplicá-las em

seus projetos de desenvolvimento *mobile* (LEITE *et al.*, 2021; GOMES; RIOS; RODRIGUES, 2020; BI *et al.*, 2022; SWALLOW; PETRIE; POWER, 2016).

Desse modo, e considerando a predominância do sistema Android em dispositivos móveis e a especificidade das práticas e ferramentas de desenvolvimento para aplicativos nativos, o objetivo geral desta pesquisa de mestrado foi **elaborar estratégias para apoiar os desenvolvedores no desenvolvimento de aplicativos acessíveis nativos para Android a partir do uso de um *framework*** que auxilia durante a etapa de Construção de Software com orientação prática sobre as soluções tecnológicas relacionadas à acessibilidade.

Para atender a este objetivo, definiram-se os seguintes objetivos específicos:

- Realizar uma revisão da literatura, incluindo:
  - Identificar os principais problemas de acessibilidade encontrados em aplicativos móveis, e as recomendações de acessibilidade para evitá-los.
  - Investigar o que existe atualmente para apoiar desenvolvedores no desenvolvimento de aplicativos móveis acessíveis.
- Investigar problemas nas práticas dos desenvolvedores relacionadas aos requisitos de acessibilidade, pela apresentação das estratégias elaboradas.
- Aperfeiçoar as estratégias criadas com foco na construção de um *framework* de apoio ao desenvolvimento de aplicativos Android acessíveis.
- Avaliar as estratégias do *framework* proposto.

1.3 Metodologia de pesquisa

O diagrama apresentado na Figura 1 ilustra as etapas de desenvolvimento de pesquisa percorridas para cada objetivo específico descrito na subseção anterior.

A primeira etapa realizada foi a revisão da literatura, visando compreender os principais problemas enfrentados por pessoas com deficiência visual ao utilizarem aplicativos para Android, bem como as recomendações e ferramentas disponíveis atualmente para orientar os desenvolvedores desses aplicativos. Além disso, dentro do escopo da revisão da literatura, investigou-se como a acessibilidade tem sido considerada nos processos de desenvolvimento móvel. Isso foi realizado principalmente por meio de um mapeamento sistemático da literatura, que posteriormente foi complementado com uma revisão de estudos envolvendo desenvolvedores.

Na segunda etapa da pesquisa, os resultados da revisão da literatura foram consolidados em um conjunto de estratégias com o propósito de envolver os desenvolvedores no processo de incorporação de acessibilidade em aplicativos móveis. A exploração e seleção dessas estratégias foram direcionadas para responder a três perguntas cruciais relacionadas aos testes de acessibilidade: 1) O que deve ser testado? 2) Quais ferramentas podem ser utilizadas e em que momento? e, 3) Quem deve ser responsável pela realização dos testes?

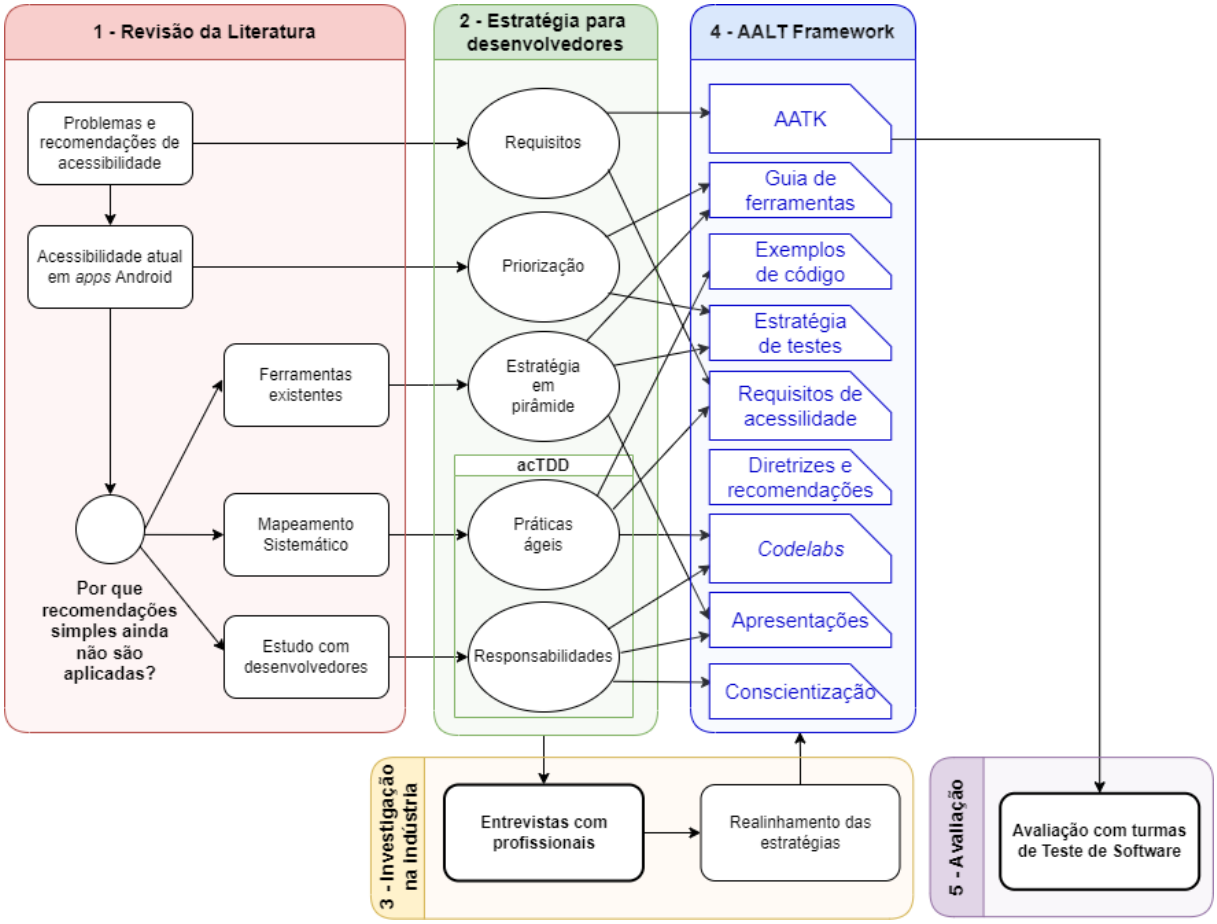


Figura 1 – Etapas de desenvolvimento da pesquisa.

Fonte: Elaborada pelo autor.

Na terceira etapa da pesquisa, com base nas estratégias delineadas na segunda etapa, foram realizadas entrevistas com profissionais da indústria de desenvolvimento de aplicativos móveis. O principal propósito deste estudo foi a apresentação e coleta de *feedbacks* sobre as estratégias previamente estabelecidas. Além disso, buscou-se compreender como a acessibilidade é abordada nos projetos de desenvolvimento móvel em que esses profissionais estão envolvidos.

A partir dos resultados e discussões deste estudo da terceira etapa, as estratégias foram realinhadas, dando origem, na quarta etapa, à principal contribuição desta dissertação de mestrado: o ***Framework* de Aprendizado e Testes de Acessibilidade para Android** (**AALT** - ***Android Accessibility Learning and Testing Framework***) – um *framework* composto de artefatos teóricos e práticos, resultantes da experiência e das lições aprendidas pela experimentação de soluções e estratégias identificadas nas pesquisas realizadas para atingir os objetivos definidos na Seção 1.2.

O termo *framework*[5] é usado aqui no seu sentido mais amplo, referindo-se a uma estrutura composta por conceitos, orientações, treinamentos, técnicas e ferramentas sobre acessibilidade,

[5] O termo framework é definido pelo dicionário Cambridge como *"1. A supporting structure around which something can be built; 2. A system of rules, ideas or beliefs that is used to plan or decide something."*

e contempla os diferentes papéis de uma equipe de desenvolvimento ágil de software: *designers*, desenvolvedores, testadores, gestores e clientes. Isso favorece a delimitação das responsabilidades de cada ator no desenvolvimento de aplicativos móveis acessíveis.

Na quinta e última etapa desta pesquisa, foi realizada a avaliação de um dos artefatos originais do AALT: um kit de testes de acessibilidade automatizados, com foco em testes locais, que não requerem o uso de dispositivos físicos ou emulados para serem executados. Esta avaliação foi conduzida com alunos de disciplinas de Teste de Software da Universidade de São Paulo (USP). Assim como os estudos com profissionais na terceira etapa, essa avaliação foi aprovada pelo Comitê de Ética em Pesquisa em Seres Humanos (CEP) da UFSCar sob o número CAAE 71178423.7.0000.5504. Em ambas as etapas, os participantes foram plenamente informados sobre todos os riscos e benefícios da sua participação, e que eles poderiam recusar ou interromper a participação a qualquer momento. Eles concordaram em participar mediante a assinatura de um Termo de Consentimento Livre e Esclarecido (TCLE), cujos modelos podem ser encontrados nos apêndices C e G.

1.4 Organização da Dissertação

Neste capítulo, foi apresentada uma visão geral do contexto e da motivação desta pesquisa, bem como seus objetivos e a metodologia adotada para alcançá-los. No Capítulo 2, são apresentadas, de forma concisa, as principais referências teóricas que embasam este estudo. Os tópicos abordados incluem conceitos de desenvolvimento e testes em aplicativos móveis, usabilidade e acessibilidade, *design* acessível, além de diretrizes de acessibilidade e estratégias para a inclusão de acessibilidade em aplicativos para dispositivos móveis, com destaque para as ferramentas disponíveis para testes de acessibilidade em aplicativos Android nativos.

No Capítulo 3, é realizada uma revisão da literatura, abordando os problemas de acessibilidade, as recomendações para usuários com deficiência visual, a acessibilidade atual em aplicativos Android, a integração da acessibilidade no processo de desenvolvimento, investigada por meio de um Mapeamento Sistemático e pela revisão de estudos com desenvolvedores, e, por fim, as ferramentas para testes de acessibilidade no Android encontradas na literatura.

No Capítulo 4, são apresentadas estratégias definidas para capacitar os desenvolvedores a criar aplicativos acessíveis, com foco em deficiência visual, como a definição de escopo das verificações de acessibilidade, a distribuição dos testes e a seleção de ferramentas. Também é discutida a importância de atribuir mais responsabilidade aos desenvolvedores. Para isso, são apresentados uma proposta para inserir requisitos de acessibilidade com TDD e os resultados de entrevistas realizadas com profissionais da indústria, com base nas estratégias selecionadas.

No Capítulo 5, é apresentado o AALT, um *framework* de apoio à produção de aplicativos acessíveis para Android. São detalhadas a estrutura do AALT e os artefatos produzidos, incluindo um conjunto de requisitos de acessibilidade, um *kit* para testes locais de acessibilidade e treina-

mentos em formato de *codelabs*. O capítulo descreve também o planejamento, a condução e os resultados de uma avaliação da usabilidade do kit de testes locais de acessibilidade do AALT, realizada com estudantes de turmas de Teste de Software.

Finalmente, no Capítulo 6, são apresentadas as conclusões deste trabalho, destacando as principais contribuições, limitações e possibilidades de trabalhos futuros.

CAPÍTULO

2

FUNDAMENTAÇÃO

O objetivo deste trabalho, apresentado na Seção 1.2, envolve conceitos das áreas de IHC e Engenharia de Software. Mais especificamente, visando apoiar desenvolvedores, de modo pragmático na produção de aplicativos móveis acessíveis, esta pesquisa enfatiza acessibilidade como alvo, bem como processos e métodos de desenvolvimento de software como meios para alcançar este alvo. Neste capítulo, esses conceitos são apresentados e discutidos no contexto da computação móvel.

2.1 Desenvolvimento de aplicativos móveis

O número de dispositivos móveis tem crescido significativamente (OLIVEIRA *et al.*, 2019; SILVA; JUNIOR; FREIRE, 2022). Grande parte da população faz uso constante de dispositivos móveis para facilitar e realizar diversas tarefas cotidianas (SILVA *et al.*, 2020). O caráter portável e ubíquo destes dispositivos apresenta desafios e oportunidades para os desenvolvedores, como a limitação de recursos e a variedade de plataformas existentes (SAKAMOTO; SILVA; MIRANDA, 2012). Alguns destes desafios, detalhados por Zhang e Adipat (2005), continuam sendo alvo de pesquisas e motivaram soluções, as quais surgiram e têm evoluído a partir dos avanços de conhecimentos na área. A seguir, são resumidos esses desafios:

- **Contexto Móvel**. Características situacionais da interação, como localização, identificação de pessoas próximas, objetos, assim como elementos do ambiente que possam desviar a atenção do usuário.
- **Conectividade**. Condições da conexão de rede sem-fio, como qualidade do sinal, largura de banda, velocidade de transferência de dados e a variação desses parâmetros de acordo com a mobilidade do usuário.
- **Telas menores**. A limitação no tamanho da tela pode afetar significativamente a usabilidade dos aplicativos móveis, seja na apresentação ou na navegação e interação dos usuários com o dispositivo.

- **Métodos de entrada de dados**. Botões e rótulos pequenos limitam a eficiência e a eficácia dos usuários, o que pode tornar a entrada de dados mais lenta e mais suscetível a erros.

Para Wasserman (2010), o desenvolvimento de aplicativos móveis apresenta questões comuns às já estudadas em Engenharia de Software para aplicações embarcadas, como por exemplo, a integração com o dispositivo de hardware e problemas relacionados a segurança, *performance*, confiabilidade e limitação de armazenamento. No entanto, Wasserman (2010) apresenta requisitos presentes no contexto *mobile* que são menos comumente encontrados nos softwares embarcados tradicionais, como:

1. Maior possibilidade de interação com outros aplicativos;
2. Uso de sensores, acelerômetro, tela sensível ao toque, GPS, microfone e câmera;
3. Aplicativos nativos ou híbridos que invocam serviços da Web, com ou sem a utilização de um navegador;
4. Variedade de famílias de hardware e plataformas de software. Por exemplo, um desenvolvedor Android deve decidir para que versões do sistema operacional seu aplicativo será compatível;
5. Questões de segurança adicionais pelo caráter aberto das plataformas e pelo acesso direto dos aplicativos à Internet;
6. Aplicativos móveis devem compartilhar elementos comuns de interface do usuário e devem aderir a diretrizes externas;
7. Maior complexidade dos testes, que muitas vezes requerem uma simulação do contexto de uso, com uso de recursos presentes nos dispositivos e uso de rede de dados móveis;
8. Aplicativos móveis podem fazer uso extensivo de recursos, como o consumo de bateria.

Para lidar com esses desafios e particularidades dos dispositivos móveis, são necessários modelos de usabilidade que capturem adequadamente as complexidades da interação com aplicativos em plataforma móvel (HARRISON; FLOOD; DUCE, 2013).

2.1.1 Tipos de aplicativos móveis

Os aplicativos para dispositivos móveis são geralmente divididos em três categorias: aplicativos web, aplicativos nativos e aplicativos híbridos (NUNKESSER, 2018; AHMAD *et al.*, 2018).

Aplicativos web são acessados por meio de um navegador e são desenvolvidos usando tecnologias web comuns, como *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript. Eles são mais rápidos de desenvolver e são facilmente portáveis entre diferentes plataformas (AHMAD *et al.*, 2018). No entanto, eles têm acesso limitado aos recursos do dispositivo e requerem uma conexão com a internet para funcionar corretamente (CARVALHO; FREIRE, 2017).

Aplicativos nativos são desenvolvidos especificamente para uma plataforma específica, como Android ou iOS, utilizando linguagens de programação nativas, como Java, Kotlin,

Objective-C e Swift. Eles têm acesso total aos recursos do dispositivo e geralmente têm melhor desempenho (BOSNIC; PAPP; NOVAK, 2017). No entanto, eles exigem habilidades e recursos adicionais para desenvolver e manter versões separadas para cada plataforma.

Aplicativos híbridos combinam elementos de aplicativos web e nativos. Eles são desenvolvidos usando *frameworks* e ferramentas voltadas para web, como HTML e JavaScript, e depois são empacotados dentro do controle do navegador da plataforma, o que permite a execução em diferentes plataformas. Isso oferece a possibilidade de reutilizar o código entre plataformas e ter acesso a recursos do dispositivo. No entanto, eles podem apresentar desempenho ligeiramente inferior em comparação com aplicativos nativos (BOSNIC; PAPP; NOVAK, 2017) e podem depender de *plugins* adicionais para acessar recursos específicos.

A escolha adequada da abordagem de desenvolvimento deve levar em consideração as vantagens e desvantagens de cada tipo de aplicativo, uma vez que essa decisão também influencia a acessibilidade do aplicativo, conforme discutido posteriormente.

2.2 Testes em dispositivos móveis

Teste de software é uma técnica importante para escrever software de qualidade (CRISPIN; GREGORY, 2009; DELAMARO; MALDONADO; JINO, 2016; WINKELMANN; TROOST; KUCHEN, 2022). Testes devem ser conduzidos desde a concepção e durante todo o processo de desenvolvimento do software (DELAMARO; MALDONADO; JINO, 2016).

A atividade de testes é dividida em fases, que diferem em objetivo, escopo e grau de isolamento, e são (DELAMARO; MALDONADO; JINO, 2016):

- **Testes de unidade** examinam as menores partes de um programa, como classes ou métodos, para identificar erros em algoritmos, estrutura de dados ou erros de programação. Esses testes podem ser realizados pelo desenvolvedor durante a implementação das unidades, sem necessidade do sistema estar totalmente finalizado.
- **Testes de integração** examinam a interação entre diversas unidades de código ou o uso de recursos externos. Eles são aplicados quando diferentes partes do software começam a operar em conjunto, para assegurar que essa interação é adequada e não gera erros. Assim como os testes de unidade, os testes de integração tendem a ser realizados pela equipe de desenvolvimento, devido ao conhecimento necessário das estruturas internas e interações do sistema.
- **Testes de sistemas** são realizados em um sistema completamente integrado para verificar se ele atende aos requisitos especificados. Esses testes, que englobam todos os componentes do sistema e suas interações com sistemas externos ou bancos de dados, avaliam aspectos como correção, completude, coerência e cumprimento de requisitos não funcionais. É comum que equipes independentes sejam designadas para conduzir esses testes.

Figura 2 – Pirâmide de sugestão para distribuição de testes de acordo com os tipos.

Fonte: Adaptada de Google (2023b).

Na documentação para desenvolvedores Android (GOOGLE, 2023b), os testes de unidade, de integração e de sistemas também são chamados, respectivamente, de **testes pequenos**, **testes médios** e **testes grandes**. É importante notar que os **testes grandes** na documentação referem-se, na maioria das vezes, aos **testes de UI**. Esses, um subconjunto dos testes de sistema que se concentra na interface do usuário de um aplicativo, assegurando que funcione como esperado. Eles avaliam a interação entre o usuário e a interface do aplicativo, sendo aplicados quando é necessário testar fluxos do usuário na tela ou o funcionamento de partes extensas do aplicativo (GOOGLE, 2023b).

Devido ao foco deste trabalho na acessibilidade digital em dispositivos móveis, portanto, em aspectos relacionados à UI, adotaremos a relação de equivalência apresentada no Quadro 1.

Quadro 1 – Relação de equivalência entre as denominações dos tipos de testes no escopo deste trabalho.

Testes pequenos	Testes de unidade
Testes médios	Testes de integração
Testes grandes	Testes de UI

Fonte: Elaborada pelo autor.

Uma abordagem de cobertura de testes é demonstrada na Figura 2. Testes pequenos são mais baratos e oferecem um *feedback* mais rápido aos desenvolvedores e devem ser privilegiados. Testes grandes são mais onerosos e demorados, porém mais fidedignos ao fluxo do usuário (ANICHE, 2014; GOOGLE, 2023b; CRISPIN; GREGORY, 2009).

Uma outra divisão dos testes no ambiente Android é quanto ao ambiente de execução. Os **testes locais** são aqueles que podem ser executados diretamente no ambiente de desenvolvimento. Já os **testes instrumentados** são aqueles que requerem algum dispositivo Android, seja físico

ou emulado, e geralmente são para testes de UI (GOOGLE, 2023b). É possível realizar testes de unidade instrumentados, porém são testes consideravelmente mais lentos que os testes de unidade locais (GOOGLE, 2023b).

Dentro das categorias de testes, destaca-se também os testes de regressão. Realizados durante a manutenção do software, eles evitam a introdução de novos erros após modificações, assegurando que alterações recentes operam corretamente e que os requisitos anteriormente testados continuam válidos (DELAMARO; MALDONADO; JINO, 2016).

Nos modelos tradicionais de processo de software, os testes de regressão ocorrem após o desenvolvimento, durante a manutenção do software (DELAMARO; MALDONADO; JINO, 2016). Porém, no desenvolvimento ágil de software (BECK *et al.*, 2001; PRESSMAN, 2011), que prioriza ciclos curtos e entrega antecipada e contínua do software, essa separação tende a ser mais tênue.

2.3 Desenvolvimento e teste ágil

O Desenvolvimento Ágil de Software (ASD - *Agile Software Development*) foi proposto para otimizar o desenvolvimento de software, priorizando a flexibilidade e a interação ao invés de planejamento e documentação prévia. A adoção de métodos ágeis promove a interação com o cliente, motiva os desenvolvedores e acelera o processo, favorecendo entregas mais rápidas e de maior qualidade (BECK *et al.*, 2001).

A abordagem *Extreme Programming* (XP) do ASD é fundamentada em cinco valores principais: comunicação, simplicidade, *feedback*, coragem e respeito (BECK; ANDRES, 2004).

Uma das práticas de XP é o Desenvolvimento Orientado a Testes (TDD - *Test-Driven Development*), que orienta a criação de testes de unidade antes da implementação de um novo requisito de software (PRESSMAN, 2011; ANICHE, 2014). Um ciclo típico de TDD segue os seguintes passos (BECK, 2002):

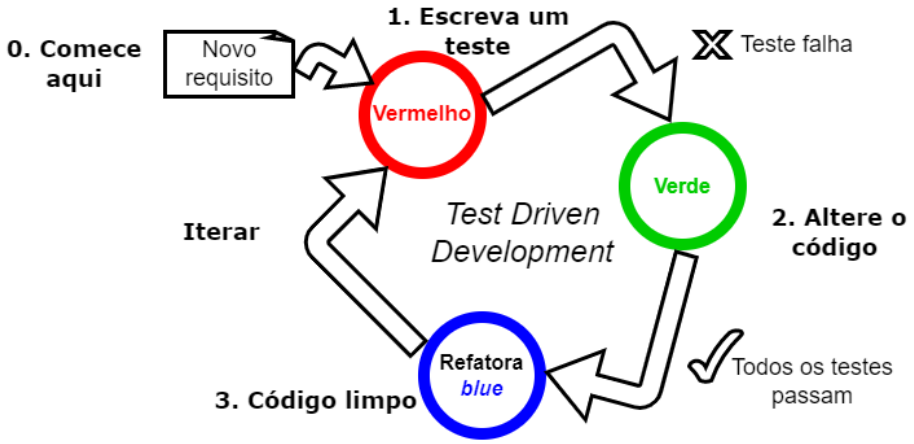
1. adicionar rapidamente um teste que falhe;
2. fazer pequenas alterações até que o teste passe;
3. refatorar para remover duplicação.

Esse ciclo também é conhecido como vermelho-verde-refatora (ANICHE, 2014). Um exemplo de ciclo de desenvolvimento iterativo focado em testes pode ser observado na Figura 3.

O TDD oferece aos desenvolvedores um *feedback* rápido a cada implementação de um novo requisito, possibilitando a identificação e correção de problemas de maneira eficiente e com menor custo (ANICHE, 2014).

Embora seja considerada uma técnica de desenvolvimento, o TDD também pode favorecer as etapas de testes. Para isso, os testes de unidade criados dentro de um ciclo TDD devem ser estruturados de modo a permitir sua automação, o que favorece estratégias de testes de regressão,

Figura 3 – Ciclos de desenvolvimento iterativos orientados a testes



Fonte: Adaptada de IBM (2023).

alinhadas à filosofia XP (PRESSMAN, 2011).

Além disso, com os testes de unidade individuais organizados em um conjunto de testes automatizados, testes de integração podem acontecer diariamente (PRESSMAN, 2011).

Na metodologia ágil, a responsabilidade pela qualidade do software é compartilhada por toda a equipe, não restringindo-se apenas aos testadores. Todos estão engajados em atividades de teste e colaboram continuamente para entregar um software de alta qualidade. Essas atividades abrangem automação de testes, testes exploratórios e projeto de código visando a testabilidade. Essa abordagem integrada de equipe promove a colaboração constante entre testadores, programadores e outros membros da equipe para assegurar a qualidade do software (CRISPIN; GREGORY, 2009).

2.3.1 Automação de testes

A automação de testes pode apoiar práticas ágeis como a Integração Contínua (CI) e Entrega Contínua (CD), que dependem da capacidade de executar testes rapidamente e frequentemente (DUVALL; MATYAS; GLOVER, 2007; CRISPIN; GREGORY, 2009).

A aplicação de testes automatizados traz benefícios como a redução do tempo de execução, uma menor propensão a erros, a liberação de recursos, a segurança para testes de regressão e o *feedback* rápido e frequente. Além disso, esses testes oferecem benefícios adicionais, como impulsionar a codificação, servir como documentação e ter um bom retorno sobre o investimento (CRISPIN; GREGORY, 2009). A automação aprimora a eficiência, a qualidade e a cobertura dos testes, permitindo que os desenvolvedores se concentrem em outras atividades importantes para as aplicações (DELAMARO; MALDONADO; JINO, 2016).

Existem muitas ferramentas e *frameworks* disponíveis que podem ajudar a implementar a automação de testes (DELAMARO; MALDONADO; JINO, 2016; CRISPIN; GREGORY,

2009). Para a plataforma Android são disponibilizadas ferramentas como o UI Automator e o Espresso para testes de UI e o Roboletric para criação de testes de unidade. Mais detalhes sobre essas ferramentas serão descritos na Subseção 2.5.2, no contexto de testes de acessibilidade.

2.4 Acessibilidade

Acessibilidade é o termo geral que qualifica um produto de modo a permitir que quaisquer pessoas tenham seu acesso, visando usufruir os benefícios da vida em sociedade. No contexto de desenvolvimento de software, a acessibilidade é um requisito não-funcional e uma subcaracterística de usabilidade (ISO 9241-11, 2018).

A Norma ISO 9241-11 (2018) define **usabilidade** como "o grau em que um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico". Por sua vez, a **acessibilidade** é definida como:

> "o grau em que produtos, sistemas, serviços, ambientes e instalações podem ser usados por pessoas de uma população com a mais ampla gama de necessidades, características e capacidades de usuários para atingir objetivos identificados em contextos de uso identificados."

Na mesma Norma ISO 9241-11 (2018), são enumeradas quatro abordagens importantes para melhorar a acessibilidade das interfaces humano-computador, sendo elas:

(1.) adotar uma abordagem centrada no ser humano,
(2.) seguir um processo de *design* baseado em contexto,
(3.) fornecer a capacidade de individualização, e
(4.) oferecer instrução e treinamento individualizado ao usuário.

Segundo Thatcher *et al.* (2002), acessibilidade é um subconjunto da usabilidade e pode se beneficiar dos mesmos cinco atributos definidos por Nielsen (1993): facilidade de aprendizado, eficiência, facilidade de memorização, prevenção de erros e satisfação. Shneiderman (2000) propõe a "usabilidade universal" como um termo que engloba tanto a acessibilidade quanto a usabilidade. Ele ressalta que "o acesso, por si só, não garante o uso bem-sucedido", sugerindo que a acessibilidade é um primeiro passo importante, mas não o único, para a usabilidade universal.

No estudo de Petrie e Kheir (2007), participantes com e sem deficiência visual evidenciaram que os problemas de acessibilidade e usabilidade não eram subconjuntos um do outro, contrariando Thatcher *et al.* (2002) e Shneiderman (2000). Eles descobriram que os usuários cegos, ao usar leitores de tela, identificaram problemas específicos não encontrados no uso visual dos sites. Isso sugere que estudar os problemas de acessibilidade pode realçar e expandir nossa compreensão dos problemas de usabilidade.

Portanto, é possível observar que o conceito de acessibilidade está fortemente relacionado ao de usabilidade. No entanto, Petrie e Bevan (2009) sugerem uma distinção prática baseada no

usuário, na qual a acessibilidade foca nos problemas enfrentados por usuários com deficiências, enquanto a usabilidade trata dos problemas dos usuários sem deficiências.

No contexto da Web, a acessibilidade visa que todas as pessoas, especialmente as pessoas com deficiência e idosos, possam acessar os conteúdos dos *sites* em uma variedade de contextos de uso, incluindo os mais usuais, e os que contam com apoio de Tecnologia Assistiva (T.A.) (PETRIE; SAVVA; POWER, 2015).[1] Considerar que diferentes usuários possuem diferentes habilidades e podem utilizar diferentes tecnologias é imprescindível para que acessibilidade seja de fato respeitada. Usuários com deficiência visual, com deficiência auditiva, com deficiência motora, com deficiência cognitiva e os idosos, de fato, podem recorrer aos mais variados recursos, serviços e tecnologias para obter acesso aos sistemas Web (BRASIL, 2015).

2.4.1 Diretrizes de acessibilidade

Para criar aplicativos acessíveis a todos os usuários, é imperativo seguir os padrões internacionais de acessibilidade (BOURAOUI; GHARBI, 2019). Para contribuir com uma Web acessível a um número cada vez maior de pessoas, a Iniciativa para a Acessibilidade na Web (*Web Accessibility Initiative* (WAI)) foi lançada em 1997 pelo W3C (DARDAILLER, 2009). Como resultado desta iniciativa, foi publicada em 1999 a primeira versão das Diretrizes para Acessibilidade do Conteúdo da Web - *Web Content Accessibility Guidelines* (WCAG), que definem um conjunto de recomendações sobre como tonar o conteúdo da Web acessível (W3C, 1999). Na versão 2.1 da WCAG (W3C, 2018), de 2018,[2] as diretrizes buscaram melhorar as orientações de acessibilidade para grupos específicos de usuários, incluindo usuários com deficiências que utilizam dispositivos móveis.

Um exemplo típico da importância dessas diretrizes na acessibilidade ao conteúdo Web é a primeira delas, que aborda textos alternativos para conteúdo não textual. Por exemplo, um usuário com deficiência visual pode não conseguir visualizar uma foto apresentada na tela, mas um leitor de telas[3] poderá ler o texto alternativo que descreve esta foto, quando fornecido, sintetizando de modo sonoro o que é lido naquele texto alternativo, para o usuário.

Para corresponder às necessidades da variedade de interessados na WCAG, o documento é estruturado em camadas de orientação que incluem (W3C, 2018):

[1] Tecnologia Assistiva é uma área do conhecimento, de característica interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social (BRASIL, 2015).

[2] A versão 2.2 está, em outubro de 2023, em etapas finais de revisão, e já existe uma versão 3.0 em andamento, disponível em <https://www.w3.org/TR/wcag-3.0/>.

[3] Leitores de tela são ferramentas que auxiliam usuários com deficiência visual a utilizarem as interfaces gráficas de um *software* Um leitor de tela incorpora fundamentalmente um sistema de narração de texto, que emite esta narração de modo audível. No entanto, ele vai além deste sistema, sendo capaz de interpretar e interagir com a interface gráfica do programa, descrevendo-a para o usuário e ajudando-o na sua utilização (AFB, 2017)

- **Princípios**. Estabelecem a base necessária para acesso ao conteúdo da Web por qualquer pessoa. São quatro os princípios:
  1. **Perceptível**. Os usuários devem ser capazes de perceber a informação apresentada.
  2. **Operável**. Os usuários devem ser capazes de operar a interface.
  3. **Compreensível**. Os usuários devem ser capazes de entender a informação apresentada e a interação esperada.
  4. **Robusto**. Os usuários devem ser capazes de acessar o conteúdo de acordo com as tecnologias atuais e futuras disponíveis, incluindo as tecnologias de apoio.
- **Diretrizes**. Procuram auxiliar a proporcionar que o conteúdo seja acessível ao maior número de pessoas e adaptável às diferentes habilidades físicas, sensoriais ou cognitivas. São 13 diretrizes, associadas aos 4 princípios estabelecidos.
- **Critério de sucesso**. Sob cada diretriz, estão alocados os critérios de sucesso, os quais são afirmações testáveis para determinar se o conteúdo satisfaz o critério. No total, são 78 critérios de sucesso que organizam-se nas 13 diretrizes, e assemelham-se a requisitos. Os critérios são definidos em três níveis de conformidade: A (o mais básico), AA e AAA (o mais elevado).
- **Técnicas suficientes e sugeridas**. São técnicas suficientes aquelas que, uma vez implementadas, desde que exista suporte a acessibilidade para o usuário (por exemplo, T. A. disponível), atenderá ao critério de sucesso. Um exemplo de técnicas sugeridas são as que podem melhorar a acessibilidade ao conteúdo, mas podem não ser consideradas suficientes.

O Critério de Sucesso 1.3.4, apresentado como exemplo no Quadro 2, é um dos critérios adicionados na versão 2.1 do documento WCAG. Esse critério é endereçado a usuários com deficiência usando dispositivos móveis.

Quadro 2 – Exemplo de critério de sucesso da WCAG 2.1

Princípio	1. Perceptível
Diretriz	1.3 Adaptável
Critério de Sucesso 1.3.4 Orientação (nível AA)	
O conteúdo não restringe sua visualização e operação a uma única orientação de exibição, como retrato ou paisagem, a menos que uma orientação de exibição específica seja essencial.	
Intenção	A intenção deste Critério de Sucesso é garantir que o conteúdo seja exibido na orientação (retrato ou paisagem) preferida pelo usuário.
Benefícios	– Usuários com deficiência de destreza, que possuem um dispositivo fixado, poderão usar o conteúdo em sua orientação fixa. – Os usuários com baixa visão poderão visualizar o conteúdo na orientação que funciona melhor para eles, por exemplo, para aumentar o tamanho do texto, exibindo o conteúdo em modo paisagem.

Fonte: Adaptada de W3C (2018).

O enunciado do Critério de Sucesso 1.3.4 da WCAG 2.1, exemplificado no Quadro 2,

é apresentado tal qual encontra-se disponibilizado pela WAI.[4] Observamos que refere-se a uma explicação clara para o contexto de uso de aplicativos móveis, mas por se tratar de uma verificação quanto ao que se espera de comportamento e da apresentação das informações do aplicativo, fica a cargo do desenvolvedor buscar por soluções técnicas que satisfaçam essa demanda.

Assim, nesse contexto de cumprimento de *guidelines* / diretrizes / critérios de sucesso, os desenvolvedores argumentam que em função de exigências relacionadas ao tempo dedicado ao trabalho e conhecimentos mais aprofundados sobre as tecnologias adotadas em seus projetos, tornam-se difíceis e custosos os esforços dispensados na busca e implementação de soluções técnicas adequadas, no ambiente profissional (ANTONELLI *et al.*, 2018; SWALLOW; PETRIE; POWER, 2016; BI *et al.*, 2022).

2.4.2 Acessibilidade em aplicativos móveis

Aplicativos para dispositivos móveis podem auxiliar na realização de tarefas cotidianas das pessoas. Porém, pessoas com deficiência podem encontrar diversas barreiras para utilizarem os recursos desses dispositivos se eles não fornecerem acessibilidade adequada.

Grande parte dos esforços encontrados na literatura para criação de aplicativos móveis mais acessíveis concentra-se na elaboração ou na organização de diretrizes e recomendações para incluir a acessibilidade neste tipo de dispositivo (W3C, 2008; BBC, 2022; GOOGLE, 2023h; SIDI, 2017). A seguir são descritos alguns desses esforços.

Guias para desenvolvimento de aplicativos móveis acessíveis

A acessibilidade de aplicativos móveis pode ser avaliada manualmente pela exploração e inspeção de cada componente da interface do usuário (ELER *et al.*, 2018), a fim de verificar possíveis barreiras de interação. Heurísticas e *guidelines* podem ser utilizadas neste processo.

Para guiar a verificação de conformidade no contexto *mobile*, em fevereiro de 2015 a WAI disponibilizou o *Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile*.[5] Esse documento descreve como os princípios, *guidelines* e critérios de sucesso da WCAG podem ser aplicados para *mobile*. A WAI já havia publicado, em 2008, o *Mobile Web Best Practices* (MWBP),[6] e em 2010, o *Mobile Web Application Best Practices* (MWABP).[7] Ambos os documentos apresentam conjuntos de práticas recomendadas para, respectivamente, entrega de conteúdo web em dispositivos móveis e desenvolvimento de aplicações web ricas e dinâmicas para dispositivos móveis. No MWABP, são reunidas as práticas de engenharia mais

[4] Critério de Sucesso 1.3.4 - <https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/#orientation>
[5] <https://www.w3.org/TR/mobile-accessibility-mapping/>
[6] <https://www.w3.org/TR/mobile-bp/>
[7] <https://www.w3.org/TR/mwabp/>

relevantes, incentivando o uso das que contribuem para a experiência do usuário e alertando para as que podem ser prejudiciais.

Além desses documentos, desenvolvedores de aplicativos *mobile* nativos também podem orientar-se por *guidelines* de usabilidade dos fabricantes. Para aplicativos Android, a Google disponibiliza um guia para tornar os aplicativos acessíveis (GOOGLE, 2023h). De modo análogo, a Apple disponibiliza o *Acessibility for Developers* (APPLE, 2022) com *guidelines* e exemplos de código para iOS.

Outras iniciativas promovem a organização de diretrizes para acessibilidade em aplicativos móveis. É o caso do *Mobile Accessibility Standards and Guidelines* (BBC, 2022), criado e disponibilizado pela *British Broadcasting Corporation* (BBC). O guia organiza recomendações de acessibilidade para *mobile* em onze categorias: áudio e vídeo, *design*, editorial, foco, formulários, imagens, *links*, notificações, *scripts* e conteúdo dinâmico, estrutura e textos equivalentes. Além disso, o guia da BBC também oferece exemplos de código específicos para cada plataforma, quando aplicável, e procedimentos para a realização de testes com o uso de T.A. para cada recomendação.

Em Siebra *et al.* (2017), os autores de GuAMA (*Guide to the Development of Accessible Mobile Applications*) classificam os requisitos de acessibilidade por tipo de deficiência, incluindo deficiência visual, deficiência auditiva e deficiência motora. Os requisitos são também qualificados em três categorias: requisitos essenciais, requisitos desejáveis e não observados. Esse trabalho serviu de base para a criação do Guia para o Desenvolvimento de Aplicações Móveis Acessíveis (SIDI, 2017), em parceria com o Samsung Instituto de Desenvolvimento para Informática (SIDI). Além dos requisitos de acessibilidade, que atualmente totalizam 48, o guia apresenta recomendações para *designers* de UX/UI, boas práticas para desenvolvedores, incluindo exemplos de implementação para Android, e estratégias para o planejamento e execução de testes de um aplicativo acessível.

2.4.3 Design e acessibilidade em aplicativos móveis

Designers de UX/UI desempenham um papel que vai além da criação dos aspectos visuais e da navegação de um aplicativo. Eles também são responsáveis por definir o conteúdo textual audível, como descrições, tamanho e apresentação, para garantir a acessibilidade quando o leitor de tela estiver em uso (SIDI, 2017).

Abordagens, sistemas de *design* (*design systems*)[8] e padrões de projeto podem ser utilizados pelos *designers* para promover o *design* de interfaces interativas acessíveis.

[8] Um *design system*, ou sistema de design, é um conjunto de diretrizes, padrões e componentes reutilizáveis que são usados para criar e manter a consistência visual e funcional em um projeto de *design*.

Design Centrado no Usuário

Uma das abordagens recomendadas para criar algo com mais acessibilidade, antecipando as necessidades potenciais de pessoas com deficiências físicas ou algum tipo de deficiência é o *Design* Centrado no Usuário (UCD - User-Centered *Design*) (PINHEIRO; MARQUES, 2021).[9]

O UCD é um processo de *design* que se concentra nas necessidades do produto sendo projetado para os usuários, e, portanto, coloca os usuários no centro das decisões de *design* (WILLIAMS, 2009). UCD auxilia o *designer* a determinar as consequências, em termos de usabilidade, das suas decisões (DIX *et al.*, 2004).

Um típico processo iterativo de *design* e desenvolvimento centrado no usuário é ilustrado na Figura 4. As seguintes fases compõem o processo UCD (PETRIE; BEVAN, 2009), (NORMAN, 2013):

1. **Conhecer os usuários.** (*Observação*) Envolve o estudo e entendimento do contexto-de-uso corrente e das expectativas para o novo produto. Pode envolver técnicas como entrevista com usuários potenciais, estudos etnográficos, análise de tarefas, entre outras ferramentas para obter os requisitos do usuário.
2. ***Design*.** Também chamada por Norman (2013) de "geração de ideias", essa é a fase de exploração das possibilidades de *design* e de como cada alternativa considerada pode atender às necessidades dos usuários. Personas e cenários são atividades usadas nesta fase.
3. **Protótipo.** Os protótipos ajudam a testar e validar as ideias levantadas no *design*. Nas fases iniciais do desenvolvimento podem ser utilizados protótipos de baixa fidelidade, como *sketches* feitos em papel, para que possam ser descartados e recriados quantas vezes for necessário. Protótipos de alta fidelidade geralmente são parcialmente funcionais e úteis para validar aspectos ou resolver problemas do *design*.
4. **Avaliação.** Avaliar os protótipos e iniciar novas iterações no processo é a chave para alcançar níveis satisfatórios de usabilidade, acessibilidade e experiência do usuário. Vários métodos podem ser utilizados nesta fase, como checagem automática de conformidade com padrões e *guidelines*, avaliações heurísticas, coleta de dados de uso e testes com usuário.

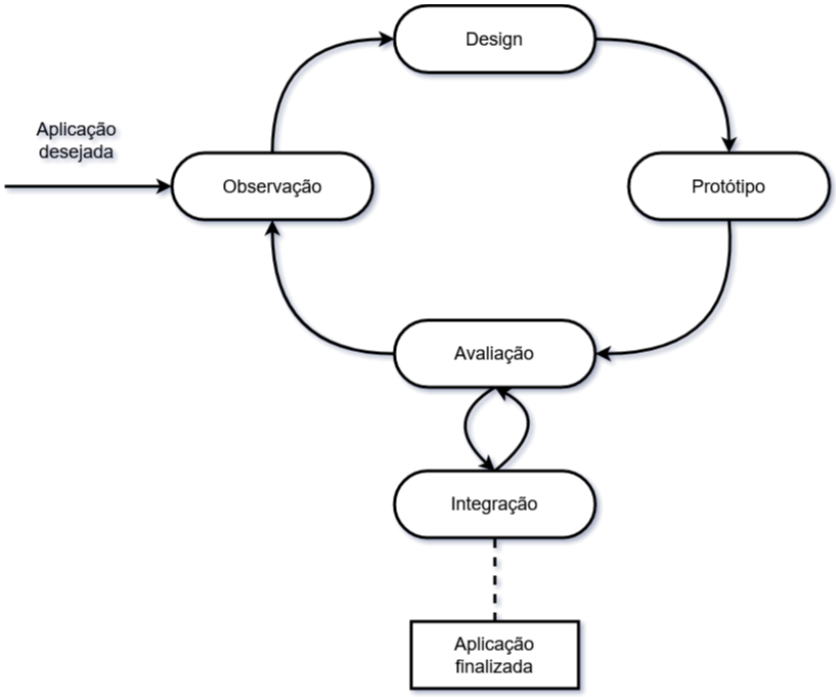
Essas quatro fases podem ser realizadas em ciclos até que um resultado satisfatório seja alcançado. Com o envolvimento dos usuários no processo, os desenvolvedores podem adquirir melhor entendimento das necessidades e dos objetivos dos usuários, resultando num produto mais útil e adequado (DIX *et al.*, 2004).

Material Design

O Material Design (GOOGLE, 2023a) é um sistema com diretrizes, componentes e boas práticas para o *design* de UI. Desenvolvido pelo Google, ele visa criar uma experiência de usuário

[9] A ISO 9241-210 (2019) atualizou a terminologia para *Design* centrado no humano (*Human-centred design*), além de alterar algumas definições. Porém, como parte do material desta pesquisa foi gerado anteriormente a essa atualização, optou-se por manter a terminologia e definições anteriores.

Figura 4 – Um típico processo iterativo de *design* centrado no usuário.



Fonte: Adaptada de Petrie e Bevan (2009).

unificada em todas as plataformas e dispositivos, sejam eles móveis ou *desktop*, baseado na metáfora dos elementos de UI com materiais do mundo real, como tinta e papel.

Alguns padrões de acessibilidade são incorporados nos componentes do Material Design para fornecer uma base para o *design* de produtos inclusivos, como contraste de cores e redimensionamento de texto. No entanto, o sistema também apresenta limitações em relação à acessibilidade, como a falta de rótulos e descrições em elementos interativos, inadequação no foco e destaque de elementos interativos e pouca flexibilidade para personalizar a interface de acordo com as necessidades específicas de acessibilidade.

Design Patterns de Interface do Usuário

Segundo Gamma *et al.* (1995), *design patterns* são descrições de objetos e classes que são personalizados para resolver um problema recorrente em um contexto específico. O uso de *design patterns* leva a melhores práticas e a um vocabulário comum entre desenvolvedores. No domínio de IHC, *design patterns* têm sido bastante adotados, sendo conhecidos como *Design Patterns* de Interface do Usuário - *User Interface Design Patterns* (UIDPs) (BATISTA, 2018).

UIDPs são descrições de melhores práticas na solução de problemas recorrentes de *design* de interface de usuário. Para essa descrição, os seguintes elementos da solução de projeto devem ser informados: problema, contexto de uso, princípio, solução, por quê, exemplos e implementação (FOLMER, 2019).

O exemplo da Figura 5 foi extraído do Material Design e representa um UIDP de *Inline*

error message (IEM). Neste *design pattern*, quando ocorre um erro de entrada, o sistema notifica o usuário com uma mensagem de erro e um indicador no campo (ou em torno dele) onde o erro foi identificado (NUDELMAN, 2013). Para Batista (2018), a procura em formulários grandes por campos que necessitam de correção na inserção é cansativa aos usuários, especialmente àqueles com alguma deficiência. O correto uso deste padrão pode levar o foco para a informação a ser corrigida, ajudando estes usuários.

Figura 5 – Exemplo de uso do UIDP *Inline error message*.

Fonte: Adaptada de Design (2019).

2.5 Testes de acessibilidade no Android

Verificações de acessibilidade em aplicativos móveis podem ser facilitadas com a utilização de ferramentas, mas para obter maior confiança de que os aplicativos estão sendo projetados de acordo com as diretrizes mencionadas, é recomendado combinar várias abordagens.

No contexto de aplicativos Android, a própria documentação para desenvolvedores da plataforma (GOOGLE, 2023f) sugere quatro abordagens:

- **Teste manual**. Os testes manuais exploram o aplicativo da forma como o usuário o utilizará. Testar manualmente a acessibilidade de aplicativos móveis envolve explorar e inspecionar os aplicativos, verificando cada componente de interface do usuário. Podem ser realizados, por exemplo, interagindo com o aplicativo utilizando os serviços de acessibilidade do Android, tais como o "*TalkBack*"[10], o "*BrailleBack*"[11] e o "*Voice Access*"[12].
- **Teste com ferramentas de análise**. Favorecem a identificação de problemas por meio do código e interface.
- **Teste automatizado**. Dentro da Plataforma Android, há suporte para diversos frameworks de teste que possibilitam a criação e execução de testes automatizados para avaliar a acessibilidade do aplicativo.

[10] *TalkBack* é software leitor de tela para smartphones Android. Trata-se de um recurso de narração que ajuda pessoas com deficiência visual a usarem as funções do aparelho e que pode ser ativada nas configurações do dispositivo.

[11] *BrailleBack* é um serviço de acessibilidade que ajuda usuários cegos a usar dispositivos Braille, e funciona em conjunto com o *TalkBack*

[12] *Voice Access* é um serviço de acessibilidade que ajuda usuários que têm dificuldade em manipular tela de toque, pois permite controlar todo o sistema com a voz

- **Teste de usuários**. Feedback de pessoas reais que interagem com o aplicativo após o seu desenvolvimento ou por meio de protótipos.

A plataforma Android disponibiliza uma série de ferramentas que podem ser utilizadas nas diferentes estratégias de testes de acessibilidade. As verificações de acessibilidade pré-definidas nessas ferramentas são baseadas na biblioteca *Accessibility Test Framework* (ATF).

Nas subseções a seguir, serão apresentadas as características e checagens de acessibilidade da ATF, uma breve descrição de cada uma das ferramentas, e, por fim, uma discussão crítica sobre suas vantagens e limitações.

2.5.1 Accessibility Test Framework

A ATF é uma biblioteca Android para avaliar a acessibilidade de aplicativos que coleta verificações relacionadas à acessibilidade em Views[13] e objetos AccessibilityNodeInfo (que o Android deriva das Views e envia para AccessibilityServices – usada por recursos de T.A.) (GOOGLE, 2022). A ATF possui um conjunto pré-definido de checagens de acessibilidade, e fornece suporte/recursos para execução ou personalização das verificações de acessibilidade. Dentre esses, uma classe para habilitar verificações de acessibilidade automatizadas (AccessibilityChecks) (GOOGLE, 2023h).

Embora seja uma biblioteca aberta, que pode ser utilizada por desenvolvedores com outros testes e ferramentas, há pouca documentação disponível. O Apêndice A contém uma relação completa das checagens da ATF, extraídas do código-fonte da biblioteca.[14] As verificações de acessibilidade atualmente disponíveis incluem presença de *labels*, tamanho da área de toque, contraste de cores e outras propriedades de elementos de UI relacionados com serviços de acessibilidade do Android.

A ATF é a biblioteca usada para as verificações de acessibilidade realizadas pelas ferramentas disponibilizadas diretamente pela plataforma para desenvolvedores Android.

2.5.2 Ferramentas para testes de acessibilidade no Android

Com o objetivo de encontrar soluções que se integrem melhor ao ambiente de desenvolvimento, foram analisadas as ferramentas disponíveis para testes automatizados no Android.

[13] A classe View é o elemento fundamental para componentes de interface do usuário no Android. Ela representa uma área retangular na tela e lida com o desenho e eventos. As Views são usadas como base para *widgets* interativos, como botões e campos de texto.

[14] <https://github.com/google/Accessibility-Test-Framework-for-Android>

Android Lint

O Lint é uma ferramenta de verificação de código do Android Studio[15] que ajuda a identificar e corrigir problemas estruturais no código-fonte. Ele reporta cada problema com uma mensagem descritiva e um nível de gravidade, permitindo que os desenvolvedores priorizem as correções necessárias. O Lint verifica arquivos de origem do projeto Android em critérios de precisão, segurança, desempenho, usabilidade, acessibilidade e internacionalização.

As inspeções de acessibilidade realizadas pelo Lint são apresentadas no Quadro 3, com informações obtidas na própria IDE do Android Studio.

Quadro 3 – Inspeções de acessibilidade realizadas pelo Lint

Inspeções	
Acessibilidade em *Views* personalizadas	A lógica para as ações de clique deve ser colocada em *performClick*, que é chamado por alguns serviços de acessibilidade.
Imagem sem *contentDescription*	Elementos visuais como *ImageViews* e *ImageButtons* devem usar o atributo *contentDescription* para especificar uma descrição textual adequada para ferramentas de acessibilidade. Para campos de texto, utilize apenas o atributo *hint*, evitando a definição de ambos atributos.
Widget inacessível por teclado	Um widget declarado como clicável deve ter também o atributo 'focusable'.
Rótulo de acessibilidade ausente	Use *android:hint* ou *android:labelFor* (*minSdkVersion* >= 17) em campos de texto editáveis e certifique-se de fornecer *android:text* ou *android:contentDescription* quando usar *android:labelFor*.
getContentDescription() sobrescrito em uma *View*	Use *setContentDescription()* em vez de substituir *getContentDescription()* para garantir que os serviços de acessibilidade possam navegar adequadamente pelo conteúdo de sua *View*.

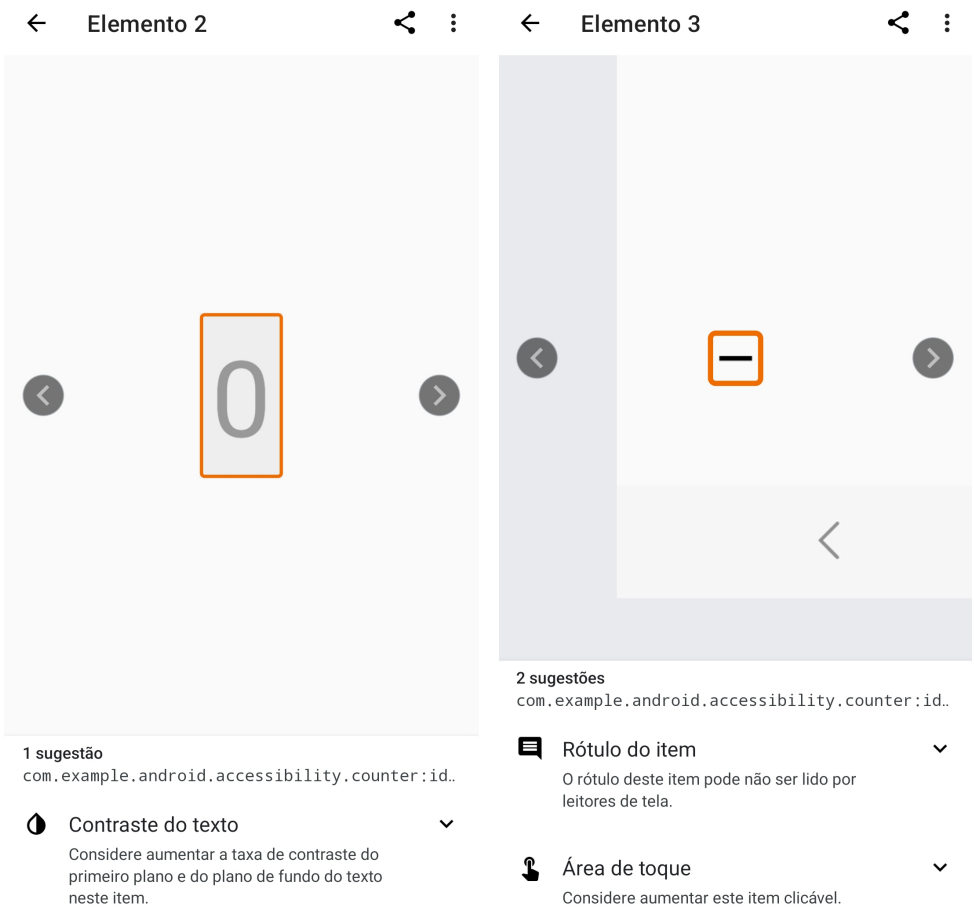
Fonte: Elaborada pelo autor.

Scanner de Acessibilidade

O Scanner de Acessibilidade é um aplicativo disponibilizado pelo Google que, quando habilitado, é capaz de analisar a tela de qualquer aplicativo e apresentar ao usuário uma visão que destaca os problemas de acessibilidade diretamente na interface de usuário. Esses pontos destacados podem ser clicados para obter informações mais detalhadas e sugestões. As verificações de acessibilidade realizadas pelo aplicativo são baseadas na ATF e incluem conteúdo dos rótulos (rótulo de item ausente, rótulos de item duplicados, campos de entrada com descrições de conteúdo e rótulos com informações redundantes), itens clicáveis (subconjuntos clicáveis de itens de texto e limites clicáveis duplicados), contraste e tamanho da área de toque.

[15] <https://developer.android.com/studio>

Figura 6 – Sugestões do Scanner de Acessibilidade sobre tela do *app Counter*



Fonte: Elaborada pelo autor.

Na Figura 6 são exibidas recomendações do Scanner de Acessibilidade para a tela do aplicativo Counter.[16]

Visualizador do UI Automator

O UI Automator (GOOGLE, 2023g) é um *framework* de teste fornecido pela plataforma Android para testar a UI de aplicativos. O UI Automator permite que os desenvolvedores escrevam testes automatizados para seus aplicativos Android que simulam a interação do usuário com os componentes da UI. As APIs do UI Automator possibilitam a interação com elementos visíveis em um dispositivo, independentemente da Activity em foco.

Já o Visualizador do UI Automator (*UI Automator Viewer*) (GOOGLE, 2023g) é uma ferramenta que permite aos desenvolvedores inspecionar os componentes da UI de um aplicativo Android. Com ele, é possível analisar a hierarquia de layouts e verificar as propriedades dos componentes visíveis. Isso possibilita a criação de testes mais precisos, como um seletor de interface do usuário com base em propriedades específicas.

[16] O aplicativo Counter é um projeto de exemplo usado em um *codelab* de acessibilidade básica para Android, do Google Codelabs.

Embora o próprio UI Automator não forneça recursos de teste de acessibilidade integrados, é possível aproveitar seus recursos para criar testes de acessibilidade personalizados. A ferramenta é útil para depurar problemas durante os testes de acessibilidade, identificando suas origens e permitindo melhorias no aplicativo.

A relação entre o UI Automator e a acessibilidade é mutuamente benéfica, pois a ferramenta obtém um desempenho melhor quando aplicada a interfaces de usuário acessíveis (GOOGLE, 2023g).

Espresso

O Espresso (GOOGLE, 2023d) é um *framework* de teste de UI para a plataforma Android. Ele funciona de modo integrado ao Android Studio e é executado em um emulador ou dispositivo Android real. O Espresso é projetado para simplificar o processo de escrita de testes de UI para aplicativos Android, permitindo que os desenvolvedores criem testes automatizados do tipo caixa branca, ou seja, testes que exploram a lógica interna e estrutura do código. Diferente dos testes caixa preta, que se concentram nas entradas e saídas do sistema (DELAMARO; MALDONADO; JINO, 2016).

O Espresso permite que os testes interajam com os elementos da UI do aplicativo, como botões, caixas de texto e menus de opções, além de realizar ações como clicar em botões e inserir texto, entre outras. Ele também pode verificar o estado dos elementos, como a presença de texto em uma caixa de texto ou a visibilidade de um botão.

Embora seja uma ferramenta para automação de testes de interface, o Espresso pode ajudar a testar acessibilidade por meio da integração com a ATF e a *Application Programming Interface* (API) AccessibilityChecks. Em cada caso de teste, o desenvolvedor pode ativar os verificadores de acessibilidade para identificar as exceções quando qualquer componente ativado por um teste falha em qualquer das checagens de acessibilidade contempladas pela ATF.

É possível, com o Espresso, habilitar que as checagens sejam realizadas em toda a hierarquia de *Views* de uma tela. Mas, é necessário que as verificações de acessibilidade sejam ativadas em cada tela, individualmente. No exemplo do Código-fonte 1, uma classe de testes instrumentados foi criada para a tela principal do aplicativo. As verificações de acessibilidade foram ativadas pela chamada do método AccessibilityChecks.enable() (linha 12). Para avaliar toda a hierarquia de Views em cada verificação, é necessário adicionar o setRunChecksFromRootView(true) (linha 13).

Código-fonte 1 – Ativação das verificações de acessibilidade da API *AccessibilityChecks* com o Espresso

```
1: @RunWith(AndroidJUnit4.class)
2: @LargeTest
3: public class MainActivityInstrumentedTest {
4:
```

```
5:      @Rule
6:      public ActivityScenarioRule<MainActivity> activityScenarioRule =
7:              new ActivityScenarioRule<>(MainActivity.class);
8:
9:      @BeforeClass
10:     public static void beforeClass()
11:     {
12:         AccessibilityChecks.enable()
13:                 .setRunChecksFromRootView(true);
14:     }
15:
16:     @Test
17:     public void test(){
18:         onView(withId(R.id.btn_test)).perform(click());
19:     }
20: }
```

2.5.3 Roboletric

Robolectric é um *framework* de testes para Android que proporciona a escrita de testes de unidade e sua execução em uma Máquina Virtual Java (JVM - *Java Virtual Machine*) de *desktop*, fazendo uso da API do Android (ELER *et al.*, 2018). Sua popularidade entre os desenvolvedores de aplicativos Android o torna uma das ferramentas mais utilizadas para testes automatizados (LINARES-VÁSQUEZ *et al.*, 2017).

A execução de testes locais em um ambiente Android simulado dentro de uma JVM faz com que as rotinas de testes sejam realizadas de forma até dez vezes mais rápida em comparação com *frameworks* que executam testes em um dispositivo real ou emulado (GOOGLE, 2023e). O Roboletric é capaz de fornecer algum controle sobre o comportamento do *framework* do Android e permite simular recursos e métodos nativos. Por exemplo, como o Roboletric consegue inflar a tela do aplicativo, é possível realizar testes de interação do usuário diretamente, sem a necessidade de *mocks*.[17]

Atualmente, o Robolectric não oferece suporte para testes de acessibilidade incorporados. Embora o *framework* anteriormente oferecesse suporte à API AccessibilityChecks, a partir da versão 4.5,[18] essa funcionalidade foi descontinuada devido a falhas em identificar os mesmos problemas que o Espresso era capaz de detectar.[19]

No entanto, apenas na versão 4.10,[20] lançada em abril de 2023, o Robolectric adicionou suporte para gráficos nativos do Android. Este suporte não é habilitado por padrão, mas quando ativado, as interações com as classes de gráficos do Android utilizam o código de gráficos nativo real do Android, proporcionando maior fidelidade (GOOGLE, 2023e).

[17] *Mocks* são imitações ou unidades falsas que simulam o comportamento de unidades reais
[18] <https://github.com/robolectric/robolectric/commit/24c8bf78bac77d95166edeb29b145c2e0abc3b1a>
[19] Ver comentários em <https://github.com/robolectric/robolectric/issues/5642>
[20] <https://github.com/robolectric/robolectric/releases/tag/robolectric-4.10>

2.5.4 Discussão sobre ferramentas disponíveis para Android

As ferramentas disponíveis para Android realizam as verificações de acessibilidade principalmente por meio da biblioteca ATF. Entre essas ferramentas, o Scanner de Acessibilidade é uma opção que não requer codificação. No entanto, uma desvantagem dessa abordagem é que o desenvolvedor precisa ativar a ferramenta em cada tela do aplicativo para obter os resultados desejados. Essa ação adicional implica na necessidade de explorar manualmente o aplicativo, o que pode não ser escalável para aplicativos maiores ou para testes frequentes.

Tanto o UI Automator quanto o Espresso são ferramentas utilizadas para automatizar testes. A principal diferença entre elas está no tipo de teste que cada um realiza. O UI Automator realiza testes caixa preta, o que significa que o testador não tem acesso à estrutura interna do código do aplicativo. Por outro lado, o Espresso realiza testes caixa branca, o que permite uma maior visibilidade e identificação da origem dos problemas.

Ambas as ferramentas são amplamente utilizadas por testadores. O UI Automator é a base para outras ferramentas como o MATE (ver Seção 3.3) e o *Appium*.[21] No entanto, é importante destacar que, com exceção do Espresso – que pode realizar testes médios, todas essas ferramentas realizam testes grandes, o que pode resultar em um *feedback* mais distante para os desenvolvedores.

Com o Espresso, os desenvolvedores têm a possibilidade de criar testes de acessibilidade de forma simples, pelo uso da API AccessibilityChecks, dentro do projeto de testes instrumentados. Porém, esses testes exigem o uso de um dispositivo físico ou emulado, o que pode torná-los mais lentos em comparação com testes locais.

O *Robolectric*, opção para criar testes locais com recursos do Android, chegou a oferecer suporte à API AccessibilityChecks. No entanto, como a funcionalidade foi descontinuada, cabe aos desenvolvedores criarem seus próprios testes de acessibilidade. Isso pode apresentar desafios, tais como falta de conhecimento técnico, limitações de tempo, ausência de requisitos claros e falta de interesse no assunto (LEITE *et al.*, 2021; GOMES; RIOS; RODRIGUES, 2020; BI *et al.*, 2022; SWALLOW; PETRIE; POWER, 2016).

No Quadro 4, são sumarizadas as ferramentas atualmente disponíveis para a realização de testes de acessibilidade no Android, destacando a quais testes cada uma delas se destina, e qual o esforço requerido do desenvolvedor ou testador para as verificações de acessibilidade.

2.6 Considerações finais

Neste capítulo, foram apresentadas as particularidades do desenvolvimento e testes para *mobile*. Além disso, foram apresentados conceitos do desenvolvimento ágil de software de particular interesse para essa pesquisa, os quais serão retomados nos capítulos seguintes.

[21] http://appium.io/docs/en/2.0/

Quadro 4 – Ferramentas para testar acessibilidade no Android

Ferramenta	Testes	Execução	Esforço requerido
Lint	Estáticos	Local	Inspeção automática em conjunto pré-definido de verificações de acessibilidade
Roboletric	Testes pequenos	Local	Escrita de testes para cada verificação de acessibilidade
Espresso	Testes médios Testes grandes	Instrumentado	Chamada para API *AccessibilityChecks* a partir de *View* raíz de cada tela
UI Automator	Testes grandes	Instrumentado	Automatização de testes de UI Pode usar ATF
Scanner de Acessibilidade	Teste grandes	Instrumentado	Execução e verificação do *app* tela a tela

Fonte: Elaborada pelo autor.

Em seguida, foram apresentados os conceitos de usabilidade e acessibilidade, e a forma como estes conceitos estão relacionados na literatura. Foi apresentado também o que existe atualmente para guiar os desenvolvedores na criação de aplicativos móveis acessíveis. Assim, foram descritos os conjuntos de diretrizes e recomendações para a acessibilidade Web, e as adequações sugeridas na literatura para a acessibilidade em dispositivos móveis.

Considerando o foco deste trabalho no desenvolvimento de aplicativos acessíveis nativos para Android, foram revisadas as ferramentas atualmente disponíveis para testes de acessibilidade nessa plataforma, e como cada uma delas pode ser utilizada nas diferentes etapas do desenvolvimento de aplicativos móveis.

A fim de identificar estratégias que contribuam para o desenvolvimento de aplicativos móveis acessíveis, bem como compreender o estado da arte em relação aos problemas de acessibilidade mais frequentes em dispositivos móveis e às técnicas de desenvolvimento e testes *mobile*, realizou-se uma revisão da literatura, que é apresentada no Capítulo 3.

CAPÍTULO

3

REVISÃO DA LITERATURA

Os avanços recentes na promoção da acessibilidade de software, em especial no domínio de aplicativos móveis, têm abrangido uma variedade de temas. Esses avanços compreendem tanto a disponibilização de tecnologias assistivas, como leitores e ampliadores de tela, que proporcionam aos usuários a superação de barreiras de acessibilidade, quanto a elaboração de normas e diretrizes que orientam a implementação da acessibilidade digital. Ferramentas destinadas a avaliar e melhorar a acessibilidade em produtos de software também têm surgido (SILVA; ELER; FRASER, 2018; LEITE *et al.*, 2021), bem como a implementação de leis que asseguram a inclusão de pessoas com deficiência (BRASIL, 2015; LAZAR, 2019).

Especificamente no contexto de aplicativos móveis para a plataforma Android, um conjunto de recursos e ferramentas de testes são disponibilizadas pela plataforma para auxiliar programadores na criação de softwares mais acessíveis, conforme discutido no Capítulo 2.

Este capítulo é dedicado à revisão da literatura, onde foram investigados os principais problemas de acessibilidade em aplicativos móveis e as abordagens de desenvolvimento e teste para lidar com elas. Primeiramente, é feita uma análise dos problemas de acessibilidade mais comuns em aplicativos Android, conforme evidenciado por estudos da literatura. Em seguida, é apresentado um mapeamento sistemático da literatura. Esse estudo foi conduzido para explorar técnicas de desenvolvimento que considerem a inclusão de requisitos de acessibilidade em aplicativos móveis durante o processo de desenvolvimento. Ainda no contexto de acessibilidade no processo de desenvolvimento, também são apresentadas as revisões de pesquisas que envolvem desenvolvedores do setor. Finalmente, são discutidas as ferramentas de teste de acessibilidade disponíveis, com particular ênfase nas que foram desenvolvidas a partir de pesquisas acadêmicas.

3.1 Problemas de acessibilidade

A despeito de todos os esforços encontrados na literatura, principalmente estabelecendo conjuntos de diretrizes e recomendações, a acessibilidade em aplicativos para dispositivos móveis ainda não é satisfatória. Mesmo entre os aplicativos mais populares disponíveis em lojas de aplicativos, parte significante dos elementos de interface do usuário não podem ser acessados, ou não podem ser facilmente utilizados por usuários com deficiência (YAN; RAMACHANDRAN, 2019; OLIVEIRA *et al.*, 2023; ALSHAYBAN; AHMED; MALEK, 2020).

Pessoas com deficiência utilizam recursos de T.A. para interagir com dispositivos móveis. Por exemplo, o leitor de telas é uma ferramenta de T.A. amplamente utilizada por pessoas com deficiência visual (SALEHNAMADI; HE; MALEK, 2023). A usabilidade dos aplicativos por esses usuários é diretamente influenciada pela compatibilidade desses softwares com tais ferramentas. Para garantir essa compatibilidade, é essencial que os elementos da interface do usuário nos aplicativos estejam devidamente rotulados e organizados. Assim, os leitores de tela conseguem interpretar corretamente e transmitir as informações necessárias aos usuários. Portanto, a acessibilidade dos aplicativos é crucial para que os usuários que dependem de leitores de tela possam usar esses aplicativos de forma eficaz.

Na Subseção 3.1.1, a seguir, é apresentada uma compilação da literatura dos principais problemas de acessibilidade enfrentados por pessoas com deficiência visual, e as respectivas recomendações de acessibilidade para remover essas barreiras. Em seguida, estudos da literatura avaliando o estado atual da acessibilidade de aplicativos Android são descritos na Subseção 3.1.2. Uma discussão sobre o cenário atual da acessibilidade em aplicativos Android em face às recomendações existentes é realizada na Subseção 3.1.3.

3.1.1 Recomendações de acessibilidade para usuários com deficiência visual

Estudos têm sido dedicados a identificar as principais barreiras enfrentadas por pessoas com deficiência visual ao interagir com aplicativos móveis (CARVALHO *et al.*, 2018; PARK; GOH; SO, 2014; FAJARDO-FLORES *et al.*, 2017). Alguns trabalhos (ACOSTA-VARGAS *et al.*, 2020; DIAS; DUARTE; FORTES, 2021) indicam a existência de problemas recorrentes, independente do tipo de aplicativo. Com o objetivo de superar essas barreiras, pesquisas têm explorado requisitos específicos para aplicativos móveis voltados para pessoas com deficiência visual (SIEBRA *et al.*, 2016; PANDEY *et al.*, 2022; BALLANTYNE *et al.*, 2018).

Com base em parte desses estudos, bem como nas diretrizes e recomendações de acessibilidade mencionadas na Seção 2.4, Dias, Duarte e Fortes (2021) compilaram os 13 problemas de acessibilidade mais comuns enfrentados por usuários com deficiência visual ao interagir com aplicativos móveis, como mostrado no Quadro 5.

Quadro 5 – Problemas de acessibilidade mais comuns em aplicativos móveis, compilados de estudos da Literatura

Id.	Problemas	Referências
$\mathbf{P}_1$	Contraste de texto e/ou imagem insuficiente	(BALLANTYNE *et al.*, 2018), (ACOSTA-VARGAS *et al.*, 2020), (ALSHAYBAN; AHMED; MALEK, 2020)
$\mathbf{P}_2$	Tamanho do alvo de toque inadequado	(BALLANTYNE *et al.*, 2018), (ACOSTA-VARGAS *et al.*, 2020), (ALSHAYBAN; AHMED; MALEK, 2020), (DAMACENO; BRAGA; MENA-CHALCO, 2018)
$\mathbf{P}_3$	Falta de rótulo no componente	(CARVALHO *et al.*, 2018), (BALLANTYNE *et al.*, 2018), (ALSHAYBAN; AHMED; MALEK, 2020), (DAMACENO; BRAGA; MENA-CHALCO, 2018)
$\mathbf{P}_4$	Descrição do elemento inadequada ou redundante	(ACOSTA-VARGAS *et al.*, 2020), (ALSHAYBAN; AHMED; MALEK, 2020), (DAMACENO; BRAGA; MENA-CHALCO, 2018)
$\mathbf{P}_5$	*Feedback* inadequado ou indisponível	(CARVALHO *et al.*, 2018), (BALLANTYNE *et al.*, 2018), (DAMACENO; BRAGA; MENA-CHALCO, 2018)
$\mathbf{P}_6$	Imagens e/ou ícones sem alternativa textual	(CARVALHO *et al.*, 2018), (BALLANTYNE *et al.*, 2018)
$\mathbf{P}_7$	Problemas e/ou limitação da Tecnologia Assistiva	(CARVALHO *et al.*, 2018), (BALLANTYNE *et al.*, 2018), (DAMACENO; BRAGA; MENA-CHALCO, 2018)
$\mathbf{P}_8$	Elementos de UI relacionados não agrupados	(ALSHAYBAN; AHMED; MALEK, 2020)
$\mathbf{P}_9$	Funcionalidade inexistente	(ALSHAYBAN; AHMED; MALEK, 2020)
$\mathbf{P}_{10}$	Organização de conteúdo inconsistente ou confuso	(CARVALHO *et al.*, 2018)
$\mathbf{P}_{11}$	Interação e/ou navegação confusa	(CARVALHO *et al.*, 2018), (DAMACENO; BRAGA; MENA-CHALCO, 2018)
$\mathbf{P}_{12}$	Apresentação padrão não adequada	(CARVALHO *et al.*, 2018)
$\mathbf{P}_{13}$	Estrutura de formulário com layout confuso	(BALLANTYNE *et al.*, 2018)

Fonte: Dias (2021).

Em seguida, mapearam cada um desses problemas para as principais recomendações e diretrizes de acessibilidade identificadas na literatura para usuários com cegueira ou baixa visão em dispositivos móveis. A compilação resultou em um total de 25 recomendações, que foram denominadas *mobile accessibility recommendations* (**maR**) . No Quadro 6, essas recomendações são associadas aos respectivos problemas de acessibilidade.

Dias (2021) também associa, a cada descrição de maR, as recomendações equivalentes da WCAG, da BBC e do SIDI, quando aplicáveis. Além disso, são citados os elementos de UI que devem ser considerados para cada recomendação.

A seguir, um exemplo da descrição e formalização de uma maR (DIAS, 2021):

- Descrição da **maR07** - um contraste adequado ajuda os usuários a identificar melhor o conteúdo do aplicativo. Deve-se utilizar uma relação de contraste de cores de pelo menos 4,5:1;
- **maR07** deve evitar $\mathbf{P}_1$ - Contraste insuficiente de texto e/ou imagem;

Quadro 6 – Problemas de acessibilidade ($\mathbf{P}_i$) em aplicativos móveis, relacionados com as 25 **maRs**

	Problema	**Recomendações aplicáveis**
$\mathbf{P}_1$	Contraste de texto e/ou imagem insuficiente	– **maR07** | Contraste de cor
$\mathbf{P}_2$	Tamanho do alvo de toque inadequado	– **maR17** | Tamanho do alvo de toque – **maR18** | Espaçamento
$\mathbf{P}_3$	Falta de rótulo no componente	– **maR10** | Rótulo de componentes
$\mathbf{P}_4$	Descrição do elemento inadequada ou redundante	– **maR24** | Ajuda contextual
$\mathbf{P}_5$	*Feedback* inadequado ou indisponível	– **maR09** | Eventos de tela – **maR19** | *Feedback* de ações – **maR23** | Confirmação – **maR25** | Relatório de erros
$\mathbf{P}_6$	Imagens e/ou ícones sem alternativa textual	– **maR01** | Conteúdo não-textual
$\mathbf{P}_7$	Problemas e/ou limitação de Tecnologia Assistiva	– **maR06** | Redimensionar conteúdo – **maR22** | Tecnologia Assistiva
$\mathbf{P}_8$	Elementos de UI relacionados não agrupados	– **maR15** | Agrupamento de elementos
$\mathbf{P}_9$	Funcionalidade inexistente	– **maR12** | Acionabilidade
$\mathbf{P}_{10}$	Organização de conteúdo inconsistente ou confusa	– **maR05** | Conteúdo extenso – **maR08** | Parágrafos – **maR20** | Formato de dados
$\mathbf{P}_{11}$	Interação e/ou navegação confusa	– **maR04** | Estrutura de visualização – **maR13** | Navegação com foco ordenável e visível – **maR14** | Título da página – **maR16** | Navegação no *app*
$\mathbf{P}_{12}$	Apresentação padrão não adequada	– **maR02** | Orientação do dispositivo – **maR03** | Características sensoriais – **maR11** | Modo teclado
$\mathbf{P}_{13}$	Estrutura de formulário com *layout* confuso	– **maR21** | Estrutura de formulário

Fonte: Dias (2021).

- **maR07** pode ser aplicado a todos os componentes da interface, fornecendo um contraste adequado para os usuários.

3.1.2 Acessibilidade atual em aplicativos Android

De acordo com Statista (2023), o número de aplicativos diferentes disponíveis na Google Play Store, desde dezembro de 2009, quando iniciou-se a aferição, até março de 2023, chegou a alcançar cerca de 3,5 milhões. Yan e Ramachandran (2019) avaliaram a acessibilidade em aplicativos móveis investigando as estruturas de UI e a conformidade com as diretrizes de acessibilidade de 479 aplicativos Android disponíveis na Google Play Store. Foi utilizado uma ferramenta automatizada, o IBM Mobile Accessibility Checker (MAC), para identificar problemas de acessibilidade, categorizados como violação, violação potencial ou aviso. Os resultados mostraram que 94.8% deles apresentam violações de acessibilidade, e 97.5% apresentam potenciais violações de acessibilidade. Além disso, identificaram que cinco categorias de widgets (TextView, ImageView, View, Button e ImageButton) representam 92% dos elementos de UI utilizados nas telas dos aplicativos avaliados e concentram cerca de 89% das violações de acessibilidade identificadas. Os problemas de acessibilidade mais encontrados nessas cinco categorias de *widgets* estão

associados a: a) falta de foco, b) falta de descrição do elemento, c) baixo contraste da cor do texto, d) falta de espaçamento suficiente entre os elementos e e) fontes e elementos de textos com tamanhos inferiores ao mínimo (YAN; RAMACHANDRAN, 2019).

Alshayban, Ahmed e Malek (2020) avaliaram mais de 1.000 aplicativos Android para entender o estado do suporte à acessibilidade nesses aplicativos. Os autores desenvolveram uma ferramenta de avaliação de acessibilidade que utiliza as verificações de acessibilidade fornecidas pela ATF para coletar onze diferentes tipos de problemas de acessibilidade em aplicativos Android. A avaliação da acessibilidade foi realizada com dois métodos principais. Primeiro, o Android Monkey, uma ferramenta de teste de interface do usuário, foi usado para simular interações do usuário e avaliar a acessibilidade dos aplicativos dinamicamente. Em seguida, um aplicativo Android chamado Listener monitorou eventos de acessibilidade, registrando capturas de tela e hierarquias de elementos de interface de usuário sempre que ocorria um evento. Eles também coletaram metadados, como nome do pacote, nome da atividade e número de elementos de interface do usuário e linhas de código para cada aplicativo. Os autores descobriram que quase todos os aplicativos apresentam problemas de acessibilidade, sendo os mais recorrentes relacionados a taxa de contraste de texto, tamanho da área de alvo de toque, taxa de contraste de imagem e elementos sem um texto pronunciável/falado.

O estudo de Oliveira *et al.* (2023) analisou avaliações de usuários com deficiências visuais ou condições oculares em aplicativos móveis. Os autores examinaram quase 180 milhões de avaliações de aplicativos populares disponíveis na Google Play Store para entender as preocupações específicas desses usuários em relação à acessibilidade. Destas, apenas 0,003% (aproximadamente 5 mil) eram comentários que expressavam preocupações de acessibilidade relacionadas a deficiências visuais ou condições oculares, associadas a 36 condições diferentes. Os principais temas identificados nestas avaliações de acessibilidade englobam aspectos como: fonte e tamanho do texto, suporte a leitores de tela, personalização e condições oculares específicas. Em relação aos componentes da interface, os elementos mais frequentemente mencionados são texto e fonte, seguidos por cor, fundo e botão.

3.1.3 Discussão sobre os problemas recorrentes

Embora nem todos os problemas de acessibilidade possam ser identificados por ferramentas tradicionais de teste de acessibilidade (SALEHNAMADI; HE; MALEK, 2023), as pesquisas envolvendo aplicativos Android sugerem que a maior parte dos problemas recorrentes de acessibilidade, em particular os que afetam usuários com deficiência visual, são de detecção e correção relativamente simples (ALSHAYBAN; AHMED; MALEK, 2020). Por exemplo, ajustes no contraste de texto podem ser efetuados com pequenas modificações nas cores da fonte e/ou do fundo.

No Quadro 7 são relacionados seis desses problemas de acessibilidade mais recorrentes nos aplicativos Android avaliados, associados às respectivas recomendações de acessibilidade

(maRs). Além disso, visando auxiliar os desenvolvedores de modo mais pragmático, e considerando reduzir os problemas de acessibilidade nos elementos de UI mais utilizados, para cada recomendação foram endereçados os *widgets* Android associados, bem como quais atributos devem ser alvos de verificações de acessibilidade nestes elementos.

Quadro 7 – Problema de acessibilidade e recomendações para os widgets mais usados

Problema	**Recomendação / Requisito**	**Widgets**	**Atributos**
Contraste de texto insuficiente	Deve-se utilizar uma taxa de contraste de pelo menos 4.5:1 (maR7)	TextView, Button	TextColor, Background Color
Tamanho do alvo de toque inadequado	Alvo de toque deve ter pelo menos 48x48dp (maR17)	Button, ImageButton	Width, Height
Falta de rótulo no componente	Para cada controle de formulário, deve existir um TextView com o atributo labelFor associado, ou deve ter o atributo hint fornecido (maR10)	*EditText*, CheckBox, RadioButton, Switch	labelFor, hint
Imagens e/ou ícones sem alternativa textual	Todo conteúdo não textual, como imagens e conteúdo de mídia, deve possuir descrição alternativa em texto (maR1)	ImageButton, ImageView	contentDescription
Interação e/ou navegação confusa	Aplicativo deve suportar navegação baseada em foco (maR13)	Accessibility NodeInfo	isAccessibility Focused
Falta de espaçamento suficiente entre os elementos	Componentes de interação devem ter um espaçamento mínimo de 8dp entre si e das bordas da tela (maR18)		getTranslationX, getTranslationY, getHeight, getWidth

Fonte: Elaborada pelo autor.

3.2 Acessibilidade e usabilidade no processo de desenvolvimento de aplicativos móveis

Embora seja considerada um atributo de qualidade de software, a usabilidade não é um tópico específico da Engenharia de Software, mas considerada como um termo, cujo conceito é central na área de conhecimento em IHC (CORONEL *et al.*, 2011). Estudos apontam para os benefícios de se incorporar a usabilidade no processo de desenvolvimento (PEISCHL; FERK; HOLZINGER, 2015; VÄÄTÄJÄ; KOPONEN; ROTO, 2009).

De mesmo modo, metas e recursos de acessibilidade devem ser inseridos no *design* do aplicativo o mais cedo possível, com um custo relativamente inferior ao da modificação do produto final para torná-lo acessível (ISO 9241-171, 2008).

No contexto da Web, estudos (DIAS, 2014; MAIA; TURINE; PAIVA, 2010) já propuseram métodos/modelos para inserir requisitos de acessibilidade nas etapas do processo de desenvolvimento. Por exemplo, Maia, Turine e Paiva (2010) sugere que, na etapa de Construção de Software, sejam realizadas tarefas de: planejar teste de acessibilidade para cada unidade de software; codificar cada unidade de software de acordo com as técnicas de acessibilidade; e executar testes de acessibilidade de cada unidade de software.

Buscando identificar técnicas ou processos para desenvolvimento de aplicativos móveis acessíveis, e para melhor compreender o estado da arte no que se refere ao desenvolvimento *mobile* foi conduzido um Mapeamento Sistemático da literatura (KITCHENHAM; BUDGEN; BRERETON, 2010), apresentado na subseção a seguir. Esta escolha foi feita por ser o Mapeamento Sistemático mais indicado para obter, segundo Kitchenham e Charters (2007a), uma visão geral do tópico de pesquisa.

3.2.1 Mapeamento Sistemático

O principal interesse para execução deste mapeamento foi conhecer as abordagens utilizadas no desenvolvimento *mobile*, considerando aspectos de acessibilidade e usabilidade.

As informações gerais que nortearam o mapeamento estão descritas, brevemente, no Quadro 8. A seguir são detalhados a questão de pesquisa e os critérios de busca e seleção de estudos.

Quadro 8 – Informações gerais do Mapeamento Sistemático realizado

Informações gerais	
Título	Mapeamento sistemático sobre processos de desenvolvimento para *mobile* considerando acessibilidade/usabilidade
Descrição	Esta pesquisa visa identificar como requisitos de acessibilidade são considerados nos processos de desenvolvimento para *mobile*
Objetivo	Investigar processos de desenvolvimento para aplicativos *mobile* que considerem acessibilidade e usabilidade

Fonte: Elaborada pelo autor.

Questão de pesquisa

Um estudo realizado por meio de mapeamento, geralmente, possui questões de pesquisas mais abrangentes (KITCHENHAM; CHARTERS, 2007a), já que o objetivo é identificar o que tem sido discutido na literatura a respeito do tópico de pesquisa.

Para a realização deste mapeamento, foi escolhida uma adaptação da abordagem **PICo** (LOCKWOOD; MUNN; PORRITT, 2015), uma revisão do conjunto de critérios PICO (População, Intervenção, Comparação e Resultados), derivado da área de conhecimentos em Medicina (PAI *et al.*, 2004; FELIZARDO *et al.*, 2017). A PICo é dirigida a estudos qualitativos, e determina os critérios como:

- **População** ou **Problema**: as características da população; ou o problema, condição ou doença de interesse.
- **Interesse**: refere-se a um evento, atividade, experiência ou processo definido.

- **Contexto**: configuração, características distintas ou especificidade do(s) objeto(s) de interesse.

A correspondência dos objetivos desta pesquisa ao conjunto PICo, para condução do mapeamento, é apresentada no Quadro 9. Com base nesses critérios PICo, foi também definida a seguinte **questão de pesquisa**:

> *– Como requisitos de acessibilidade e usabilidade são contemplados durante o processo de desenvolvimento mobile?*

Quadro 9 – Identificação de critérios PICo

PICo	
População ou problema	Processo de desenvolvimento
Interesse	Requisitos de acessibilidade e usabilidade
Contexto	*Mobile*

Fonte: Elaborada pelo autor.

Busca e seleção de estudos

Para os objetivos deste mapeamento, e para a estratégia de busca desejada, "acessibilidade" e "usabilidade" foram considerados sinônimos, em razão das relações apresentadas na Seção 2.4.

Quanto a "processo de desenvolvimento", as expressões equivalentes foram selecionadas com a contribuição de pesquisadores da área de Engenharia de Software e pela combinação de estudos secundários por eles indicados (IDRI; AMAZAL; ABRAN, 2015; GIUFFRIDA; DITTRICH, 2013).

Após a aplicação dos operadores lógicos, a ***string* de busca** foi formulada como a seguir:

```
("mobile") AND ("development process" OR "requirement engineering"
OR "requirement specification" OR "software development"
OR "software engineering" OR "software process*"
OR "software quality" OR "software testing")
AND ("accessibility" OR "usability")
```

Para atender aos objetivos deste estudo, o seguinte **critério de inclusão** foi aplicado:

– Estudo primário, escrito em inglês ou português, com texto completo disponível, que descreve processo de desenvolvimento para *mobile* e considera aspectos de acessibilidade

Com relação aos critérios de exclusão, eles foram definidos pelo não atendimento do critério de inclusão. Também foram incluídos critérios de exclusão gerais, encontrados na literatura (FELIZARDO *et al.*, 2017). Assim, os **critérios de exclusão** adotados foram:

1. Estudo não primário
2. Estudo primário não escrito em inglês ou português.
3. Estudo primário que aborda acessibilidade ou usabilidade fora do contexto de processo de desenvolvimento.
4. Estudo primário que não aborda acessibilidade ou usabilidade.
5. Estudo primário que não aborda desenvolvimento *mobile*.
6. Estudo primário que não descreve o processo de desenvolvimento.
7. Estudo primário sem o texto completo disponível.
8. Versão resumida de um mesmo estudo.

No Quadro 10 são listadas as seis bases bibliográficas selecionadas para as buscas, com respectivas URLs de acesso. Essas bases bibliográficas estão entre as mais relevantes fontes na área de Ciência da Computação (GARCéS *et al.*, 2017), (KITCHENHAM; CHARTERS, 2007b), (DYBå; DINGSøYR; HANSSEN, 2007).

Quadro 10 – Bases bibliográficas selecionadas

Fonte de informação	URL
ACM Digital Library	<http://portal.acm.org>
IEEE Digital Library	<http://ieeexplore.ieee.org>
ISI Web of Science	<http://www.isiknowledge.com>
Science@Direct	<http://www.sciencedirect.com>
Scopus	<http://www.scopus.com>
Springer Link	<http://link.springer.com>

Fonte: Elaborada pelo autor.

Condução

Para a execução das buscas, algumas adequações foram necessárias para *Scopus* e *SpringerLink*, devido a peculiaridades do funcionamento destas bases, mas sem afetar a lógica das buscas. Outra decisão do estudo foi incluir restrições **por domínio (*Computer Science*)** nas bases em que havia essa opção de filtro, e também por período (**publicações a partir de 1999**, ano de publicação da WCAG 1.0 (DARDAILLER, 2009)). Mesmo com essas restrições, a primeira execução das buscas em todas as bases retornou um total de 3.493 estudos.

Após análise superficial deste primeiro conjunto de estudos retornados nas primeiras buscas, verificou-se que as buscas, como formuladas, estavam retornando um grande número de trabalhos relacionados com a área de infraestrutura de redes móveis, e não pertenciam ao

escopo deste estudo. Após conferência entre os pesquisadores, foi incluído um critério adicional de negação à *string* de busca. A nova versão ficou assim[1]:

```
("mobile") AND ("development process" OR "requirement engineering"
OR "requirement specification" OR "software development"
OR "software engineering" OR "software process*"
OR "software quality" OR "software testing")
AND ("accessibility" OR "usability")
NOT ("networks" OR "wireless")
```

Uma amostra de 100 estudos já classificados na busca anterior foi utilizada para comparação, e os estudos pré-selecionados permaneciam presentes na nova busca, ajudando a suportar a decisão de projeto. Com a inclusão dessas restrições, e reaplicando as buscas nas bases, o número de estudos obtidos diminuiu para 2.167. Desses, 1.005 eram da base *SpringerLink*, onde o mecanismo de busca apresenta algumas peculiaridades, e muitos estudos retornados destoavam do escopo do mapeamento. Por essa razão e por decisão de pesquisa, especificamente nesta base foi aplicada uma lógica adicional, restringindo os resultados para publicações que possuíssem *"mobile"*, *"accessibility"* ou *"usability"* entre as palavras-chave. Com mais essa restrição, o total de estudos obtidos foi **1.531**. Os totais de estudos por base, após cada restrição adicionada, são apresentados na Tabela 1.

Tabela 1 – Totais de estudos por base em cada busca

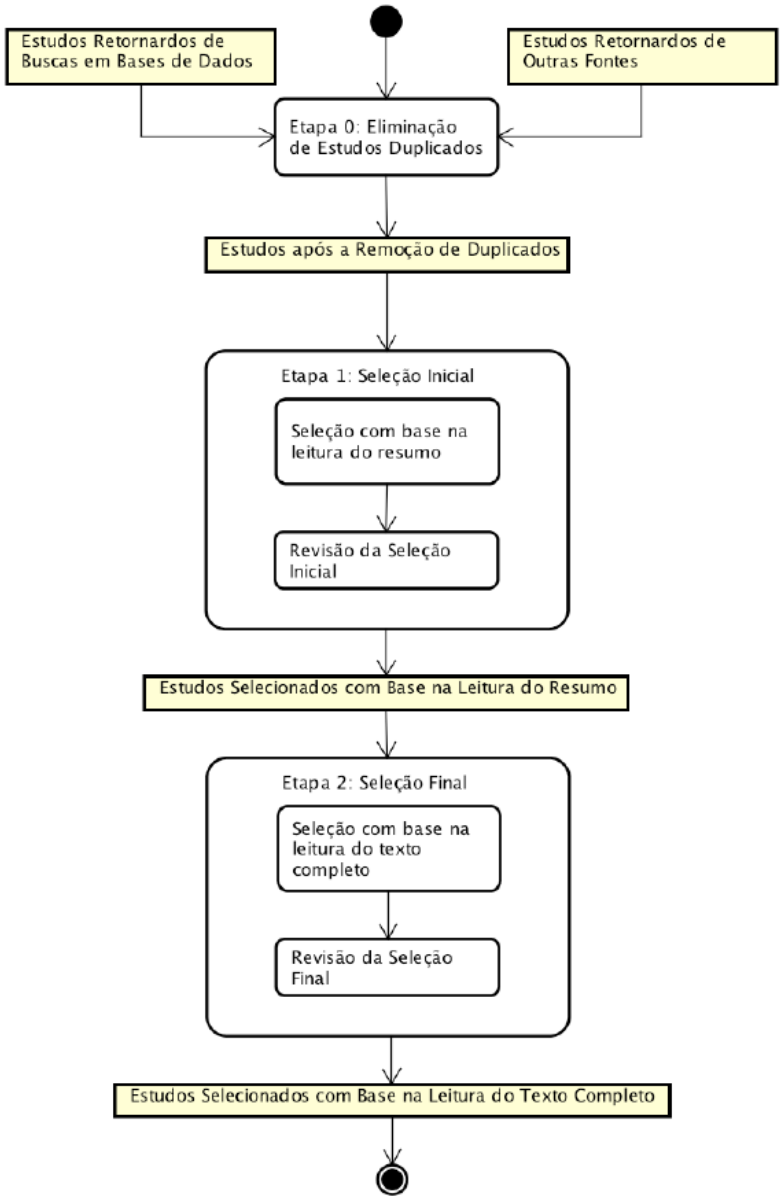
Base de busca	1ª busca	2ª busca	3ª busca
ACM Digital Library	586	338	338
IEEE Digital Library	328	239	239
ISI Web of Science	256	246	246
Science@Direct	11	11	11
Scopus	1.411	328	328
Springer Link	901	1005	369
Total	**3.493**	**2.167**	**1.531**

Fonte: Dados da pesquisa.

Todas as buscas descritas foram realizadas no mês de dezembro de 2019. As etapas de seleção dos estudos seguiram o fluxo representado na Figura 7. Para auxiliar a organização, o acesso, a identificação de duplicados e as tarefas de seleção dos estudos foram utilizadas em conjunto as ferramentas *Mendeley*[2] e *Parsif.al*[3]. O conteúdo do relatório gerado pela ferramenta pode ser visto no Apêndice B.

[1] Para a base *Science@Direct* foi necessário incluir um AND antes do NOT
[2] Disponível em mendeley.com
[3] Disponível em parsif.al

Figura 7 – Etapas da seleção de estudos do Mapeamento Sistemático.



Fonte: Felizardo *et al.* (2017).

Após a remoção de 256 estudos duplicados, foram selecionados **1.275** estudos para a etapa de seleção inicial, com base na leitura de títulos e resumos. Posteriormente, **115** estudos foram escolhidos para a seleção final e leitura dos textos completos. Após a leitura completa, critérios de exclusão foram atribuídos a outros 69 estudos, resultando em **46** estudos selecionados para a extração de dados.

Extração de dados

O objetivo da extração de dados é registrar informações que os pesquisadores obtêm do conteúdo dos estudos principais (KITCHENHAM, 2004). Para isso, com auxílio da ferramenta *Parsif.al*,

foi elaborado um formulário contendo campos abertos, multivalorados e booleanos. Informações básicas dos estudos, como título, autores, palavras-chave e resumo, não foram incluídas no formulário pois já são vinculadas a cada estudo pela própria ferramenta.

Uma dificuldade para a confecção deste questionário foi a definição das denominações, taxonomias e escolha das abordagens a serem mapeadas. Como exemplo, a abordagem UCD pode ser considerada uma prática, um campo, uma habilidade, um *framework*, uma filosofia, uma disciplina ou um método (WILLIAMS, 2009). Também não eram conhecidas pelos pesquisadores todas as possibilidades de abordagens utilizadas. Neste sentido, foram tomadas algumas importantes decisões de projeto.

1. Ao invés de pré-determinar as abordagens a serem verificadas nos estudos, optou-se por coletar os nomes das abordagens tal como citadas nos estudos.
2. Uma tabela de equivalência foi utilizada para modos diferentes de referência a uma mesma abordagem (exemplo: "*Model-driven desing*", "*Model-based design*", "*Model-driven architecture*" e "*Model-driven approach*").
3. Adicionalmente, foram registradas as denominações com que estas abordagens eram apresentadas nos estudos (exemplos: processo, modelo, método, metodologia, *framework*, etc.).
4. Para suportar os itens 1 e 3, durante as fases de seleção dos estudos (ver início desta Seção 3.2.1), foram anotados os nomes das principais abordagens e denominações mencionadas nos estudos, que foram incluídos como opções de seleção no formulário de extração.
5. Abordagens não incluídas no formulário ou muito específicas de alguns estudos foram anotadas em um campo texto adicional.
6. A mesma estratégia foi utilizada para registrar as expressões de acessibilidade e usabilidade tais como encontradas nos estudos. Barreiras, *guidelines*, requisitos e padrões são alguns exemplos.
7. Tanto para as abordagens quanto para as expressões de acessibilidade/usabilidade, as anotações foram registras em inglês, idioma de 45 dos 46 estudos selecionados, para evitar erros ou divergências de traduções.

Todas as informações e perguntas que compuseram o formulário de extração de dados, assim como as opções dos campos multivalorados, podem ser visualizadas na Tabela 2.

Tabela 2 – Campos do formulário de extração de dados

Descrição	Valores	Tipo
Base de busca	ACM, IEEE, ISI Web of Science, Science@Direct, Scopus, Springer Link	Seleção múltipla
Tipo de estudo	Qualitativo, Quantitativo	Seleção múltipla

Continua na próxima página

Tabela 2 – Campos do formulário de extração de dados (continuação da página anterior)

Descrição	Valores	Tipo
Metodologia(s) de pesquisa utilizada(s)	*Design Science Research*, Estudo de campo, Estudo de Caso, Estudo empírico, Estudo piloto, Levantamento Bibliográfico, Prova de conceito, *Survey*, Teste com usuário	Seleção múltipla
Identificação / nome da abordagem		Campo texto
Breve descrição da proposta		Campo texto
Denominação(ões) utilizada(s) para a proposta apresentada	Abordagem, *Checklist*, Ferramenta automatizada, *Framework*, *Guidelines*, Método, Metodologia, Modelo, Processo, Técnica	Seleção múltipla
Processo(s) da norma ISO 12207 apoiado(s) pela abordagem	Implementação, Análise de Requisitos, Projeto de Arquitetura, Projeto Detalhado, Construção, Integração, Testes	Seleção múltipla
Filosofias, paradigmas, metodologias e/ou processos abordados	*Agile Software Development* (ASD), *Aspect-Oriented Programming* (AOP), *Component-based Development* (CBD), *Design patterns*, *Extreme Programming* (XP), *Incremental process model*, *Iterative process model*, *Lean Software Development*, *Model-driven Development* (MDD), *Problem-solving paradigm*, *Rapid Application Development* (RAD), *Rational Unified Process* (RUP), *Scenarios-based*, *Scrum*, *Software development tool*, *Spiral Model*, *Task analysis*, *Test-driven Development* (TDD), *Usability-driven model*, *Usability Engineering* (UE), *Usability Pattern*, *User-centered Design* (UCD), *User eXperience* (UX), *User interface design pattern* (UIDP), *Waterfall model*	Seleção múltipla
Que outras abordagens são citadas?		Campo texto
Quanto à acessibilidade / usabilidade, aborda	*Accessibility barriers*, *Accessibility features*, *Accessibility guidelines*, *Accessibility issues*, *Accessibility requirements*, *Accessibility standards*, *Accessibility techniques*, Android/iOS *guidelines*, *Generic usability improvements*, ISO 9241, *Usability attributes*, *Usability evaluations*, *Usability goals*, *Usability guidelines*, *Usability patterns*, *Usability principles*, *Usability problems*, *Usability properties*, *Usability requirements*, *Usability techniques*, *Usability testing*	Seleção múltipla
Quanto à acessibilidade, é direcionada para	Deficiência auditiva, Deficiência cognitiva, Deficiência motora, Deficiência visual, Idosos, Aborda somente usabilidade acessibilidade, Outra	Seleção múltipla
Quanto à acessibilidade, é direcionada para (especifique)		Campo texto
Como acessibilidade é considerada na etapa do desenvolvimento *mobile*?		Campo texto

Continua na próxima página

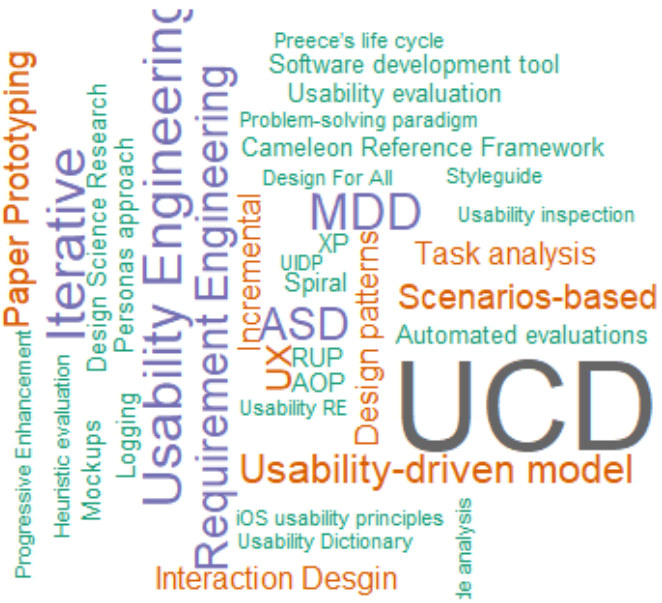
Tabela 2 – Campos do formulário de extração de dados (continuação da página anterior)

Descrição	Valores	Tipo
A acessibilidade tratada foi validada?		Booleano
Existe alguma implementação na abordagem descrita?		Booleano
Foi feita alguma validação com desenvolvedores?		Booleano
Domínio no qual a proposta foi apresentada/experimentada		Campo texto
Plataforma de desenvolvimento	Android, iOS, Híbrido, Web, Não especifica, Outra	Seleção múltipla

Resultados

O principal interesse para a execução deste mapeamento foi conhecer as abordagens utilizadas no desenvolvimento *mobile*, considerando aspectos de acessibilidade e usabilidade. A forma gráfica escolhida para apresentar a ocorrência de cada abordagem mapeada nos estudos foi a nuvem de *tags*. Na Figura 8 é possível notar a prevalência da abordagem de UCD, citada em 22 dos 46 estudos selecionados (quase 50%).

Figura 8 – Nuvem de *tags* das abordagens considerando as ocorrências nos estudos



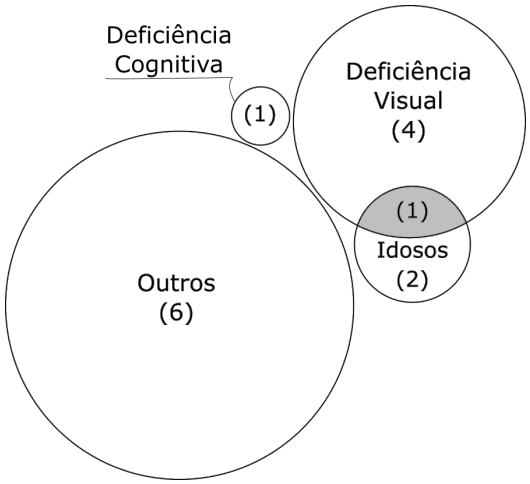
Fonte: Elaborada pelo autor.

Um importante recorte foi feito durante o mapeamento, por meio de uma questão da extração de dados que pretendia identificar o tipo de deficiência abordado pela proposta apresentada. Ao filtrar as respostas a essa questão, verificou-se que 32 dos 46 estudos abordavam

apenas usabilidade (quase 70%). Já os outros 14 estudos abordavam diretamente aspectos de acessibilidade, e por isso mereceram uma atenção especial para a síntese deste mapeamento.

Na Figura 9 está representada a distribuição desses 14 estudos em conjuntos (diagramas de *Venn*) das publicações por tipo de deficiência que mencionam. Deficiência visual (5 publicações) e idosos (3) foram os mais abordados, sendo que um dos estudos aborda ambos os tipos de deficiência (HOLZINGER; SAMMER; HOFMANN-WELLENHOF, 2006). A deficiência cognitiva aparece em um estudo direcionado ao aprendizado de crianças com dislexia (ARTEAGA; RIVERA, 2018). Já nos outros seis trabalhos são seguidas diretrizes de acessibilidade sem especificar o tipo de deficiência.

Figura 9 – Tipos de deficiências abordadas pelos estudos que consideram acessibilidade



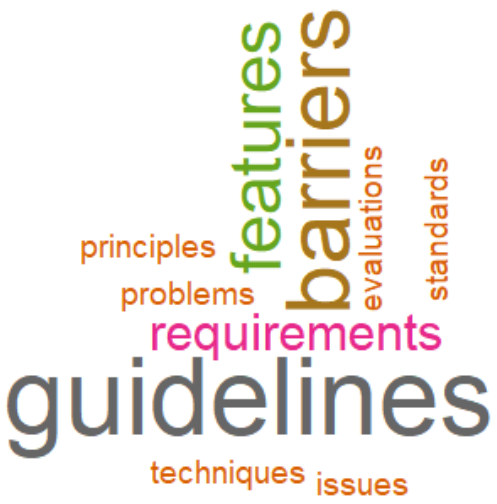
Fonte: Elaborada pelo autor.

A Figura 10 apresenta uma nuvem de *tags* das ocorrências de expressões de acessibilidade e usabilidade nestes 14 trabalhos. Assim, o termo *guidelines* é notadamente o mais recorrente. As demais expressões (*barriers*, *features* e *requirements*) são também reconhecidas como artefatos e aspectos com alto nível de abstração, sugerindo uma possível tendência de pesquisas com poucas intervenções práticas, orientadas a desenvolvedores.

Quanto às abordagens utilizadas nesses estudos, ainda há prevalência de UCD, presente em nove dos quatorze estudos (quase 65%). Também destacam-se o Desenvolvimento Dirigido a Modelos (*Model-driven Development* (MDD)) e a Engenharia de Requisitos, abordadas em quatro estudos cada.

Para facilitar uma visão geral sobre como a acessibilidade foi considerada nesses 14 trabalhos, na Tabela 3 são resumidas as estratégias adotadas em cada estudo.

Figura 10 – Nuvem de *tags* das expressões usadas referentes à acessibilidade



Fonte: Elaborada pelo autor.

Tabela 3 – Acessibilidade considerada nas estratégias selecionadas

Identificação da abordagem	Como a acessibilidade é considerada
1. Processo de desenvolvimento de aplicativos centrado no usuário (HOLZINGER; SAMMER; HOFMANN-WELLENHOF, 2006)	Requisitos de acessibilidade obtidos na fase de levantamento de requisitos por meio de entrevistas com especialistas.
2. Quick Accessibility Evaluation (QAC) - 15 métricas pré-definidas para coletar problemas de acessibilidade (FEINER; KRAINZ; ANDREWS, 2018)	*Checklist* de acessibilidade a ser executado com auxílio de outras ferramentas, como o Accessibility Scanner (Android). Critérios definidos pela intersecção do *Material Accessibility Design Guidelines* (GOOGLE, 2023a) e da WCAG. Os problemas encontrados são documentados em formato que facilita a visualização pelos desenvolvedores.
3. Modelo de processo para desenvolvimento de aplicativos educacionais para crianças com dislexia (ARTEAGA; RIVERA, 2018)	– No levantamento de requisitos, métodos de UCD são utilizados com especialistas e usuários para elaboração de *guidelines* de usabilidade. – No design da aplicação são usados utilizados *sketches* e identificados padrões de projeto para atender às *guidelines*. – Na prototipação, especialistas e usuários voltam a participar, selecionando as melhores alternativas. – Na etapa final, são realizadas sessões de treinamentos.
4. Método de Engenharia de Requisitos para ser usado com participantes cegos (TUUNANEN; PEFFERS; HEBLER, 2010)	Indiretamente, pela inclusão dos participantes cegos na engenharia de requisitos.
5. MDD para criação de *app scaffold* para pessoas com deficiência visual no domínio de aplicativos de transporte (KRAINZ; FEINER; FRUHMANN, 2016)	– Menciona que acessibilidade é inserida no modelo, baseada em "melhores práticas" e "padrões comuns" (WCAG é citada como referência). – Após o processo de transformação do metamodelo, obtém-se um *app scaffold* com recursos de acessibilidade. Não apresenta exemplos.

Continua na próxima página

Tabela 3 – Acessibilidade considerada nas estratégias selecionadas (continuação da página anterior)

Identificação da abordagem	Como a acessibilidade é considerada
6. MDD para criação de *chats mobile* acessíveis (CALVO; IGLESIAS; MORENO, 2014)	– Na etapa de Engenharia de Requisitos, os requisitos de acessibilidade são considerados requisitos tangenciais e são estendidos a todos os requisitos categorizados. – São utilizadas técnicas como *Brainstorming*, Personas, Cenários e alguns padrões e *guidelines*. – No *design* conceitual, com MDD, utiliza *User Interface Description Language* (UIDL) baseado no *Cameleon Reference Framework*.
7. Acessibilidade para *mobile* web em plataforma de aprendizado (ZHOU; XIA, 2010)	– Para a etapa de "análise prévia", sugere a utilização de questionários, observação e entrevista. – Para a etapa de "planejamento", divide em planejamento do tema, planejamento do conteúdo de estudo e planejamento de navegação. – Na implementação apenas cita que os designers devem respeitar estritamente os padrões de acessibilidade e princípios de usabilidade.
8. *S4S Medication Assistant* (FERREIRA *et al.*, 2014)	– Um processo iterativo de prototipação é realizado e cada protótipo é avaliado por usuários seguindo uma método de avaliação composto de três fases: validação conceitual, teste de protótipo e teste piloto. – As melhorias em acessibilidade e usabilidade em tese viriam, de modo indireto, da participação dos usuários nesse processo.
9. Estudo de caso de identificação de barreiras de acessibilidade web em dispositivos móveis (SAKAMOTO; SILVA; MIRANDA, 2012)	Apresenta 10 exemplos e barreiras de acessibilidade para *mobile*, e depois apresenta boas práticas e soluções.
10. MoCoMed-Graz: UCD e Desenvolvimento (HOLZINGER; SAMMER; HOFMANN-WELLENHOF, 2006)	Benefícios esperados pelo uso dos métodos UCD, como *feedback* antecipado nos protótipos de baixa fidelidade, e mais realísticos nos protótipos de alta fidelidade, entre outros.
11. Geração semi-automática de UI acessível com MDD (BOURAOUI; GHARBI, 2019)	Por meio de um metamodelo de UI acessível, baseado em *guidelines* da literatura.
12. *Ways4All* (KRAJNC; FEINER; SCHMIDT, 2010)	– Não explicita barreiras superadas ou requisitos atendidos, apenas faz a análise dos usuários, que no exemplo do experimento eram pessoas com deficiência visual.
13. Processo de desenvolvimento de interação do *LeVer* (*m-Learning* para idosos) (REITHINGER; RUSS; SCHUMACHER, 2015)	– Propõe um "guia de estilo" para guiar a construção da UI com *guidelines* apropriados. – Quanto à acessibilidade, analisou dois conjuntos de *guidelines*: de Jaeger e Xie (2009) e da W3C
14. *Chats* acessíveis em *m-Learning* com UCD (CALVO; IGLESIAS; MORENO, 2014)	– Métodos de IHC (basicamente métodos UCD) são misturados com os de Engenharia de Software em todas as fases do desenvolvimento. Padrões e *guidelines* são incluídos na fase de elicitação de requisitos. – Do ponto-de-vista da acessibilidade, foram seguidas as *guidelines* da WCAG 2.0. Além disso, foram consideradas as práticas sugeridas na MWABP e MWBP 1.0. Também é citado um conjunto de *guidelines* que provê especificações de acessibilidade para ferramentas CSCL (CONSORTIUM, 2018).

Fonte: Elaborada pelo autor.

Discussão

O maior número de estudos selecionados (32), que abordam exclusivamente usabilidade, em comparação com aqueles que tratam de acessibilidade (14), deve-se à estratégia adotada nas buscas, que incluiu tanto "usabilidade" quanto "acessibilidade" na *string* de busca. A inclusão

destes trabalhos foi importante para estabelecer um mapa mais abrangente das abordagens utilizadas integrando IHC e Engenharia de Software.

Um exemplo de abordagem recorrente nesses estudos, embora menos registrada nos estudos que abordam acessibilidade, é o Desenvolvimento Ágil de Software (ASD - *Agile Software Development*). Losada *et al.* (2013) apresentam uma proposta de integração de técnicas de engenharia de usabilidade com o processo de desenvolvimento ágil. É utilizada uma metodologia chamada *InterMod* para avaliação de usabilidade desde as fases iniciais do desenvolvimento. Os autores apresentam três principais estratégias no processo: uso de protótipos de papel, avaliação contínua da usabilidade por meio de heurísticas e análise de *logs* de interação do usuário.

Wolkerstorfer *et al.* (2008) propõem um "processo de usabilidade ágil", no qual combinam as vantagens da metodologia XP, como ciclos curtos de entrega e estreita integração com o cliente, com instrumentos de usabilidade inspirados em práticas de UCD. Hess *et al.* (2013) descrevem um método centrado no usuário, baseado em um *framework* de requisitos orientados a tarefa, chamado *mConcAppt*. O método é similar ao processo Scrum e provê conceitos de interação testáveis após cada iteração.

Dentre os estudos que abordam acessibilidade, descritos na Tabela 3, apenas um menciona ASD (KRAJNC; FEINER; SCHMIDT, 2010), combinados com métodos UCD, como Personas e protótipos de papel. Porém, não descreve o processo formalmente.

A maior parte dos estudos selecionados apresenta abordagens, envolvendo UCD, com apoio às etapas de Análise de Requisitos e Projeto Arquitetural do software. Porém, foram identificados poucos estudos com abordagens que apoiem as etapas de Construção e Testes, em relação ao que era esperado pelos pesquisadores, sobretudo para etapa de Testes. Esse número baixo pode estar relacionado com as estratégias escolhidas para o mapeamento, e também podem sinalizar uma carência na formalização dos testes nos processos que consideram acessibilidade/usabilidade. Por exemplo, durante as etapas de seleção, alguns estudos foram excluídos por apresentarem apenas uma avaliação de usabilidade como uma validação de um produto apresentado, mas sem formalização da avaliação e sem vinculá-la ao processo de desenvolvimento, quando descrito.

Uma outra observação relativa aos dados extraídos dos artigos estudados, e selecionados, foi o baixo número de estudos que apresentaram uma validação da proposta com desenvolvedores. Apenas quatro dos 46 estudos selecionados (~9%), e apenas um dentre os que abordam acessibilidade.

Finalmente, os dados coletados nessa revisão da literatura indicam a importância de estabelecer estratégias para o desenvolvimento de aplicativos móveis acessíveis, colocando o desenvolvedor em um papel central. Essas estratégias precisam apoiar os desenvolvedores de

forma pragmática, ou seja, oferecer assistência durante o processo de criação, com orientações práticas sobre soluções tecnológicas voltadas para a acessibilidade.

3.2.2 Estudos com desenvolvedores

A implementação efetiva de requisitos de acessibilidade no desenvolvimento de software é um desafio contínuo na indústria de tecnologia. Apesar da ampla disponibilidade de recursos relacionados à acessibilidade, como tecnologia assistiva, diretrizes, legislação e ferramentas, muitos desenvolvedores ainda enfrentam dificuldades para incorporar a acessibilidade em seus produtos (BI *et al.*, 2022; LEITE *et al.*, 2021).

Pesquisas têm buscado entender por que *designers* e desenvolvedores não implementam produtos acessíveis (LEITE *et al.*, 2021), especialmente no contexto da Web (FREIRE; RUSSO; FORTES, 2008; SWALLOW; PETRIE; POWER, 2016; OLIVEIRA; ELER, 2017; INAL; RıZVANOğLU; YESILADA, 2019). Um estudo realizado com profissionais de Experiência do usuário (**UX** - ***U**ser e**X**perience*) na Turquia (INAL; RıZVANOğLU; YESILADA, 2019) revelou que, embora acreditem ter treinamento suficiente em acessibilidade na Web, muitos não estão familiarizados com os padrões e as tecnologias assistivas usadas por pessoas com deficiências, e raramente consideram a acessibilidade na Web em seus projetos. Eles acreditam que é responsabilidade dos gerentes de projeto tornar as aplicações Web acessíveis.

Swallow, Petrie e Power (2016) exploraram os desafios que os desenvolvedores Web enfrentam ao criar sites acessíveis. A pesquisa revelou que, embora os desenvolvedores estejam interessados em acessibilidade na Web, muitos lutam para desenvolver sites acessíveis devido à falta de conhecimento e orientação prática.

No Brasil, um estudo de Antonelli *et al.* (2018) revelou que, apesar das várias iniciativas para promover a acessibilidade na Web, muitos desenvolvedores ainda não estão engajados com essa questão. A pesquisa, que contou com 404 participantes de todo o Brasil, mostrou que a maioria nunca desenvolveu um site acessível e 33,2% não estão preocupados com a acessibilidade em futuros projetos. Os autores enfatizam a necessidade de políticas públicas para melhorar a conscientização sobre acessibilidade na Web e garantir a conformidade com a legislação brasileira.

Em um estudo que aborda a acessibilidade no contexto de desenvolvimento de software em geral, Bi *et al.* (2022) coletaram dados de 15 entrevistas e 365 respondentes de um questionário. O perfil dos participantes foi bastante diversificado, incluindo programadores, testadores, *designers* e gerentes. A pesquisa revelou que a maioria dos profissionais não tem experiência de trabalho em acessibilidade e que a acessibilidade geralmente é considerada apenas em um estágio tardio de desenvolvimento ou durante a refatoração e manutenção do sistema.

A maneira como a acessibilidade é percebida e implementada pode variar dependendo do domínio (BI *et al.*, 2022). Porém, estudos sobre a acessibilidade no desenvolvimento de aplicativos móveis ainda são escassos (LEITE *et al.*, 2021). Nesse contexto, Leite *et al.* (2021) conduziram uma pesquisa realizada com 872 pessoas envolvidas no desenvolvimento de aplicativos móveis na indústria brasileira. As principais evidências do estudo incluem (LEITE *et al.*, 2021):

- A maioria dos participantes possui uma consciência moderada sobre acessibilidade, mas demonstra baixos níveis de conhecimento e aplicação prática.
- A acessibilidade frequentemente não é considerada nos projetos devido à ausência de requisitos, tempo, treinamento e foco em usuários com deficiências.
- Normas e diretrizes de acessibilidade específicas tendem a ser adotadas mais por *designers*, profissionais de organizações maiores e por aqueles que desenvolvem aplicativos para a plataforma iOS.

Os autores sugerem ainda que sejam realizadas ações como a inclusão adequada da acessibilidade em cursos formais, pois o conhecimento e o treinamento adequado podem ser decisivos para influenciar os membros da equipe e os gerentes a incluir a acessibilidade no processo de desenvolvimento.

Por fim, no contexto de ASD, Miranda e Araujo (2022) investigaram como as empresas que utilizam métodos ágeis lidam com os requisitos de acessibilidade. O estudo focou em entender como esses requisitos são especificados, quais são os principais desafios enfrentados pelas equipes e como esses desafios são superados. Os autores identificaram desafios semelhantes aos encontrados em outros estudos, incluindo conhecimento superficial sobre acessibilidade, falta de treinamento e a ausência de demanda expressa pelos clientes como as principais razões para a falta de priorização desses requisitos nos projetos.

O estudo também indica que a maioria dos participantes concorda que modelos, listas e catálogos poderiam ajudar as equipes ágeis a especificar requisitos de acessibilidade. No entanto, alguns participantes discordam, argumentando que o uso do *backlog* do produto, *backlog* da *sprint* e protótipos são suficientes, e que “nenhum artefato auxiliará um processo que a empresa não está disposta a priorizar” (MIRANDA; ARAUJO, 2022, p. 1315).

Independente do contexto das pesquisas, elas convergem em suas conclusões. Desenvolvedores têm apresentado alguma consciência sobre a importância da acessibilidade, porém o conhecimento sobre o tema é superficial, faltam orientações práticas, requisitos de acessibilidade não são bem especificados, as responsabilidades não são bem definidas e a acessibilidade não é priorizada pelas partes interessadas. Entre as possíveis soluções, apontadas nos estudos para esses problemas, destacam-se: oferecer cursos e treinamentos para ampliar o conhecimento

dos desenvolvedores sobre como resolver os problemas de acessibilidade; utilizar suítes de testes apropriadas, que sejam simples e fáceis de serem usadas, por exemplo, incluindo testes automatizados; e atribuir responsabilidades mais objetivas e diretas a todos os envolvidos no desenvolvimento de produtos acessíveis. No Quadro 11 são indicadas as estratégias sugeridas por pelo menos dois dos estudos revisados.

Quadro 11 – Estratégias sugeridas a partir de estudos com desenvolvedores

Estratégia	**Referências**
Ampliar conscientização geral sobre acessibilidade	(INAL; RıZVANOğLU; YESILADA, 2019) (ANTONELLI *et al.*, 2018)
Prover políticas públicas e aumentar fiscalização	(INAL; RıZVANOğLU; YESILADA, 2019) (ANTONELLI *et al.*, 2018)
Oferecer cursos e treinamentos para desenvolvedores	(INAL; RıZVANOğLU; YESILADA, 2019) (ANTONELLI *et al.*, 2018) (MIRANDA; ARAUJO, 2022) (LEITE *et al.*, 2021)
Demonstrar benefícios comerciais para empresas	(INAL; RıZVANOğLU; YESILADA, 2019) (LEITE *et al.*, 2021)
Conscientizar clientes para priorização do tema	(BI *et al.*, 2022) (MIRANDA; ARAUJO, 2022)
Tornar requisitos de acessibilidade mais claros	(BI *et al.*, 2022) (LEITE *et al.*, 2021)
Utilizar suítes de testes apropriadas	(ANTONELLI *et al.*, 2018) (BI *et al.*, 2022) (MIRANDA; ARAUJO, 2022)
Atribuir responsabilidades de forma objetiva	(INAL; RıZVANOğLU; YESILADA, 2019) (ANTONELLI *et al.*, 2018) (BI *et al.*, 2022)

Fonte: Elaborada pelo autor.

Uma forma de ofertar estratégias práticas, do ponto-de-vista dos desenvolvedores, é reduzindo o esforço de desenvolvimento pelo uso de ferramentas já estabelecidas. Para conhecer quais outras ferramentas estão disponíveis para testar a acessibilidade de aplicativos Android, além das oferecidas pela plataforma e descritas na Subseção 2.5.2, foram realizadas revisões constantes da literatura durante todo o período dessa pesquisa, com o início em 2019. As principais ferramentas destacadas pelos autores serão descritas na Seção 3.3.

3.3 Ferramentas para testes de acessibilidade no Android

Apesar do crescente número de diretrizes, padrões e recomendações para acessibilidade em dispositivos móveis, ainda existem problemas de acessibilidade em muitos aplicativos móveis. Por isso, além das diretrizes, são necessárias ferramentas para automatizar o processo de integração da acessibilidade nos aplicativos (OLIVEIRA; FILGUEIRAS, 2018).

Muitos estudos têm sido dedicados a propor, validar ou melhorar ferramentas para testes de acessibilidade (ELER *et al.*, 2018; VONTELL, 2019; OLIVEIRA; FILGUEIRAS,

2019; SALEHNAMADI *et al.*, 2021; GEMOU *et al.*, 2016; PARK *et al.*, 2019; SILVA; ELER; FRASER, 2018; HAO *et al.*, 2014)

Em um estudo secundário, Oliveira e Filgueiras (2018) analisaram ferramentas de suporte aos desenvolvedores para a criação de aplicativos móveis nativos acessíveis a pessoas com deficiência visual. As 13 ferramentas encontradas foram caracterizadas quanto a etapa do processo de desenvolvimento e seu grau de maturidade tecnológica. Os autores destacam, em suas conclusões, o baixo número de estudos dedicados a aplicativos nativos e que há pouca evidência de que as ferramentas são eficazes em ambientes profissionais para desenvolvimento de software.

No estudo de Silva, Eler e Fraser (2018), os autores pesquisaram sobre as ferramentas de avaliação automática disponíveis para a acessibilidade em aplicativos móveis. Cada ferramenta é descrita brevemente e os autores identificam quais diretrizes de acessibilidade cada ferramenta suporta para sua avaliação automática. As ferramentas discutidas para Android incluem, além do Android Lint, Espresso, Roboletric e Scanner de Acessibilidade, já descritos na Subseção 2.5.2, as seguintes:

PUMA (HAO *et al.*, 2014): É um *framework* que permite a criação de verificações personalizadas para avaliar a acessibilidade em aplicativos, focando em aspectos como tamanho do alvo de toque, espaçamento do alvo de toque e contraste de cores.

forApp:[4] É um serviço online pago que verifica automaticamente a conformidade de aplicativos móveis com os padrões de acessibilidade. Os resultados, que incluem texto alternativo para elementos não textuais e identificação de foco, podem ser baixados em formato PDF.

IBM AbilityLab Mobile Accessibility Checker:[5] É uma ferramenta que identifica e sugere ações para resolver problemas de usabilidade e acessibilidade em aplicativos móveis híbridos e nativos para iOS e Android. Ela verifica a falta de descrição, problemas de baixo contraste de cores, tamanho dos elementos e falta de foco do teclado.

Além dessas, Silva, Eler e Fraser (2018) também citam a ferramenta MATE (*Mobile Accessibility Testing*) (ELER *et al.*, 2018). A MATE explora aplicativos automaticamente ao aplicar diferentes verificações para problemas de acessibilidade relacionados à deficiência visual e motora. MATE faz checagens de acessibilidade nos elementos da UI e reporta o número de falhas únicas de cada tipo de checagem implementado, assim como um relatório detalhado de todas falhas de acessibilidade. É executado no ambiente Android (emuladores ou dispositivos físicos) e usa a estrutura do UI Automator para interagir com o aplicativo. A ferramenta coleta informações sobre a atividade em execução por meio de informações de acessibilidade disponíveis em cada visão. Sem necessitar do código fonte, implementa uma estratégia aleatória para explorar o

[4] <https://www.forapp.org/>
[5] <https://www-03.ibm.com/able/mobile-accessibility-checker.html>

aplicativo sob teste, identificando todas as possíveis interações do usuário na tela atual e seleciona aleatoriamente uma delas. Durante a exploração, o MATE aplica verificações de acessibilidade em *widgets* encontrados. As propriedades de acessibilidade que são verificadas são:

- Texto pronunciável/falado.
- Área de tamanho de toque.
- Taxa de contraste.
- Limites clicáveis duplicados.
- Extensão clicável.

Em uma análise feita principalmente pela perspectiva da ferramenta MATE, Silva (2020) propõe um "guia de automação de testes de acessibilidade para deficiências visuais", que é o resultado de um estudo sobre a possibilidade de automação da avaliação das recomendações para deficiência visual do guia da BBC (BBC, 2022), apresentado na Subseção 2.4.2. O estudo examinou cinco das onze seções do guia da BBC, classificando cada recomendação como totalmente automatizável, parcialmente automatizável, não automatizável ou não aplicável.

Vontell (2019) desenvolveu o *framework* Bility, cujo objetivo principal era fornecer uma estrutura de teste de acessibilidade fácil de usar (com configuração mínima) e que cobrisse uma ampla gama de testes de acessibilidade, de modo a encorajar os desenvolvedores a avaliar seus aplicativos em relação à acessibilidade e contribuir para o desenvolvimento de aplicativos móveis mais acessíveis. Implementado em Java e Kotlin, esse *framework* navega automaticamente por um aplicativo, encontrando problemas dinâmicos e estáticos de acessibilidade . O *framework* introduz novas técnicas para analisar interfaces de usuário, como determinar mudanças no contexto apenas por meio de informações da UI, bem como navegar por um aplicativo automaticamente com pouca ou nenhuma entrada do desenvolvedor. Suas principais contribuições são (VONTELL, 2019):

- Um algoritmo para gerar e reduzir automaticamente o espaço de estado de uma UI no contexto de uma análise específica.
- Uma linguagem independente de plataforma usada para descrever interfaces de usuário.
- Um processo automatizado para iniciar, navegar e gravar um aplicativo Android em uma tentativa de explorar completamente o espaço de estado do aplicativo.
- Um processo para testar problemas dinâmicos de acessibilidade, como navegação pelo teclado e espontânea no contexto de aplicação através da construção de uma máquina de estado que representa o aplicativo.

Foram realizadas comparações entre a capacidade do Bility de detectar problemas com as ferramentas existentes, como o Scanner de Acessibilidade. Foi constatado que Bility apresentou um desempenho significativamente melhor e é capaz de detectar problemas estáticos

adicionais, bem como descobrir problemas dinâmicos de acessibilidade que as ferramentas atuais da plataforma não detectam (VONTELL, 2019).

A Latte (SALEHNAMADI *et al.*, 2021) é uma ferramenta que avalia a acessibilidade de aplicativos Android por meio da reutilização de testes escritos para verificar a correção funcional do aplicativo. Ela extrai os casos de uso dos testes e os executa usando serviços assistivos, simulando a interação de usuários com deficiência. A Latte não analisa o código-fonte do aplicativo, mas avalia sua funcionalidade e acessibilidade com base na forma como é usado, especialmente por meio de serviços assistivos.

Park *et al.* (2019) propõem uma ferramenta de avaliação de acessibilidade focada na verificação da existência (ou não) de texto alternativo para conteúdo não textual, que é um fator de acessibilidade importante para usuários com deficiência visual. A ferramenta utiliza a biblioteca Beautiful Soup[6] do Python para analisar os arquivos ***Ex**tensible **M**arkup **L**anguage* (XML) do aplicativo móvel. Ele descobre a falta de texto alternativo verificando o atributo contentDescription do elemento ImageView nos arquivos XML do Android.

O Accessibility Engine for Android (Axe Android)[7] é um aplicativo comercial desenvolvido pela Deque Systems Inc., que realiza varreduras de acessibilidade em outros aplicativos enquanto estão em execução. A ferramenta verifica a conformidade com as diretrizes WCAG 2.0 e WCAG 2.1, além de outras melhores práticas, totalizando 10 diretrizes diferentes. A ferramenta é especialmente útil para identificar problemas de acessibilidade para pessoas com deficiências visuais e físicas. O Axe utiliza análise dinâmica, fazendo uma varredura na aplicação durante a execução, capturando uma única tela de cada vez. O usuário precisa ativar a ferramenta para cada tela. Os resultados podem ser visualizados em tela ou exportados como um arquivo *Comma-separated values* (CSV).

Por fim, outra ferramenta encontrada na literatura é o AccessibiLint (OLIVEIRA; FILGUEIRAS, 2019), que estende o Android Lint para aumentar o escopo das regras de verificação estática de código. Os autores adicionaram 17 regras baseadas em recomendações de acessibilidade para pessoas com deficiência visual. Isso permite que o desenvolvedor realize uma revisão de acessibilidade enquanto desenvolve aplicativos Android nativos. A ferramenta é *open source*, o que permite que a comunidade de desenvolvedores contribua para aprimorar suas funcionalidades. Um resumo das regras implementadas pelo AccessibiLint são mostradas no Quadro 12.

Oliveira e Filgueiras (2019) também vincularam, quando aplicável, as regras do AccessibiLint (ver Quadro 12) às recomendações do guia da BBC. Como a ferramenta realiza verificações estáticas de código, isso se mostrou particularmente útil, uma vez que tais veri-

[6] <https://pypi.org/project/beautifulsoup4/>
[7] <https://www.deque.com/android-accessibility/>

Quadro 12 – Regras do AccessibiLint

Id.	Regra
R01	Relação de contrastes de componentes devem ter uma taxa de 4.5
R02	Área de toque para componentes de interação
R03	Evitar ações de click no plano de fundo
R04	Componente de barra de ferramenta dentro de um layout com rolagem
R05	Componentes devem possuir textos únicos
R06	Evitar usar palavras separadas
R07	Evitar usar *sliders*
R08	A aplicação deve fornecer recursos que reduzam esforço do usuário
R09	Providenciar o título da página
R10	Teclado personalizado para o contexto da aplicação
R11	Ações importantes e de fácil acesso nas extremidades
R12	*Links* indevidamente rotulados
R13	Componentes de interação devem possuir espaçamentos relacionados a bordas ou outros componentes
R14	Rótulos alternativos em componentes para leitores de tela
R15	Possuir poucos itens de interação na tela
R16	Não ter textos muito longo
R17	Elementos decorativos não devem receber foco

Fonte: Adaptada de Oliveira e Filgueiras (2019).

ficações são menos complexas de serem abordadas por meio de testes pequenos, conforme apresentado na Seção 5.2.

Discussão sobre ferramentas existentes

Embora ferramentas baseadas em verificação estática de código, como o AccessibiLint (OLIVEIRA; FILGUEIRAS, 2019), sejam de baixo custo e devam ser integradas ao ambiente de desenvolvimento sempre que possível, elas geralmente funcionam apenas em nível sugestivo para os desenvolvedores e podem ser facilmente ignoradas ou não percebidas.

A proposta de Park *et al.* (2019) também realiza verificações estáticas, que podem ser integradas a rotinas de testes automatizados locais. No entanto, atualmente é feita apenas uma verificação de acessibilidade (texto alternativo para conteúdo não-textual), e o teste é realizado por uma biblioteca escrita em python que é executada a parte da IDE de desenvolvimento.

A maioria das ferramentas disponíveis, tais como PUMA, forApp, IBM AbilityLab Mobile Accessibility Checker e Axe, concentram-se em avaliações de produtos prontos, e a maioria delas abrange essencialmente o mesmo conjunto de diretrizes, o que pode indicar que essas diretrizes são as mais simples e evidentes de serem abordadas, ou que são as recomendações que podem ser avaliadas automaticamente (SILVA; ELER; FRASER, 2018).

Em relação às ferramentas de Eler *et al.* (2018), Vontell (2019) e (SALEHNAMADI *et al.*,

2021), ainda que utilizem técnicas que permitam uma maior cobertura de testes, elas priorizaram tais testes após a codificação e verificam subconjuntos dos problemas de acessibilidade definidos pelos autores. Isso indica que os testes de acessibilidade apresentam limitações em relação à abrangência e são realizados principalmente após o desenvolvimento do software, sendo essencialmente testes de UI, que são testes com *feedback* mais distante do desenvolvedor.

É importante destacar a escassez de estudos voltados para a inclusão de testes de acessibilidade durante a fase de Construção de Software, especialmente desde o início da codificação. As alternativas que se integram ao ambiente de desenvolvimento, sem necessidade de execução em dispositivos Android, estão geralmente limitadas a avisos. São escassas as alternativas para automatização de testes de acessibilidade com testes pequenos.

3.4 Considerações finais

Neste capítulo, foi apresentada uma revisão da literatura sobre acessibilidade em aplicativos móveis, com foco nos principais problemas de acessibilidade enfrentados por usuários com deficiência visual. Observou-se que muitos desses problemas podem ser resolvidos de forma simples, como com ajustes no contraste de cores de textos.

Os resultados do mapeamento sistemático realizado destacam a importância de estabelecer estratégias para o desenvolvimento de aplicativos móveis acessíveis, com mais participação dos desenvolvedores.

Os estudos com desenvolvedores evidenciaram a necessidade de maior conhecimento sobre acessibilidade, orientações claras, especificações de requisitos de acessibilidade e definição de responsabilidades. Recomenda-se oferecer cursos e treinamentos, utilizar suítes de testes apropriadas e automatizadas, além de atribuir responsabilidades diretas a todos os envolvidos no desenvolvimento de produtos acessíveis.

Na análise das ferramentas, identificadas na literatura, para testes de acessibilidade em aplicativos Android, destaca-se a escassez de estudos sobre a inclusão de testes de acessibilidade durante a fase de Construção de Software, com poucas opções para automatização de testes locais de acessibilidade.

Com base nessas evidências da literatura, no próximo capítulo é apresentada uma estratégia de testes de acessibilidade integrados ao processo de desenvolvimento de aplicativos móveis. Essa proposta busca capacitar os desenvolvedores a criar aplicativos acessíveis desde o início da codificação, facilitando a inclusão de requisitos de acessibilidade e a realização de testes automatizados de forma mais eficiente e eficaz.

CAPÍTULO

4

ANTECIPAÇÃO DE TESTES: UMA ESTRATÉGIA PARA ENVOLVER OS DESENVOLVEDORES

A despeito de todos os esforços encontrados na literatura, principalmente estabelecendo conjuntos de diretrizes e recomendações, a acessibilidade em aplicativos para dispositivos móveis ainda não é satisfatória (YAN; RAMACHANDRAN, 2019). Ferramentas disponíveis identificadas na literatura, como as propostas por Eler *et al.* (2018) e por Vontell (2019), propõem o uso de testes automatizados para verificação de acessibilidade. Porém os testes são executados após a codificação, e testam subconjuntos de problemas de acessibilidade definidos pelos autores.

Visando aumentar o envolvimento dos desenvolvedores na produção de aplicativos móveis acessíveis, este trabalho buscou identificar uma abordagem de desenvolvimento de aplicativos móveis acessíveis, que possa ser incorporada às atividades rotineiras dos desenvolvedores, reduzindo as barreiras para sua adoção.

Para elaborar essa proposta de abordagem, foram exploradas e selecionadas estratégias, com base no material bibliográfico e artefatos gerados durante a revisão da bibliografia. Com foco nos principais desafios de acessibilidade enfrentados por usuários com deficiência visual, as estratégias levaram em consideração os seguintes aspectos:

- Os problemas mais recorrentes são contemplados pelas técnicas existentes.
- São escassas as opções para testes locais de acessibilidade.
- Alguns estudos sugerem integração de requisitos de acessibilidade com práticas ágeis.
- Estratégias sugeridas na literatura incluem requisitos bem definidos, suítes de testes apropriadas e atribuições de responsabilidades objetivas.

Neste capítulo, descreve-se o percurso realizado nesta exploração, o qual abrange as etapas segunda e terceira descritas na Seção 1.3. De modo específico, na Seção 4.1 é feito um

recorte da literatura, delimitando o escopo das verificações de acessibilidade a serem consideradas neste trabalho. Na Seção 4.2 é discutida uma combinação de ferramentas e práticas para distribuição dos testes. Na Seção 4.3 são apresentadas algumas premissas, como a atribuição de responsabilidade ao desenvolvedor para incluir acessibilidade em *apps mobile*, e uma primeira proposta elaborada para o desenvolvimento de aplicativos móveis acessíveis. Na Seção 4.4 são relatadas impressões obtidas com profissionais da indústria por meio de entrevistas semiestruturadas, e, por fim, uma reflexão sobre os *feedbacks* coletados com desenvolvedores, e as consequentes estratégias que orientam e fundamentam a principal contribuição deste estudo, como será apresentado no capítulo subsequente.

4.1 Definição de escopo das verificações de acessibilidade

Testar todos os problemas de acessibilidade, em todos os componentes e estruturas de interface pode alocar muito tempo dos desenvolvedores, que tendem a não escrever testes de acessibilidade dedicados. Assim, as soluções existentes podem atender apenas a um subconjunto restrito de problemas de acessibilidade (ELER *et al.*, 2018; VONTELL, 2019).

Tentando entender como mitigar esse problema, foram considerados os estudos da literatura apresentados na Subseção 3.1.1 para analisar e sugerir um escopo de cobertura de testes de acessibilidade. Com isso, foram enfatizados os principais problemas de acessibilidade enfrentados por usuários com deficiência visual, tendo por base os problemas compilados por Dias, Duarte e Fortes (2021), e as respectivas recomendações para evitar tais problemas.

Considerou-se também, os elementos de UI em que são encontrados a maioria das violações de acessibilidade nos aplicativos mais populares de acordo com Yan e Ramachandran (2019), Alshayban, Ahmed e Malek (2020) e Oliveira *et al.* (2023), seguindo a relação apresentada no Quadro 7.

Parte da dificuldade na criação de testes automatizados está em torná-los genéricos e abrangentes o suficiente. Porém, ao priorizar quais testes e em quais elementos realizá-los, são estabelecidos critérios mais objetivos, com potencial de solucionar a maior parte dos problemas encontrados em recorrência.

Definido o escopo inicial a ser atendido, o passo seguinte foi a escolha das ferramentas e estratégias de testes.

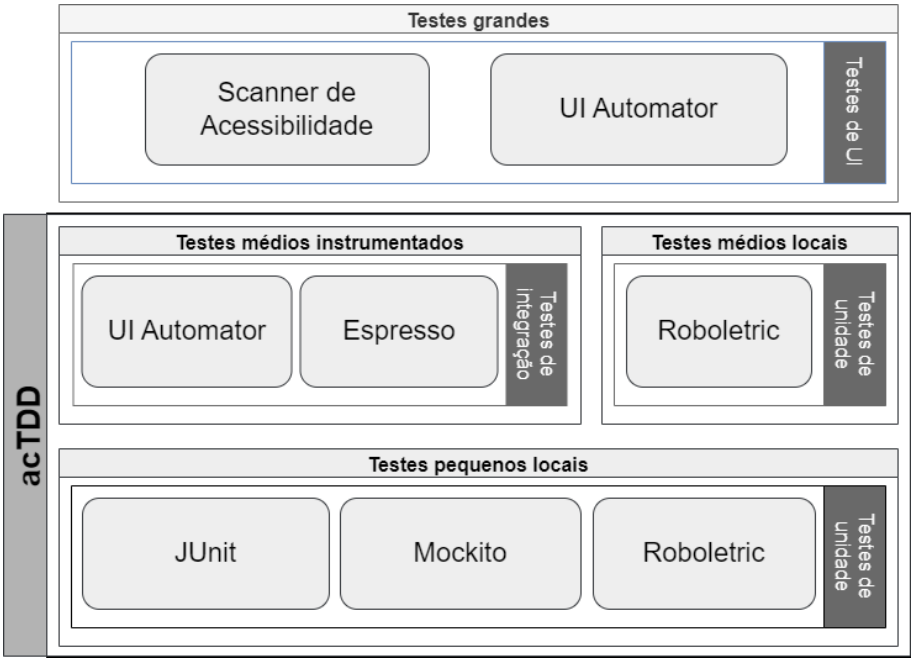
4.2 Distribuição dos testes e seleção de ferramentas

Muitas são as ferramentas disponíveis para desenvolvedores Android implementarem testes. Além das citadas na Subseção 2.5.2 e na Seção 3.3, existem bibliotecas e ferramentas de testes

automatizados como JUnit[1], Mockito[2], MonkeyRunner[3], Robotium[4], entre outras (KOCHHAR *et al.*, 2015). A utilização das ferramentas e os tipos de teste a serem realizados por cada uma, são definidos de acordo com a estratégia adotada. A documentação de testes do site *Android Developers* (GOOGLE, 2023c) recomenda que a proporção siga o modelo de pirâmide da Figura 2, com 70% de testes pequenos, 20% de testes médios e 10% de testes grandes.

Uma possível composição do conjunto de testes é sugerida pelos autores deste trabalho e ilustrada na Figura 11. A composição pode incluir testes pequenos com *frameworks* para testes de unidade como JUnit, Mockito e Roboletric, todos eles podendo ser executados de forma não instrumentada. Para testes médios, é possível a utilização do Espresso, inclusive por meio de testes de unidade instrumentados, ou ainda pelo UI Automator, também de modo instrumentado. Já as opções para os testes grandes são mais numerosas, como o próprio UI Automator, o Scanner de Acessibilidade, ou ferramentas como MATE (ELER *et al.*, 2018), Bility (VONTELL, 2019), Latte (SALEHNAMADI *et al.*, 2021), entre outras.

Figura 11 – Exemplo de composição de conjunto de ferramentas para estratégia de testes de acessibilidade.



Fonte: Elaborada pelo autor.

São encorajadas estratégias que combinem testes de UI com testes instrumentados ou mesmo locais (GOOGLE, 2023b). No entanto, conforme discutido na Seção 3.3, atualmente os testes de acessibilidade são geralmente realizados após a codificação. Os autores acreditam

[1] <https://junit.org/junit4/>
[2] <https://site.mockito.org/>
[3] <https://developer.android.com/studio/test/monkeyrunner?hl=pt-br>
[4] <https://github.com/RobotiumTech/robotium>

que, estabelecendo responsabilidades mais claras, todos os envolvidos podem trabalhar de forma integrada para promover a acessibilidade.

4.3 Mais responsabilidade aos desenvolvedores

Durante o mapeamento sistemático realizado neste trabalho, apresentado na Seção 3.2, foi observado que há poucos estudos que abordam a inclusão de acessibilidade durante a fase de Construção de Software. Porém, alguns estudos que tratam de usabilidade oferecem alternativas que também podem ser desenvolvidas para evitar problemas de acessibilidade.

A abordagem proposta por Wolkerstorfer *et al.* (2008), por exemplo, combina práticas de XP com métodos UCD para um "processo de usabilidade ágil". Um dos instrumentos propostos pelos autores foi o teste de unidade estendido (*Extended unit test*), no qual são adicionados casos de teste específicos de usabilidade. Esses casos de teste incluem semântica (o rótulo correto de um botão, por exemplo) aos testes baseados em código, permitindo assim a verificação do cumprimento de diretrizes (como o uso de letras maiúsculas em botões, no mesmo exemplo). A abordagem é baseada no TDD, ou seja, em escrever os testes antes de desenvolver a funcionalidade. Assim, ao adicionar testes de unidade com semântica relacionados à usabilidade, os desenvolvedores conseguem definir a usabilidade da aplicação.

Seguindo o mesmo princípio, a primeira sugestão desta pesquisa para incentivar desenvolvedores a realizarem testes de acessibilidade em seus projetos, consiste em uma proposta orientada a testes para o desenvolvimento de requisitos de acessibilidade, chamada **acTDD**. A acTDD tem o objetivo de auxiliar os desenvolvedores a criarem aplicativos móveis acessíveis e é fundamentada em três pilares:

1. A transformação do conjunto de recomendações de acessibilidade (maRs) em requisitos de software.
2. A priorização dos elementos de interface que possuem mais violações de acessibilidade, de acordo com a literatura.
3. A distribuição dos testes para que haja mais testes locais do que testes instrumentados, reduzindo o tempo de feedback.

Com essa proposta, detalhada na subseção seguinte, os desenvolvedores assumem a responsabilidade de implementar requisitos de acessibilidade como parte dos requisitos de software.

4.3.1 Uma proposta para inserir requisitos de acessibilidade com TDD

Durante o desenvolvimento dirigido por testes, o desenvolvedor é estimulado a criar um teste para cada novo requisito de software. Em uma proposta de acessibilidade dirigida por testes,

cada recomendação de acessibilidade a ser considerada se torna um novo requisito e requer um novo teste. Seguindo essa sugestão de fluxo de desenvolvimento, o desenvolvedor pode buscar resolver os problemas de acessibilidade que desejar, nos elementos de UI que escolher, pois terá como ponto de partida um novo requisito.

Embora essa proposta não limite a quantidade de problemas ou elementos a serem abordados, desenvolver testes de acessibilidade para uma ampla gama de problemas e elementos pode não ser escalável (ELER *et al.*, 2018). No entanto, com base nos resultados extraídos da literatura (YAN; RAMACHANDRAN, 2019; ALSHAYBAN; AHMED; MALEK, 2020; OLIVEIRA *et al.*, 2023), é possível assumir que se o desenvolvedor priorizar os problemas e elementos elencados no Quadro 7, testes bem sucedidos podem evitar boa parte dos problemas mais recorrentes em aplicativos Android.

No exemplo a seguir, será demonstrado o fluxo de implementação de um requisito de acessibilidade utilizando a acTDD.

Exemplo: baixo contraste da cor do texto

O primeiro passo, no TDD, é sempre a criação de um teste para um novo requisito de software. Neste exemplo, o que se deseja é evitar o problema de acessibilidade de contraste de texto e/ou imagem insuficiente, apresentado no Quadro 5 como P1. Para esse problema, a literatura apresenta como requisito a recomendação de acessibilidade **maR07**.

"maR07 - Contraste de cor: deve-se utilizar uma taxa de contraste de cores pelo menos **4.5:1**, pois ajuda os usuários a identificar melhor o conteúdo do *app*" (DIAS, 2021).

Para inferir a taxa de contraste de cores entre um texto e seu plano de fundo, é necessário acessar as cores utilizadas em ambos. Neste exemplo, a taxa de contraste foi calculada utilizando a biblioteca ColorUtils do Android, que implementa o cálculo de contraste conforme a fórmula proposta pela W3C.[5]

Uma maneira de realizar essa inferência no Android é por meio de um teste instrumentado utilizando o Espresso.[6] Para verificar a taxa de contraste de um TextView em tela, o teste deve_UtilizarTaxaDeContrasteAdequada_Espresso acessa a cor do texto e do *background*, e obtém a taxa de contraste. Se o resultado for inferior a 4.5, o teste falha, e será necessário realizar alterações nas cores do elemento até que o teste passe.

[5] WCAG 2.0 *contrast ratio* - <https://www.w3.org/TR/2008/REC-WCAG20-20081211/#contrast-ratiodef>

[6] O exemplo apresentado foi implementado apenas para ilustrar a aplicação da técnica. No entanto, é possível verificar a taxa de contraste da cor do texto, com o Espresso, habilitando a API AccessibilityChecks, como demonstrado na Subseção 2.5.2. As considerações em relação ao tempo de *feedback*, no entanto, são as mesmas.

Código-fonte 2 – Teste da taxa de contraste de um *TextView* com o Espresso

```
1:  @Test
2:  public void deve_UtilizarTaxaDeContrasteAdequada_Espresso(){
3:      onView(withId(R.id.text_label))
4:              .check(matches(utilizaTaxaDeContrasteAdequada(MIN_CONSTRAST_RATIO)));
5:  }
6:
7:  Matcher<View> utilizaTaxaDeContrasteAdequada(final double ratio) {
8:
9:      return new TypeSafeMatcher<View>() {
10:
11:         @Override
12:         public boolean matchesSafely(View view) {
13:             // código de validação aqui
14:
15:             ColorDrawable cd = (ColorDrawable) view.getBackground();
16:
17:             int backgroundColor = cd.getColor();
18:             int textColor;
19:             if (view instanceof TextView) {
20:                 textColor = ((TextView) view).getCurrentTextColor();
21:             }
22:
23:             double r = ColorUtils.calculateContrast(backgroundColor, textColor);
24:
25:             return r >= ratio;
26:         }
27:     };
28: }
```

Embora eficaz, a execução do teste pelo Espresso requer tempo de execução, pois envolve a inicialização do emulador e a reinstalação da aplicação no dispositivo para executar o teste.

Uma alternativa mais adequada, de acordo com a distribuição da pirâmide de testes (ver Figura 2), seria implementar a mesma verificação como um teste local. No Código-fonte 3, é demonstrado um exemplo de implementação do mesmo teste utilizando o Roboletric, que não requer a execução por meio de um dispositivo físico ou emulado.

Código-fonte 3 – Teste da taxa de contraste de um *TextView* com o Roboletric

```
1:  @Test
2:  public void deve_UtilizarTaxaDeContrasteAdequada_Roboletric() {
3:      MainActivity activity = Robolectric.setupActivity(MainActivity.class);
4:      View view = activity.findViewById(R.id.text_label);
5:
6:      if (view != null) {
7:          if (view.getBackground() instanceof ColorDrawable) {
8:              ColorDrawable cd = (ColorDrawable) view.getBackground();
9:
10:             int backgroundColor = cd.getColor();
11:
12:             int textColor;
13:             if (view instanceof TextView) {
14:                 textColor = ((TextView) view).getCurrentTextColor();
15:             }
16:
17:             double ratio = ColorUtils.calculateContrast(backgroundColor, textColor);
18:
19:             assertThat(ratio, greaterThan(MIN_CONSTRAST_RATIO));
20:         }
```

```
21:             }
22:         }
```

O exemplo apresentado não é um modelo definitivo, mas sim uma demonstração simplificada do fluxo de implementação com o acTDD. É importante ressaltar que o teste pode ser refatorado e aprimorado em um ciclo típico de TDD. Além disso, os desenvolvedores podem considerar a melhoria do teste, aplicando-o a todos os elementos da interface do usuário de uma tela que pertença ao subconjunto especificado.

Ao finalizar um ciclo de implementação de requisito com o acTDD, o desenvolvedor alcança dois principais benefícios: a) o requisito de acessibilidade atendido; e b) um teste automatizado é criado para verificar eventuais alterações nas propriedades da interface, garantindo que a acessibilidade seja preservada.

É importante lembrar que o TDD é uma técnica de desenvolvimento – e não de testes. Assim, a proposta do acTDD exige que os testes sejam escritos durante o desenvolvimento – e não em etapas posteriores. Isso atribui mais responsabilidade aos desenvolvedores e requer uma mudança de paradigmas.

A fim de obter uma perspectiva pragmática, não apenas do acTDD, mas também de todas as estratégias abordadas neste capítulo, como a delimitação do escopo, a seleção de ferramentas e a distribuição dos testes, foram realizadas entrevistas com profissionais da indústria, conforme apresentado na Seção 4.4.

4.4 Entrevistas com profissionais da indústria

Como parte do processo de construção e consolidação das estratégias elaboradas e elencadas anteriormente, foram conduzidas entrevistas com profissionais de empresas de desenvolvimento de software com experiência em aplicativos móveis.

Os objetivos deste estudo foram:

- Compreender como a acessibilidade é abordada nos projetos em que os profissionais atuam.
- Compreender como são atribuídas responsabilidades para verificações de acessibilidade em projetos de desenvolvimento *mobile*.
- Obter a percepção dos desenvolvedores sobre a viabilidade e adequação de uma proposta dirigida por testes para inserção de acessibilidade.

Planejamento

Neste estudo, optou-se pela realização de entrevistas semiestruturadas, conduzidas com o uso de um guia com questões e tópicos a serem discutidos, mas sem a obrigatoriedade de seguir uma ordem rigorosa. Essa estratégia permite ao entrevistador adaptar o roteiro de acordo com a condução da entrevista, tornando-a mais flexível e oportunística, permitindo assim aprofundar o entendimento e explorar diferentes aspectos (GIL, 2008; LAZAR; FENG; HOCHHEISER, 2017).

De modo preliminar, foi elaborado um roteiro para as entrevistas contendo os principais tópicos a serem abordados em formato de perguntas. O roteiro utilizado neste estudo pode ser visto no Apêndice E.

Os participantes foram recrutados por meio de *e-mails* e mensagens em redes sociais (LinkedIn e Slack), bem como por contato direto com empresas de Tecnologia da Informação (TI) de São Carlos, no interior de São Paulo, e região. Foram convidados profissionais de diferentes perfis. A amostra foi por conveniência (YIN, 2016). O Apêndice D ilustra a mensagem convite, contendo as etapas envolvidas.

Condução

Participaram das entrevistas cinco profissionais, oriundos de três diferentes empresas da área de TI. No Quadro 13 estão relacionadas a equipe de trabalho desses profissionais e as principais experiências de cada um deles – no que se refere ao escopo desta pesquisa –, associadas a uma identificação que será usada nas discussões subsequentes.

Quadro 13 – Perfil dos profissionais entrevistados

Identificação	Equipe de trabalho	Principais experiências
E1	Garantia da Qualidade	Automação de testes
E2	Desenvolvedor	Desenvolvimento *mobile*
E3	Garantia da Qualidade	Automação de testes para web
E4	Garantia da Qualidade	XP e Automação de testes para *mobile*
E5	*Tech Leader*	Projetos Android e testes no Android

Fonte: Elaborada pelo autor.

As entrevistas foram realizadas entre fevereiro e abril de 2023 via Google Meet. Ao todo, foram realizados três encontros, sendo que três profissionais de uma mesma empresa participaram concomitantemente em um desses encontros.

Em cada encontro foi feita, inicialmente, uma breve apresentação do contexto da pesquisa e das estratégias em elaboração – delimitação de escopo para testes, ferramentas, distribuição de testes e acTDD. Depois, cada tópico do roteiro era coberto, oportunamente, de acordo com a condução da conversa informal. Este formato possibilitou que os participantes apresentassem suas percepções, experiências e sugestões de forma livre. Uma síntese dessas contribuições é apresentada na subseção seguinte.

Feedbacks dos entrevistados

Os *feedbacks* obtidos a partir das entrevistas semiestruturadas agregaram importantes reflexões sobre essa pesquisa. Além das opiniões dos entrevistados, foram consideradas para essa análise, os *feedbacks* recebidos na revisão de um artigo científico submetido em outubro de 2022 para o *Symposium On Applied Computing* - SAC.[7]

De acordo com um dos revisores da publicação submetida, embora as diretrizes sejam pré-definidas, não é necessário que os testadores construam os testes individualmente de antemão, e o TDD pode não ser a abordagem adequada para resolver o problema apresentado. A mesma opinião foi defendida pelo entrevistado E5, que lembra que TDD tem sido mais útil explorando conteúdo não conhecido e que *apps mobile* normalmente possuem pouca regra de negócio.

A proposta deste trabalho, que busca incentivar o uso de testes de acessibilidade por parte dos desenvolvedores de forma prática, pressupõe maior participação desses atores. No entanto, como discutido na Subseção 3.2.2 e na Seção 3.3, essa não é a maneira como a acessibilidade tem sido considerada atualmente. Durante as entrevistas, quando perguntados sobre como são feitas as verificações de acessibilidade nos projetos em que atuam, os entrevistados indicaram que elas costumam acontecer essencialmente nas etapas de Testes ou Manutenção, ou seja, nas etapas finais do ciclo de desenvolvimento, e são realizados por testadores, não necessariamente os programadores. O entrevistado E1, por exemplo, afirma que os testadores automatizam testes de integração e de UI, e os testes automatizados são usados para acelerar os testes de regressão.

O entrevistado E5 acredita que um conjunto de testes de acessibilidade entregaria mais valor com testes instrumentados. Ele entende que a pirâmide de testes não é fidedigna ao contexto *frontend*, sendo preferível uma distribuição em formato hexagonal, ou seja, com mais testes de integração, em comparação com testes de unidade e testes de UI. Opinião parecida foi apontada por um dos revisores da publicação submetida, que argumenta ser preferível que testes de acessibilidade sejam realizados por testes de UI.

Para provocar uma mudança na forma como os testes são pensados e executados pela indústria de software atualmente, é necessário discutir responsabilidades. Conforme destacado

[7] <https://www.sigapp.org/sac/sac2023/>

na Subseção 3.2.2, a indefinição das responsabilidades tem sido apontada como uma das razões pelas quais a acessibilidade não tem sido incorporada nos processos de desenvolvimento. Especialmente no ASD, é importante promover a colaboração entre equipes para garantir produtos de alta qualidade (CRISPIN; GREGORY, 2009; MIRANDA; ARAUJO, 2022). Assim, a responsabilidade deve ser dividida entre toda a equipe, incluindo *designers*, desenvolvedores e testadores (MIRANDA; ARAUJO, 2022). Esse argumento corrobora com a definição de UX cunhada por Donald Norman, que considera que a boa experiência do usuário alvo deve ser pensada por todos do time de desenvolvimento, cada um dando a sua contribuição de acordo com o seu papel no processo.[8]

O que se observou com as entrevista, no entanto, é que os diferentes perfis acham que a responsabilidade deve ser de outros atores, e não necessariamente do seu perfil, assim como observado em outros estudos com desenvolvedores (BI *et al.*, 2022; LEITE *et al.*, 2021; MIRANDA; ARAUJO, 2022). O entrevistado E1 entende que faz sentido antecipar a implementação de requisitos de acessibilidade, pois quanto mais cedo resolver o requisito, menor o custo de desenvolvimento. Esse entendimento pressupõe que a responsabilidade seja do *designer*. E2, o entrevistado com o perfil de desenvolvedor e, portanto, presente na etapa de desenvolvimento, entende que uma abordagem TDD entrega muita responsabilidade ao desenvolvedor, e que, mesmo com um conjunto definido de problemas e elementos a serem verificados, esses atores podem ter dificuldade para implementar soluções generalistas. Ao mesmo tempo, E2 entende que o desenvolvimento de uma ferramenta que entregue soluções prontas para este fim, ainda que possível, demandaria um trabalho grande para ser desenvolvida. Para E2, essa atribuição – garantir a acessibilidade dos *apps* – deve ser dos *designers* de UI/UX. E5 corrobora com essa opinião e afirma que um bom *design system* já contempla requisitos de acessibilidade.

Quanto ao suporte recebido para lidar com questões de acessibilidade, E1 afirma que não é dada prioridade à alocação de tempo para implementar testes de acessibilidade. Já E5 relata que em todas as empresas e projetos em que trabalhou, a acessibilidade tem sido tratada apenas como uma estratégia comercial.

4.5 Discussão

Os argumentos contrários à proposta baseada em TDD são relevantes. Partem de um padrão estabelecido na literatura: testes de acessibilidade são essencialmente testes de UI, pois testam requisitos não funcionais (OLIVEIRA; ELER, 2017; MIRANDA; ARAUJO, 2022; BI *et al.*, 2022). Porém, esta prática não tem sido suficiente para evitar os problemas mais triviais e recorrentes de acessibilidade em aplicativos móveis (YAN; RAMACHANDRAN, 2019; ALSHAYBAN;

[8] <https://www.nngroup.com/articles/definition-user-experience/>

AHMED; MALEK, 2020; OLIVEIRA *et al.*, 2023).

A distinção prevalente é de que requisitos funcionais descrevem características de funcionalidade do software, enquanto requisitos não funcionais descrevem características de qualidade, tais como usabilidade, confiabilidade, eficiência, manutenibilidade e portabilidade (TURINE; MASIERO, 1996). De acordo com Eckhardt, Vogelsang e Fernández (2016), os requisitos não funcionais geralmente são documentados separadamente dos requisitos funcionais, com descrições consideradas superficiais, não precisas e sem medidas quantitativas, o que torna difícil analisá-los e testá-los. Os autores sugerem que muitos requisitos "não funcionais" não são realmente não funcionais, pois descrevem o comportamento de um sistema. Eles destacam a importância de tratar esses requisitos como funcionais, para que possam ser analisados e testados de maneira semelhante. No contexto de acessibilidade, Bi *et al.* (2022) defendem que "a acessibilidade pode ser tratada como um requisito funcional para garantir que as pessoas usem projetos de software com mais eficiência".

Existem desafios técnicos importantes na criação de testes de acessibilidade pelo desenvolvedor. Nem todo requisito de acessibilidade pode ser verificado sem o suporte de um dispositivo físico ou emulado (OLIVEIRA; FILGUEIRAS, 2019; SILVA, 2020), além de ainda serem insubstituíveis os testes manuais e os testes de usuários. Entretanto, se for possível criar testes automatizados ao menos para os problemas de acessibilidade mais recorrentes, e nos elementos em que esses mais incidem, bem como integrar esses testes em ciclos de CI/CD, a indústria produzirá e entregará aplicativos móveis mais acessíveis.

Nesse sentido, ainda que a abordagem TDD não seja a solução ideal para resolver o problema, é recomendado incentivar o uso de testes locais de acessibilidade, integrados ao ambiente de desenvolvimento, e promover a participação dos desenvolvedores na inserção da acessibilidade em aplicativos móveis.

O uso dessa estratégia não significa, no entanto, que as responsabilidades dos outros papéis devam ser diminuídas. Ao contrário, o envolvimento de todos e a auto-organização das equipes fazem parte dos princípios do desenvolvimento ágil (BECK *et al.*, 2001; PRESSMAN, 2011; MIRANDA; ARAUJO, 2022).

As subseção a seguir destaca algumas redefinições das estratégias abordadas neste estudo, considerando o que foi observado com a revisão de literatura, o estudo das ferramentas disponíveis, a entrevista com profissionais e as discussões aqui apresentadas.

4.5.1 Redefinição das estratégias

A proposta de incluir requisitos de acessibilidade em um ciclo semelhante ao do TDD é relevante, embora necessite de mais investigação. As premissas que sustentaram essa proposta continuam

válidas, incluindo a distribuição e delimitação de responsabilidades, maior envolvimento dos desenvolvedores, uso de testes automatizados locais e aproveitamento de soluções já existentes.

A partir dessas premissas, foram consideradas cinco estratégias diferentes, apresentadas no Quadro 14, como contribuições para o desenvolvimento de aplicativos móveis acessíveis.

Quadro 14 – Possíveis estratégias para aprofundamento de pesquisa

Estratégias	
Orientação de testes	Indicação dos principais problemas de acessibilidade a serem verificados e sugestões de como abordar esses problemas, por papéis, com técnicas e ferramentas existentes.
Testes locais	Criar e incentivar mais testes locais, por meio de exemplos, atribuindo mais responsabilidade aos desenvolvedores.
Kit de testes	Implementar testes automatizados possíveis para cada papel, e disponibilizar suíte de testes *opensource*.
Pirâmide de testes	Distribuir soluções existentes na estrutura da pirâmide de testes.
Extension Libs	Trabalhar com extensões dos *widgets* que ou já tratem possíveis problemas de acessibilidade, ou facilitem a realização de testes automatizados.

Fonte: Elaborada pelo autor.

Com base no conhecimento técnico e científico adquirido nesta pesquisa, sugere-se que não seja solucionada uma das estratégias elencadas, mas sim, que seja oferecida uma proposta abrangente. Essa proposta consiste em um *framework* conceitual que disponibiliza aos pesquisadores e desenvolvedores um conjunto de conceitos, orientações, técnicas e ferramentas sobre acessibilidade. O *framework*, descrito no Capítulo 5, considera as estratégias apresentadas no Quadro 14, fornecendo soluções iniciais e orientações para futuras explorações. O objetivo é apresentar o conteúdo de maneira simples e prática, para incentivar o seu uso.

4.6 Considerações finais

Neste capítulo foram relatadas as experiências de pesquisa explorando soluções práticas para o desenvolvimento de aplicativos móveis acessíveis, com especial enfoque na maior atribuição de responsabilidade aos desenvolvedores. Para isso, foram estabelecidas estratégias como a priorização de problemas e elementos a serem verificados e a preferência por testes automatizados locais. Uma proposta de inclusão de requisitos de acessibilidades utilizando TDD foi apresentada. No entanto, com o objetivo de apresentar uma contribuição mais abrangente, agrupando os diversos aspectos aprendidos durante o desenvolvimento desta pesquisa, optou-se pela elaboração de um *framework* de soluções práticas para auxiliar na produção de aplicativos acessíveis Android nativos.

No Capítulo 5 será descrita a composição deste *framework*, incluindo a descrição dos primeiros artefatos produzidos.

CAPÍTULO

5

AALT: *FRAMEWORK* DE APOIO À PRODUÇÃO DE *APPS* ACESSÍVEIS PARA ANDROID

Com o objetivo de reunir uma coleção de soluções práticas para apoiar o desenvolvimento de aplicativos acessíveis nativos para Android, foi elaborado o ***Framework* de Aprendizado e Testes de Acessibilidade para Android** (**AALT Framework** - ***Android Accessibility Learning and Testing Framework***).

A decisão de apresentar, como principal contribuição desta pesquisa de mestrado, um *framework* conceitual foi fundamentada na premissa de que todas as experiências adquiridas, as lições aprendidas e os artefatos gerados podem contribuir para avanços no tema da acessibilidade em aplicativos móveis, especialmente em aplicativos nativos para Android. Além disso, um *framework* pode servir como orientação prática na indústria, ao mesmo tempo em que permite adaptações e uma abordagem incremental (NERIS, 2010).

Na próxima seção, é descrita a estrutura do AALT e as premissas que a orientaram. Em seguida, são apresentados os artefatos que compõem o *framework* e foram produzidos durante este trabalho. Por fim, são fornecidas informações e orientações para a aplicação do AALT.

5.1 Estrutura do AALT Framework

A estrutura do *framework* AALT é organizada de tal maneira que os artefatos que o integram são distribuídos em dois eixos. São eles:

1. **RESPONSABILIDADES (eixo vertical)**. Foram considerados neste eixo papéis relevantes em projetos de ASD: ***designers***, **desenvolvedores**, **testadores**, **gestores e clientes**, sendo que os dois últimos foram agrupados nesta estrutura.
2. **CATEGORIA (eixo horizontal)**. Cada artefato foi categorizado conforme seu conteúdo

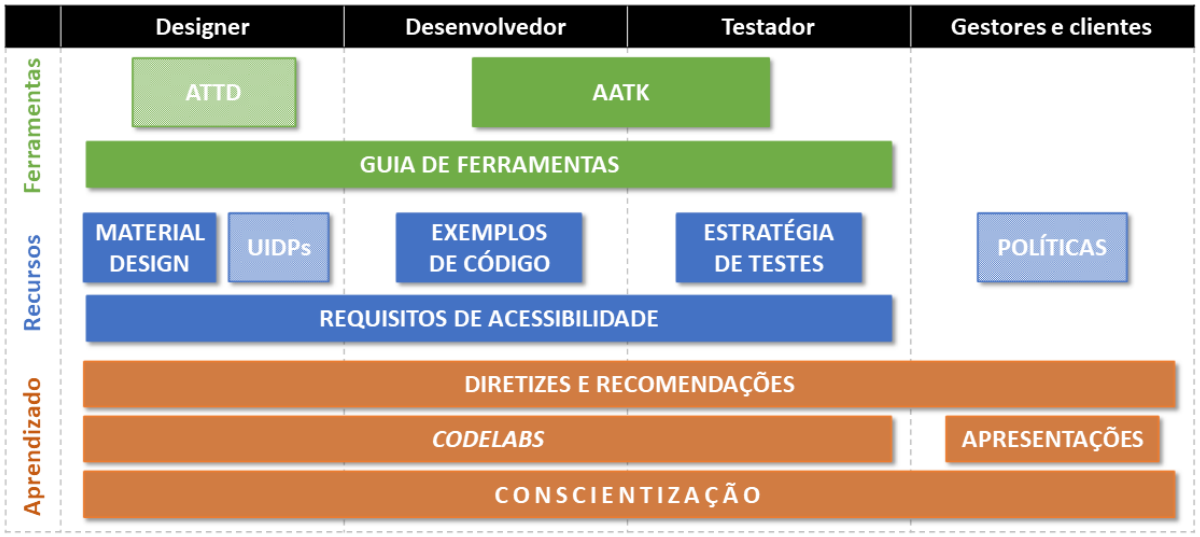
da seguinte maneira:

- **Aprendizado**. Conteúdo que contribui para a formação e conscientização dos envolvidos no projeto.
- **Recursos**. Conteúdo, gerado por esta pesquisa ou derivado da literatura, que pode ser utilizado para orientar as ações de cada papel.
- **Ferramentas**. Recursos para aplicação direta, especialmente para auxiliar em testes de acessibilidade automatizados.

As categorias foram organizadas de cima para baixo em níveis crescentes de abstração, sendo "ferramentas" o nível mais concreto e "aprendizado" o nível mais abstrato.

Na Figura 12, é apresentada a estrutura do AALT, onde os artefatos são organizados de acordo com sua categoria e responsabilidade. Alguns artefatos são aplicáveis a mais de uma responsabilidade. Um exemplo disso é o artefato "Conscientização", que é proposto para todos os eixos de responsabilidade. Ele é considerado o elemento fundamental do *framework*, pois é essencial para a sustentação de toda a estrutura.

Figura 12 – Estrutura do *Framework* AALT



Fonte: Elaborada pelo autor.

A estruturação dos eixos e a seleção dos artefatos para o AALT foram embasadas nas discussões e nos apontamentos apresentados nos capítulos 3 e 4. Especificamente, foram consideradas as principais estratégias sugeridas nos estudos com desenvolvedores, apresentadas no Quadro 11, e as possíveis estratégias para aprofundamento de pesquisa, apresentadas no Quadro 14. Com isso, determinou-se que o *framework* deveria contemplar:

a) Definições claras de responsabilidades.
b) Requisitos de acessibilidade mais objetivos e, quando possível, com afirmações testáveis.

c) Conjuntos de testes adequados para cada fase do projeto, preferencialmente com testes automatizados.
d) Uma estratégia de testes orientada a aumentar o número de testes locais.
e) Programas de treinamento para toda a equipe sobre como testar e melhorar a acessibilidade em aplicativos Android.
f) Campanhas e materiais de conscientização direcionados a todos os participantes, incluindo gestores e clientes.

A divisão do eixo vertical cumpre atender ao item (a). Embora um dos objetivos deste trabalho seja engajar mais os desenvolvedores no processo de obtenção de aplicativos móveis acessíveis, acredita-se que a definição clara de responsabilidades para todos os papéis ajudará a mitigar a sensação de que o problema é sempre de outrem – conforme discutido na Seção 4.4.

Quanto aos demais itens, eles são abordados por meio dos artefatos, organizados de acordo com as suas respectivas categorias. Parte dos artefatos correspondem a ferramentas ou recursos explorados durante este projeto de pesquisa. Por exemplo, o "Guia de Ferramentas" reúne informações sobre as ferramentas para testes de acessibilidade experimentadas pelo autor, além de conter dicas e recomendações de uso para cada uma delas. No entanto, alguns artefatos foram desenvolvidos especificamente para o AALT, e serão detalhados nas seções subsequentes.

A relação completa dos artefatos, contendo suas descrições, categorias e responsabilidades, pode ser vista no Apêndice F.

5.2 Artefato: Requisitos de acessibilidade

Um dos artefatos do AALT foi projetado para atender à necessidade de requisitos de acessibilidade mais concretos e objetivos, conforme destacado pela literatura (BI *et al.*, 2022; LEITE *et al.*, 2021). Para isso, esta pesquisa fundamentou-se em estudos da literatura para definir o escopo das recomendações de acessibilidade a ser considerado. Especificamente, foram considerados os principais problemas de acessibilidade enfrentados por usuários com deficiência visual compilados por Dias, Duarte e Fortes (2021) e as respectivas recomendações de acessibilidade móvel (maRs), para derivar os requisitos de software. Posteriormente, essas recomendações foram correlacionadas com outros estudos (BBC, 2022; SILVA, 2020; OLIVEIRA; FILGUEIRAS, 2019) para determinar quais desses requisitos são os melhores candidatos para implementações de testes de acessibilidade automatizados.

A finalidade deste levantamento foi reescrever as maRs consideradas automatizáveis em declarações testáveis, com inspiração nos critérios de sucesso da WCAG 2.1 (W3C, 2018).

Ao reescrever as maRs como requisitos de software, também foram consideradas as classificações das recomendações do guia da BBC. As recomendações classificadas como

MUST e MUST NOT são mais facilmente testáveis com critérios não subjetivos e podem ser alcançadas com a tecnologia assistiva atual em dispositivos móveis. Por outro lado, as recomendações classificadas como SHOULD ou SHOULD NOT são menos testáveis, mas ainda são consideradas importantes para aplicativos móveis (BBC, 2022). Portanto, as recomendações com classificação MUST e MUST NOT são as mais propensas a serem implementadas com sucesso, visto que estão mais alinhadas com as responsabilidades do desenvolvedor (OLIVEIRA; FILGUEIRAS, 2019).

Na apresentação e descrição das maRs, Dias (2021) já indica a quais recomendações do guia da BBC e a quais critérios de sucesso da WCAG 2.1 elas estão associadas ou se assemelham. Essa informação foi usada para realizar o cruzamento das informações apresentado no Quadro 15. A coluna "Automatizável" apresenta as definições encontradas no estudo de Silva (2020), que classifica cada recomendação como automatizável, parcialmente automatizável, não automatizável ou não aplicável. A coluna "Teste local" indica as recomendações encontradas como regras do AccessibiLint (OLIVEIRA; FRANÇA, 2019), e por isso candidatas a implementação de testes locais.[1] Estão assinaladas como Não Informada (N.I.) onde as respectivas recomendações não foram identificadas.

5.2.1 Reescrita das maRs como requisitos de acessibilidade

Para a reescrita das maRs como requisitos de acessibilidade, com declarações testáveis, a partir das informações identificadas no Quadro 15, definiu-se os seguintes critérios de prioridade:

1. maRs identificadas como candidatas a testes locais.
2. maRs definidas como automatizáveis e classificação MUST ou MUST NOT.
3. maRs definidas como parcialmente automatizáveis e classificação MUST ou MUST NOT.

Por decisão de projeto, foram realizadas as seguintes alterações nas classificações:

- maRs classificadas como SHOULD e definidas como automatizáveis e candidatas a testes locais foram reclassificadas para MUST.
- maRs classificadas como MUST, porém definidas como não automatizáveis, foram reclassificadas para SHOULD.
- maRs não identificadas nos outros estudos foram classificadas como SHOULD.

Com essas alterações, foi possível destacar, na escrita dos requisitos, quais devem ser priorizados nas estratégias de testes, e quais ainda requerem inspeção manual, ou novos estudos para viabilizar suas automações.

[1] Algumas regras do AccessibiLint tem implementação parcial das recomendações. Por decisão de projeto, as respectivas recomendações foram consideradas automatizáveis dentro do mesmo critério da ferramenta.

Quadro 15 – Viabilidade de automação das maRs, de acordo com a literatura.

maR	Critério de Sucesso	Classificação BBC	Automatizável	Teste local
maR01 - Conteúdo não-textual	1.1.1	MUST	Parcial	Sim
maR02 - Orientação do dispositivo	1.4.10	N.I.	N.I.	N.I.
maR03 - Características sensoriais	1.3.3	MUST NOT	Parcial	N.I.
maR04 - Estrutura de visualização	3.2.4	SHOULD	Não	N.I.
maR05 - Conteúdo extenso	1.3.2	N.I.	N.I.	N.I.
maR06 - Redimensionar conteúdo	1.4.4	MUST	Não	N.I.
maR07 - Contraste de Cor	1.4.3	MUST	Sim	Sim
maR08 - Parágrafos	1.4.13	N.I.	N.I.	N.I.
maR09 - Eventos de tela	N.I.	MUST	Parcial	N.I.
maR10 - Rótulos de componentes	2.4.6	MUST	Sim	N.I.
maR11 - Modo teclado	N.I.	SHOULD	Sim	Sim
maR12 - Acionabilidade	N.I.	MUST	Parcial	N.I.
maR13 - Navegação com foco ordenado e visível	2.4.3 2.4.7	MUST	Sim*	N.I.
maR14 - Título da página	2.4.10	MUST	N.I.	N.I.
maR15 - Agrupamento de elementos	1.3.1	MUST	Não	N.I.
maR16 - Navegação no app	N.I.	N.I.	N.I.	N.I.
maR17 - Tamanho do alvo de touch	3.2.4	MUST	Sim	Sim
maR18 - Espaçamento	N.I.	SHOULD	Sim	Sim
maR19 - Feedback de ações	N.I.	SHOULD	Não	N.I.
maR20 - Formato dos dados	3.3.2	MUST	Sim	Sim
maR21 - Estrutura de formulário	N.I.	MUST	N.I.	N.I.
maR22 - Tecnologia Assistiva	N.I.	N.I.	N.I.	N.I.
maR23 - Confirmação	3.3.4	N.I.	N.I.	N.I.
maR24 - Ajuda contextual	3.3.5	SHOULD	Não	N.I.
maR25 - Relatório de erros	3.1.1	MUST	Parcial	N.I.

Fonte: Elaborada pelo autor.

A reescrita das maRs, após a aplicação desses critérios, é apresentada no Quadro 16, e podem ser utilizadas como um documento de requisitos de acessibilidade para a realização de testes de acessibilidade no Android. Cada requisito foi escrito no formato: **alvo do teste [condição] + MUST|SHOULD + critério do teste**. Para este documento, e por efeito prático, *must* foi traduzido como "deve" e *should* como "deveria". Adicionalmente, foi acrescentada a indicação de quais requisitos são candidatos a testes locais. Além daqueles relacionados às regras do AccessibiLint, e que portanto podem ser realizados por verificações estáticas, o requisito R09 também foi considerado candidato, pois já foi implementado pelos autores, conforme será discutido na Seção 5.3. Foram identificados como "Não" candidatos os requisitos que se relacionam com recomendações definidas como não automatizáveis, e os demais receberam a denominação de "Não Definido" (N.D.), pois não é possível confirmar ou descartar a possibilidade de automação destes requisitos com base na literatura.

Este conjunto de requisitos de acessibilidade ajudou a orientar o desenvolvimento do conjunto de testes que será apresentado na próxima seção, mas, acima de tudo, tem como objetivo servir como referência para desenvolvedores e pesquisadores.

Quadro 16 – Requisitos de acessibilidade por ordem de prioridade para automação de testes.

Id.	Requisito de Acessibilidade	maR	Candidato a teste local
R01	Elementos de texto devem utilizar uma taxa de contraste de pelo menos 4.5:1 entre a cor do texto e a cor do background.	maR07	Sim
R02	Elementos de interação devem ter um tamanho mínimo de 48x48dp.	maR17	Sim
R03	Campos de entrada de formulário devem indicar o formato de dados esperado, como datas e entradas numéricas.	maR20	Sim
R04	Todo conteúdo não textual deve ter uma descrição de texto alternativa.	maR1	Sim
R05	Elementos de interação devem ter um espaçamento mínimo de 8dp da borda da tela e pelo menos 9dp de espaçamento entre eles.	maR18	Sim
R06	Campos de entrada de formulário devem garantir que o modo de teclado seja apropriado para o conteúdo do campo esperado.	maR11	Sim
R07	Todas as telas do app devem ter um título único e facilmente identificável na parte superior da tela.	maR14	Sim
R08	Todas as telas do app devem suportar navegação baseada em foco, destacando o componente focado.	maR13	Sim
R09	Para cada controle de formulário deve haver um TextView com o atributo labelFor associado a ele ou deve ter o atributo hint fornecido.	maR10	Sim
R10	Todos os possíveis erros devem ser claramente informados ao usuário.	maR25	Sim
R11	Todo elemento de interação deve permitir identificar o estado de acionabilidade, visualmente e por leitor de tela.	maR12	N.D.
R12	Todos os elementos que dependem de informações sensoriais (como localização no mapa ou cores para representar informações) para que o conteúdo seja compreendido devem fornecer descrição textual associada.	maR03	N.D.
R13	Quaisquer mensagens ou alterações na tela, como pop-ups, notificações e atualizações dinâmicas de conteúdo, devem ser anunciadas pelo leitor de tela.	maR09	N.D.
R14	A estrutura do formulário deveria ser organizada com um campo por linha, evitando múltiplas colunas.	maR21	N.D.
R15	Ações iniciadas a partir da interação do usuário, e que não podem ser desfeitas, deveriam solicitar confirmação antes de serem executadas.	maR23	N.D.
R16	Todas as Views personalizadas deveriam oferecer suporte à tecnologia assistiva.	maR22	N.D.
R17	Os parágrafos deveriam ter uma altura de interlinha de pelo menos 1,5 vezes o tamanho da fonte e o espaçamento dos parágrafos seguintes de pelo menos 2 vezes o tamanho da fonte.	maR08	N.D.
R18	As telas internas do app deveriam fornecer um elemento de navegação para retornar à tela inicial.	maR16	N.D.
R19	Todas as telas do app deveriam manter um layout consistente em diferentes tamanhos e orientações de tela.	maR02	N.D.
R20	Todas as telas do app deveriam ser redimensionáveis.	maR06	Não
R21	Itens relacionados deveriam ser agrupados.	maR15	Não
R22	Quando aplicável, os controles do formulário de dados de entrada deveriam fornecer dicas para conclusão.	maR24	Não
R23	Todas as ações do usuário executadas em segundo plano deveriam fornecer feedback de sucesso ou falha.	maR19	Não
R24	Toda a estrutura e layout do app deveriam permanecer consistentes em diferentes telas.	maR04	Não
R25	Todo o conteúdo que exceda as dimensões da tela (por exemplo, textos longos) deveriam ser apresentado usando representações alternativas apropriadas, como listas paginadas ou navegação hierárquica.	maR05	N.D.

Fonte: Elaborada pelo autor.

5.3 Artefato: *Kit* de testes locais de acessibilidade (AATK)

Com o objetivo de facilitar a verificação de requisitos de acessibilidade, mediante a condução de testes locais, foi desenvolvida uma nova ferramenta focada nos problemas de acessibilidade mais frequentes para pessoas com deficiência visual e nos *widgets* mais utilizados, apresentados na Quadro 7. O *Kit* de Testes de Acessibilidade Automatizados (AATK - *Automated Accessibility Testing Kit*) é uma biblioteca Android escrita em Java e disponibilizada publicamente para ajudar desenvolvedores na criação de testes de acessibilidade para projetos Android. O AATK oferece um conjunto de testes de acessibilidade automatizados, projetados para serem executados com o Robolectric, permitindo que sejam conduzidos como testes locais, sem a necessidade de um dispositivo físico ou emulado.

O *kit* inclui cinco testes para problemas comuns de acessibilidade que afetam negativamente a usabilidade de aplicativos nativos para usuários com deficiência visual. Esse conjunto inicial de testes pode ser estendido, uma vez que cada teste implementa uma interface que automatiza a execução por toda a hierarquia de *layout*, o que possibilita que novos testes possam ser criados a partir dessa mesma interface. Os testes atualmente contemplados são listados no Quadro 17.

Quadro 17 – Testes disponíveis no AATK.

Teste	**Nome da classe de teste**	**Requisito (Ri)**
Contraste de cor	TestAdequateContrastRatio	R01
Tamanho do alvo de toque	TestTouchTargetSize	R02
Texto alternativo	TestMustHaveAlternativeText	R04
Espaçamento	TestInteractionElementSpacing	R05
Rótulo de componente	TestMustFormControlHaveLabel	R09

Fonte: Elaborada pelo autor.

Um exemplo de como preparar uma classe de teste para usar a biblioteca e adicionar um teste, por exemplo, para verificar uma taxa de contraste adequada, é demonstrado no Código-fonte 4.

Código-fonte 4 – Teste da taxa de contraste de um TextView com o AATK

```
1:  @RunWith(RobolectricTestRunner.class)
2:  public class MyActivityTest {
3:      private View rootView;
4:      private AccessibilityTestRunner runner;
5:
6:      @Rule
7:      public ErrorCollector collector = new ErrorCollector();
8:
```

```
 9:          @Before
10:          public void setUp() {
11:              Activity activity = Robolectric.buildActivity(MyActivity.class).create().get()
        ;
12:
13:              // Get the root node of the view hierarchy
14:              rootView = activity.getWindow().getDecorView().getRootView();
15:              runner = new AccessibilityTestRunner(collector);
16:          }
17:
18:          @Test
19:          public void mustUseAdequateContrastRatio(){
20:          runner.runAccessibilityTest(rootView, new TestAdequateContrastRatio());
21:        }
22:      }
```

De forma análoga, os demais testes do *kit* podem ser instanciados, ou novos testes podem ser criados a partir de uma das interfaces disponíveis. Por exemplo, a interface IAccessibilityTestViewHierarchy, apresentada no Código-fonte 5, permite ao executor (AccessibilityTestRunner) percorrer a hierarquia de Views de uma Activity e realizar a verificação em cada View. A interface IAccessibilityTestSelfContained, por outro lado, é útil para testes de acessibilidade que implementam sua própria forma de navegar pela hierarquia de Views.

Código-fonte 5 – Interface do AATK para criação de testes de acessibilidade que percorram a hierarquia de Views

```
1: public interface IAccessibilityTestViewHierarchy {
2:     /**
3:      * Get all applicable widgets for given test
4:      * @return an array of applicable widget classes
5:      */
6:     Class[] getApplicableWidgets();
7:
8:     /**
9:      * Run test for current view. Each test implements your own checks.
10:     * Any error is added to the collector
11:     * @param view current view
12:     * @param collector error collector
13:     */
14:    void runTest(View view, ErrorCollector collector);
15: }
```

É possível, ainda, executar todas as checagens prontas do *kit* em um só teste, invocando o método AccessibilityTestRunner.runAllAccessibilityTests. No entanto, a criação de testes individualizados pode ser uma boa prática, dependendo da estratégia de testes, pois facilita a identificação e correção de falhas, além de oferecer maior controle e unicidade ao teste.

5.3.1 Avaliação do AATK

Para avaliar a compreensão e usabilidade da biblioteca AATK, foram conduzidas avaliações do *kit* com desenvolvedores de diferentes níveis de experiência. A avaliação ocorreu no contexto de turmas de graduação de disciplinas de Teste de Software do ICMC-USP. Estavam previstos,

na ementa do curso, conteúdos sobre testes de usabilidade e acessibilidade. Foi feito o convite, pelo docente responsável, para que fossem apresentados conteúdos teóricos e práticos, incluindo ferramentas de testes. Assim, foi realizada uma apresentação teórica, seguida da apresentação das ferramentas disponíveis e a aplicação da atividade de avaliação.

Foram apresentadas as seguintes ferramentas:

- Scanner de acessibilidade, como exemplo de ferramenta para testes grandes.
- Espresso, como exemplo de ferramenta para testes médios.
- AATK, como exemplo de ferramenta para testes pequenos.

Para a realização da atividade, todos os participantes receberam o *link* para um formulário Google (ver Apêndice G) que continha as instruções divididas em quatro etapas:

1. Preenchimento do **Termo de Consentimento Livre e Esclarecido (TCLE)**.
2. **Identificação de perfil**: 14 perguntas para entender a experiência do participante com ferramentas e conceitos envolvidos.
3. *Link* para um ***codelab*** com as instruções, passo-a-passo, para a realização da atividade.
4. **Registro de experiência**: perguntas para registrar a experiência do participante com a ferramenta utilizada e os resultados obtidos.

Para esta última etapa, optou-se por utilizar a *System Usability Scale* (SUS) (BROOKE, 1996), uma escala reconhecida e consolidada para mensurar a percepção de usabilidade de um sistema. A escolha pela SUS se deu devido ao foco principal do *kit* AATK ser a avaliação da estratégia de testes e sua possível adesão por parte dos desenvolvedores, não estando a eficácia do AATK em identificar problemas de acessibilidade no escopo deste trabalho.

As avaliações envolveram a realização de melhorias de acessibilidade em um aplicativo de exemplo, utilizando os testes do AATK como guia. Grupos de controle realizaram as mesmas tarefas com a ajuda do Espresso e do Scanner de Acessibilidade. Ou seja, cada participante deveria realizar os testes com uma das três ferramentas. Para qualquer das opções, as tarefas compreendiam:

1. Preparar e configurar a ferramenta.
2. Verificar três problemas de acessibilidade (taxa de contraste de cores, conteúdos não textuais sem descrição alternativa e tamanho dos alvos de toque), e para cada um deles:
   a) Executar teste.
   b) Visualizar resultados.
   c) Efetuar melhorias.
   d) Re-executar os testes.

Condução

As avaliações aconteceram no dia 5 de Junho de 2023, em dois períodos. Participaram duas turmas: a primeira, no período vespertino, composta predominantemente por alunos do nono período do curso do Bacharelado em Ciências da Computação (BCC), e a segunda, no período noturno, composta predominantemente por alunos do terceiro período do Bacharelado em Sistemas de Informação (BSI). No entanto, também houve a presença de alunos de Engenharia da Computação na turma vespertina, além de dois alunos do BSI. No total, 51 estudantes participaram da atividade, sendo 31 do curso de BSI, 12 de BCC e 8 de Engenharia da Computação. Os alunos estavam em diferentes semestres, variando do primeiro ao 11º. A maioria estava no terceiro semestre (24/51), e treze estavam no nono semestre. Portanto, eles têm experiências distintas em desenvolvimento de software.

Para a realização das tarefas com as ferramentas Scanner de Acessibilidade e Espresso, dispositivos Android (*tablets*) foram disponibilizados aos participantes, uma vez que os equipamentos do laboratório didático não permitiam a instalação de dispositivos emulados até a data da realização do experimento. No entanto, alguns participantes optaram por utilizar seus próprios *smartphones* para realizar as tarefas, por iniciativa própria. Essa opção não interferiu na realização da atividade.

Tentou-se distribuir as atividades igualmente entre as ferramentas em cada turma, porém, especialmente na turma do período noturno, essa divisão ficou prejudicada. Devido à dimensão da turma, os alunos se dividiam em duas salas interligadas. A apresentação da atividade acontecia a partir de uma delas. Os participantes que estavam na sala diferente do aplicador enfrentaram dificuldades para entender as instruções comunicadas verbalmente para a distribuição das avaliações por ferramenta, mas não reportaram o problema e seguiram apenas as orientações disponíveis no formulário fornecido, onde a escolha da ferramenta era livre para o participante. Por essa razão, e presumivelmente por se tratar de um aplicativo, ou seja, uma ferramenta mais familiar aos participantes – sobretudo em uma turma de terceiro período –, o Scanner de Acessibilidade foi utilizado por uma amostra superior à metade do total de participantes (27/51).

Além disso, em ambas as turmas, os participantes enfrentaram dificuldades para concluir as tarefas utilizando o Espresso. Por exemplo, na turma do período noturno, parte daqueles designados para utilizar essa ferramenta não conseguiram finalizar as tarefas, embora todas as implementações tenham sido realizadas corretamente. Isso ocorreu devido a uma mensagem de *timeout* exibida no console do Android Studio após alguns minutos de espera.

Espera-se, com esses relatos, contribuir para o planejamento de futuras atividades relacionadas. Ainda que a amostra de avaliações do AATK tenha sido reduzida (12/51), foi possível inferir alguns resultados, que são apresentados a seguir.

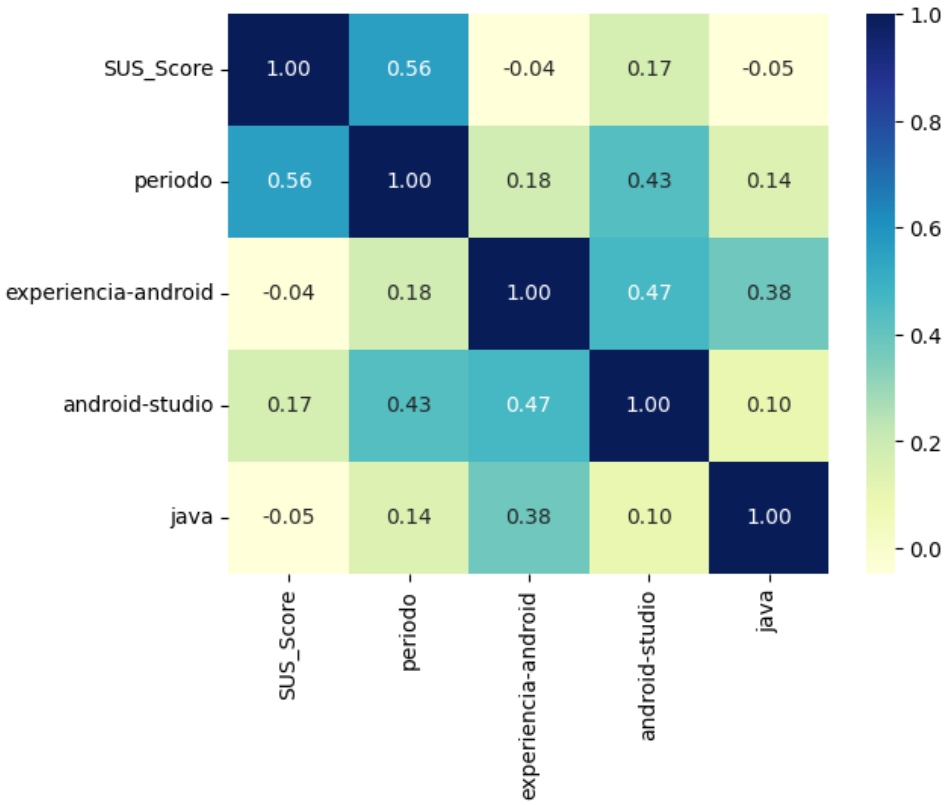
Resultados

Ao utilizar a System Usability Scale (SUS) para avaliar a usabilidade, é possível obter um único número que representa uma medida composta da usabilidade geral do sistema em estudo, conhecido como pontuação SUS (*SUS Score*) (BROOKE, 1996). Embora essa pontuação isoladamente não forneça um diagnóstico completo (USABILITY.GOV, 2023), foram calculadas as médias das pontuações SUS para cada ferramenta: Scanner de Acessibilidade (74,3), Espresso (32,3) e AATK (51,9).

Além disso, foi realizada uma análise para verificar como a pontuação SUS se relacionou com outras variáveis da identificação de perfil. O mapa de calor apresentado na Figura 13 mostra a correlação entre as variáveis pontuação SUS, período letivo, experiência em desenvolvimento Android, experiência com o Android Studio e experiência com a linguagem de programação Java.[2] Os valores do mapa de calor são representados por cores, que variam de acordo com o grau de correlação. Valores próximos de 1 indicam uma correlação forte, valores próximos de 0 indicam uma correlação fraca e valores próximos de -1 indicam uma correlação negativa forte. No mapa de calor, é possível observar que a pontuação SUS apresenta uma correlação forte com o período letivo. Isso significa que, quanto mais avançado o período letivo, maior a pontuação SUS.

[2] Para o cálculo da matriz de correlação, foi utilizado o método de Pearson (BENESTY *et al.*, 2009).

Figura 13 – Mapa de calor das correlações com a pontuação SUS

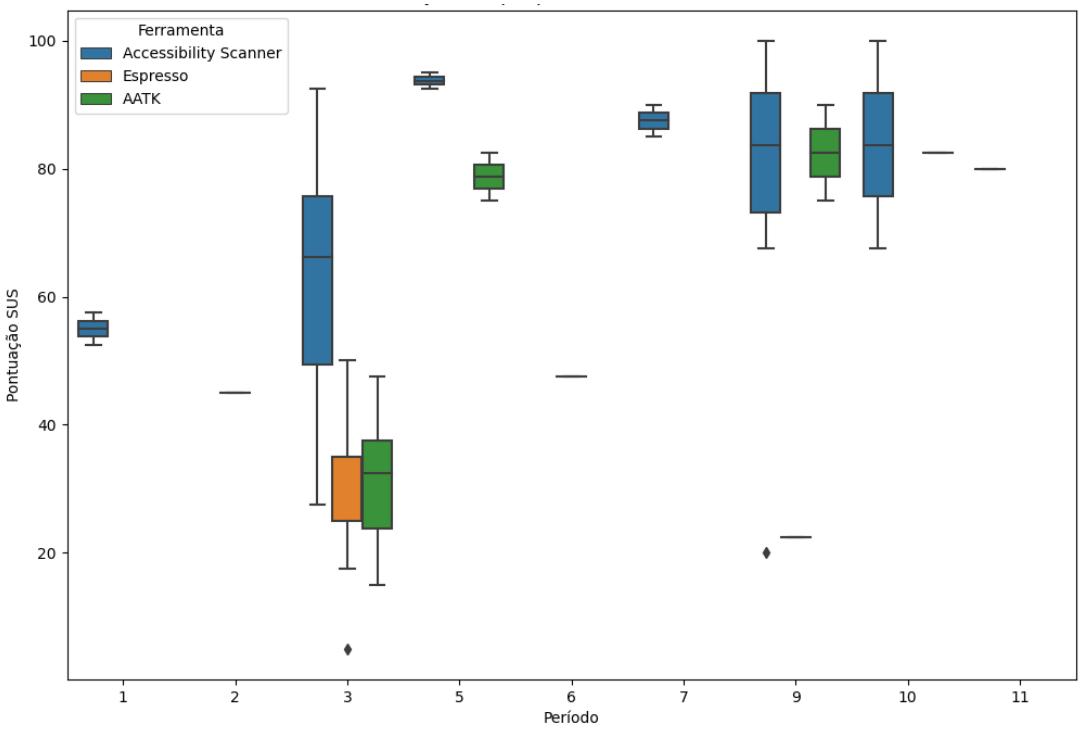


Fonte: Elaborada pelo autor.

A partir desta análise, verificou-se a distribuição da pontuação SUS por ferramenta, de acordo com o período letivo. Para visualizar essas informações, utilizou-se o *boxplot*.[3] Na Figura 14, pode-se observar que, entre os alunos do terceiro período, a pontuação do Scanner de Acessibilidade foi significativamente maior em comparação com as outras ferramentas. No entanto, essa diferença é atenuada para estudantes mais avançados. Também ficam destacadas as diferenças nas amostras de cada ferramenta e as dificuldades encontradas para obter resultados com o Espresso.

[3] Um *boxplot* é um gráfico que resume a distribuição de um conjunto de dados por meio de um retângulo que representa o intervalo interquartil (IQR), uma linha que representa a mediana e "bigodes" que indicam a amplitude dos dados. Ele fornece uma visão geral da distribuição dos dados de forma concisa.

Figura 14 – Pontuação SUS por período letivo e ferramenta

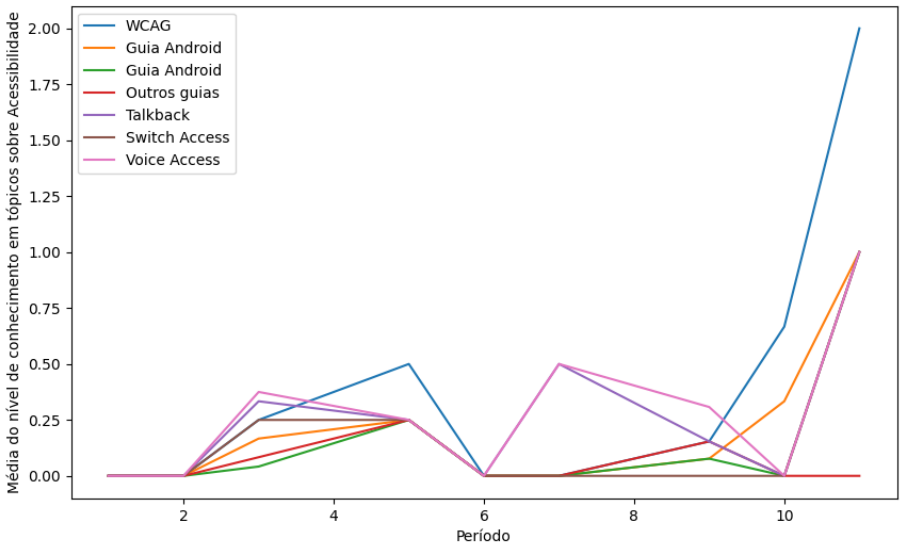


Fonte: Elaborada pelo autor.

Não foram identificadas correlações significativas entre o nível de conhecimento em tópicos sobre acessibilidade e a pontuação SUS. No entanto, foi analisada a relação entre essa variável (conhecimento sobre acessibilidade) e o período letivo dos participantes. Para isso, as respostas às perguntas sobre esses tópicos foram convertidas em uma escala numérica, onde "Desconheço" correspondia a 0, "Conheço, mas NÃO SEI utilizar" correspondia a 1 e "Conheço, e SEI utilizar" correspondia a 2. Em seguida, foi calculada a média da pontuação obtida por tópico para cada período. Na Figura 15, é possível observar um aumento significativo na média a partir dos últimos períodos.[4]

[4] De acordo com informações obtidas no sistema Jupiterweb, da graduação da USP, a disciplina de "Acessibilidade em Sistemas Computacionais" é oferecida atualmente no sexto período letivo para o BSI, como disciplina obrigatória, e no oitavo período para o BCC, como disciplina eletiva. Não foi possível confirmar, no entanto, se a grade atual é a mesma cumprida pelos participantes.

Figura 15 – Conhecimento em acessibilidade por período letivo

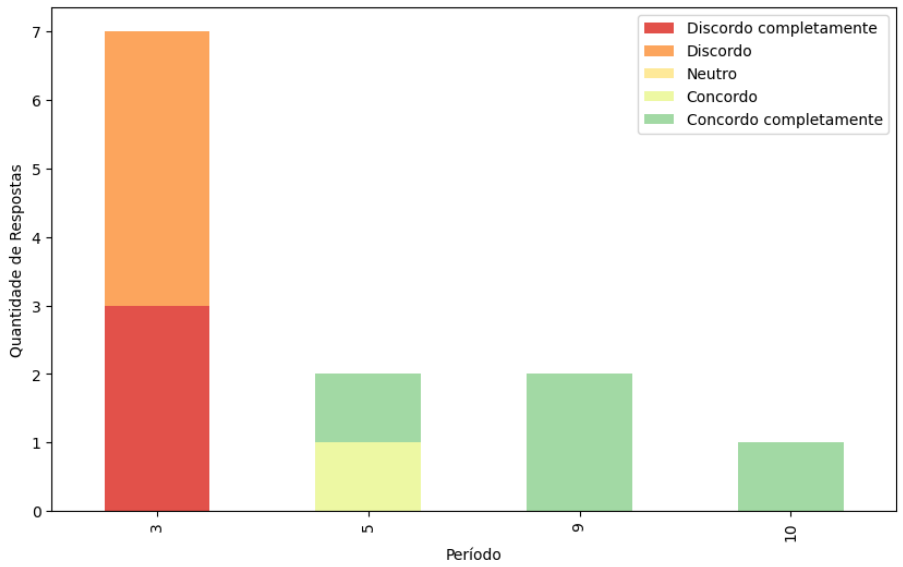


Fonte: Elaborada pelo autor.

Discussão

Os resultados obtidos indicam que as dificuldades encontradas estão mais relacionadas à formação dos participantes. Conforme demonstrado na Figura 16, dos participantes que utilizaram o AATK, cinco consideraram a ferramenta fácil de usar e estavam no quinto período ou além. Por outro lado, sete participantes discordaram dessa afirmação e todos estavam no terceiro período.

Figura 16 – Respostas à afirmação “eu achei a ferramenta fácil de usar” para o AATK, por período letivo.



Fonte: Elaborada pelo autor.

A maior preferência pelo Scanner de Acessibilidade entre os estudantes dos primeiros anos sugere que sua interface gráfica é mais intuitiva e, portanto, mais fácil de usar para testes de

acessibilidade por novatos. No entanto, é importante ressaltar que essa ferramenta não é prática para uso em processos de desenvolvimento com ciclos curtos. A similaridade na pontuação SUS entre os estudantes mais avançados indica uma maior afinidade desses desenvolvedores mais experientes com ferramentas que envolvem programação direta. No entanto, devido à baixa amostra de resultados com o Espresso, a comparação entre essa ferramenta e o AATK se torna difícil.

Em relação aos conceitos de acessibilidade, uma parcela significativa dos participantes não estava familiarizada com recursos e diretrizes de acessibilidade. Por exemplo, 40 dos 51 participantes não conheciam as diretrizes da WCAG, 43 não conheciam o Guia de Acessibilidade para Desenvolvedores Android e 47 não conheciam o Guia de Acessibilidade para Desenvolvedores iOS. No entanto, entre aqueles que estavam familiarizados com esses recursos, a maioria estava nos períodos mais avançados, destacando a importância da formação adequada sobre o tema.

Quanto às ferramentas de testes automatizados, também houve um número significativo de participantes que não as conhecia. Por exemplo, 42 participantes não conheciam o Android Lint, 44 não conheciam o Scanner de Acessibilidade, 45 não conheciam o UI Automator e 43 não conheciam o Espresso. Apenas o JUnit era conhecido por todos os participantes (exceto dois alunos). No entanto, mesmo entre aqueles que conheciam essas ferramentas, nenhum deles afirmou saber utilizá-las, o que indica a necessidade de treinamentos específicos nessa área.

5.4 Artefato: *Codelabs*

Uma das principais lacunas identificada na literatura é a falta de treinamentos práticos e direcionados que abordam os desafios da acessibilidade em aplicativos móveis, conforme relatado por desenvolvedores nos estudos de Antonelli *et al.* (2018), Inal, Rızvanoğlu e Yesilada (2019), Leite *et al.* (2021) e Miranda e Araujo (2022).

Nesse sentido, considerou-se crucial dentro da categoria de aprendizado do AALT, a oferta de uma coleção de treinamentos em formato de *codelabs*. Esses são baseados na estrutura de um tutorial guiado, proporcionando uma jornada passo a passo que permite aos usuários aprender fazendo. Os *codelabs* desempenham um papel fundamental no AALT, servindo como uma plataforma interativa para o treinamento de *designers*, desenvolvedores e testadores.

Os *codelabs* do Google,[5] que fornecem uma variedade de tutoriais interativos, incluindo alguns sobre acessibilidade, foram uma importante inspiração para os *codelabs* no AALT. No entanto, esses *codelabs* tendem a se concentrar nas ferramentas oferecidas diretamente pela plataforma Android, como o Scanner de Acessibilidade e o Espresso. Para o AALT, a intenção é

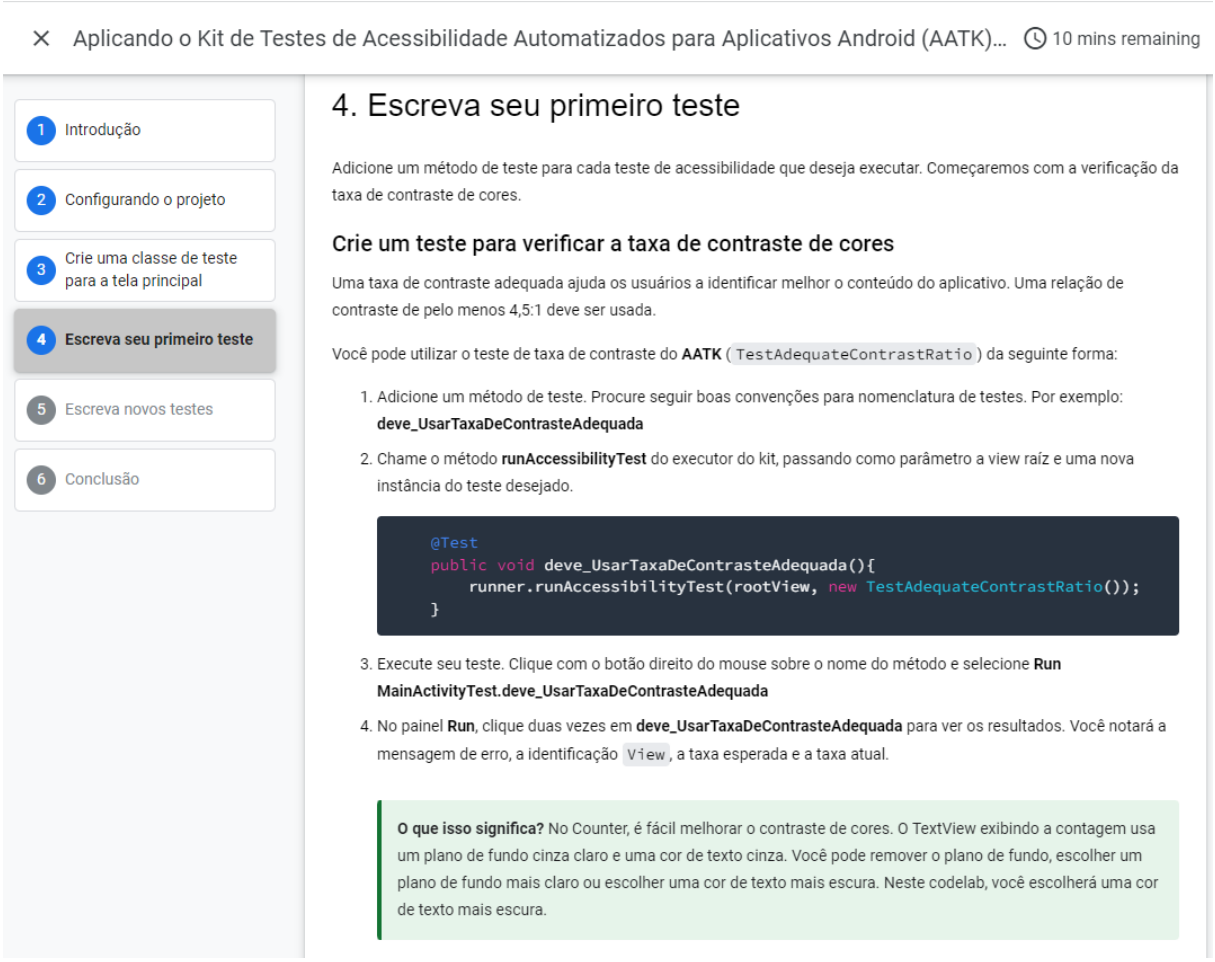
[5] <https://codelabs.developers.google.com/>

abranger uma gama mais ampla de ferramentas e técnicas, proporcionando uma cobertura mais completa dos desafios da acessibilidade em aplicativos móveis.

Assim como foi feito para a seleção de artefatos, na coleção de *codelabs* do AALT também são reunidos *codelabs* do Google e criados novos treinamentos. Por exemplo, no codelab "Aplicando o *Kit* de Testes de Acessibilidade Automatizados para Aplicativos Android (AATK) para o projeto do *app Counter*", o desenvolvedor é orientado a configurar a biblioteca AATK em um projeto de exemplo, criar alguns testes de acessibilidade usando as classes do *kit*, e realizar melhorias de acessibilidade no aplicativo até que os testes sejam bem-sucedidos.

No exemplo mostrado na Figura 17, é ensinado como utilizar o AATK para inserir um teste para verificar a taxa de contraste de cores. Além de criar e executar o teste, o desenvolvedor aprende como realizar as alterações necessárias para corrigir este problema de acessibilidade, além de receber informações adicionais sobre como isso pode afetar o acesso ao aplicativo por pessoas com deficiência visual.

Figura 17 – Quarto passo do *codelab* do AATK - Inclusão de teste para verificar a taxa de contraste de cores



Fonte: Elaborada pelo autor.

Os *codelabs* foram o formato selecionado para as avaliações das ferramentas, apresentada na Subseção 5.3.1. Embora não tenham sido parte direta da avaliação, eles se revelaram um recurso de ensino potencialmente valioso, pois os participantes conseguiram realizar as atividades com bastante autonomia. Além disso, o material contribuiu para a disseminação de informações sobre acessibilidade entre os participantes.

Para a expansão futura do AALT, considera-se crucial a inclusão de um *codelab* para cada ferramenta apresentada. Isso tornaria o AALT uma fonte abrangente e acessível de aprendizado prático sobre acessibilidade em aplicativos móveis.

Por fim, sugere-se que as empresas considerem a inclusão desses *codelabs* – ou de outros similares – em seus programas de treinamento e qualificação para projetos de software. Isso não apenas aumentaria o conhecimento e a competência da equipe em relação à acessibilidade, mas também demonstraria um compromisso da empresa com a criação de software inclusivo e acessível.

5.5 Informações e orientações para utilização do AALT

O AALT foi idealizado para ser utilizado em projetos de desenvolvimento de aplicativos Android nativos. Seu objetivo é orientar o time de desenvolvimento sobre como cada equipe pode atuar para reduzir as barreiras de acessibilidade nos aplicativos produzidos. O *framework* disponibiliza seus artefatos para auxiliar nesse processo. Porém, é importante que cada time e cada equipe encontrem suas próprias configurações e determinem quais instrumentos desejam utilizar. Assim, a estrutura do AALT deve funcionar principalmente como um arcabouço para que os líderes de projeto estabeleçam suas estratégias, utilizando os artefatos disponíveis ou adicionando novas ferramentas e recursos conforme necessário.

Formas de acesso

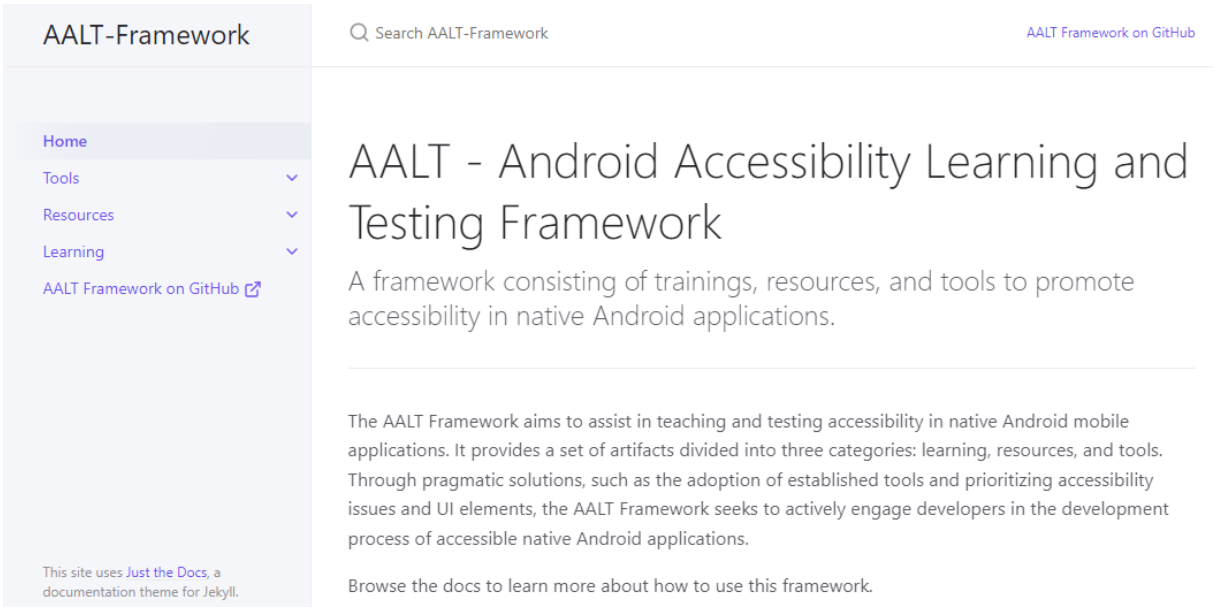
Para disponibilizar todos os artefatos mencionados, produzidos ou reunidos, foi criado um espaço no GitHub (GITHUB, 2023), plataforma de hospedagem de código-fonte e arquivos com controle de versão com uma base de usuários de mais de 94 milhões de desenvolvedores.[6] O espaço, configurado como uma organização,[7] reúne repositórios para cada projeto criado para o AALT, além recursos, *codelabs* e projetos de exemplo.

[6] <https://octoverse.github.com/>

[7] As organizações no GitHub são contas compartilhadas que permitem a colaboração entre empresas e projetos de código aberto em várias iniciativas simultâneas, oferecendo recursos administrativos e de segurança avançados (GITHUB, 2023).

Todo o conteúdo está organizado em um site, hospedado com o GitHub Pages.[8] Uma captura de tela da página inicial do AALT Framework pode ser vista na Figura 18. Com essa configuração, espera-se reforçar o objetivo principal deste trabalho, de oferecer soluções pragmáticas a partir de uma visão realista dos desenvolvedores, adotando práticas com as quais estão familiarizados, e reduzindo o esforço de implementação, usando ferramentas consolidadas.

Figura 18 – Página inicial do AALT Framework



Fonte: Elaborada pelo autor.

O AALT Framework está aberto a contribuições. Os *links* de acesso são:

1. <https://github.com/AALT-framework> (acesso às bibliotecas e recursos)
2. <https://aalt-framework.github.io/> (site público)
3. <https://aalt-framework.github.io/codelabs> (*codelabs*)

5.6 Considerações finais

Neste capítulo foi apresentado o AALT Framework, um conjunto de soluções destinadas a melhorar a acessibilidade em aplicativos Android nativos, com foco especial nos desafios enfrentados por pessoas com deficiência visual. Este *framework* é o resultado do realinhamento das estratégias reunidas ao longo deste projeto de pesquisa.

O conjunto de requisitos de acessibilidade, por exemplo, baseia-se na compilação de (DIAS; DUARTE; FORTES, 2021), que reúne os problemas de acessibilidade enfrentados por pessoas com deficiência visual mais recorrentes na literatura, e adiciona uma nova camada. A

[8] O GitHub Pages é um serviço de hospedagem de sites estáticos que permite o uso direto de arquivos HTML, CSS e JavaScript de um repositório no GitHub (GITHUB, 2023).

partir do cruzamento com outros estudos, as recomendações maRs foram reescritas de modo a facilitar a criação de testes de acessibilidade por desenvolvedores. Além disso, a indicação, com base na literatura, dos requisitos de acessibilidade que são candidatos para automação de testes locais, pode orientar o avanço no estado da arte.

O desenvolvimento do AATK teve como premissas as priorizações de problemas de acessibilidade e o endereçamento de elementos de UI, conforme apresentado no Quadro 7. Considera ainda a estratégia de testes em pirâmide, oferecendo recursos para que os desenvolvedores realizem testes locais de acessibilidade. Estes testes podem ser planejados e executados durante o desenvolvimento, antecipando as verificações de acessibilidade de modo prático, o que, pela experiência deste autor, colabora com o comprometimento dos desenvolvedores com o tema.

Embora a motivação essencial deste trabalho tenha sido apresentar soluções mais integradas ao ambiente de desenvolvimento, a revisão de estudos envolvendo desenvolvedores, apresentados na Subseção 3.2.2 e as entrevistas com profissionais da indústria, apresentada na Seção 4.4 revelaram a importância de fatores organizacionais. Questões como formação adequada dos profissionais, distribuição de responsabilidades, alocação de tempo e recursos, e conscientização de toda a equipe a respeito sobre a acessibilidade são fundamentais em todos os aspectos do desenvolvimento e influenciam a adoção e implementação das soluções propostas. Essa preocupação é refletida no AALT Framework tanto em sua estrutura, com eixos verticais de responsabilidades e um eixo horizontal de aprendizado, quanto em sua composição, com artefatos de treinamento, formação e conscientização.

Por fim, as formas de disponibilizar o AALT Framework e seus artefatos foram definidas principalmente com o objetivo de promover a acessibilidade digital junto à comunidade de desenvolvedores.

CAPÍTULO

6

CONCLUSÕES

Nesta pesquisa, foram investigados:

- os problemas de acessibilidade mais recorrentes em aplicativos Android, especialmente aqueles que afetam pessoas com deficiência visual;
- técnicas e processos para o desenvolvimento de aplicativos móveis considerando requisitos de acessibilidade;
- os desafios enfrentados por desenvolvedores na produção de aplicativos móveis acessíveis;
- e as ferramentas atualmente disponíveis para testes de acessibilidade em aplicativos Android nativos.

Foi constatado que, apesar de desenvolvedores possuírem conhecimento superficial sobre a importância da acessibilidade e contarem com um repertório de diretrizes e recomendações para guiar o desenvolvimento de aplicativos móveis acessíveis, a adoção de práticas e técnicas que asseguram essa acessibilidade ainda é limitada. Durante o estudo, foram identificadas lacunas nas ferramentas e técnicas disponíveis atualmente, como a pouca oferta de opções para testes locais de acessibilidade – aqueles que não precisam de um dispositivo físico ou emulado para serem executados. Também foram identificados desafios como a falta de treinamento, falta de interesse e má compreensão sobre as responsabilidades dos atores de uma equipe de desenvolvimento.

Diante desses desafios, foi proposto o AALT Framework com o objetivo de auxiliar no ensino e nos testes de acessibilidade em aplicativos móveis. O AALT oferece um conjunto de artefatos divididos em três categorias: aprendizado, recursos e ferramentas. Por meio de soluções pragmáticas, como a adoção de ferramentas consolidadas e a priorização de problemas de acessibilidade e elementos de UI, o AALT tem o intuito de engajar os desenvolvedores de forma ativa no processo de desenvolvimento de aplicativos acessíveis Android nativos.

A avaliação de um dos artefatos do AALT, o AATK, proporcionou indícios de facilidade de uso da ferramenta por desenvolvedores com alguma experiência, embora mais estudos sejam

necessários para avaliar sua aplicabilidade projetos reais.

Nas próximas seções, serão abordadas as principais contribuições e limitações deste trabalho, além de serem sugeridas possíveis direções para futuras pesquisas. Em seguida, será apresentada a seção de considerações finais, encerrando esta dissertação.

6.1 Principais contribuições

A principal contribuição desta pesquisa de mestrado foi a disponibilização do AALT – um *framework* composto por treinamentos, recursos e ferramentas para promover a acessibilidade em aplicativos Android nativos.

Entre os artefatos produzidos para o AALT Framework a partir desta pesquisa, destacam-se:

- Um conjunto de requisitos de acessibilidade derivados das 25 recomendações de acessibilidade propostas por Dias, Duarte e Fortes (2021). As recomendações foram reescritas para serem direcionadas a critérios de testes de acessibilidade.
- O AATK – um *kit* de testes de acessibilidade automatizados para aplicativos Android nativos, disponibilizado como uma biblioteca Android.
- Uma coleção de treinamentos, em formato de *codelabs*, para capacitar desenvolvedores e disseminar informações sobre a acessibilidade digital em aplicativos móveis.

Além dessas, destacam-se como contribuições acadêmicas deste trabalho:

- O mapeamento sistemático da literatura, que iniciou com mais de 1500 resultados de pesquisa. Destes, 46 foram selecionados para a extração de dados. As informações obtidas apresentaram uma visão geral sobre como a acessibilidade e a usabilidade tem sido consideradas nos processos de desenvolvimento, de acordo com a literatura.
- O relato das experiências com as ferramentas de testes de acessibilidade, e de como elas podem ser orientadas para uma estratégia de testes em pirâmide.
- Discussões sobre o uso de técnicas de TDD para a inclusão de requisitos de acessibilidade.

De modo especial, este trabalho concentrou-se na identificação e organização de soluções de incorporação de acessibilidade em aplicativos móveis integradas ao ambiente de desenvolvimento. Assim, discutiu-se ao longo deste documento a importância de estratégias pragmáticas, por exemplo, considerando prioridades e escalonamento nas estratégias de testes. Sob essa perspectiva, destacou-se a escassez de opções para que os desenvolvedores realizem testes locais de acessibilidade, sem uso de dispositivo. Nesse contexto, uma questão relevante que esta pesquisa levanta é a descontinuação da integração do Robolectric com a API AccessibilityChecks. Portanto, a afirmação de que é possível realizar os mesmos testes com o Robolectric que com o Espresso

requer novas validações. Considerando que o Robolectric recentemente lançou uma versão com suporte a gráficos nativos do Android, avanços nesse sentido devem ser explorados.

Ressalta-se ainda que os estudos apresentados neste trabalho contribuem para discussões emergentes, uma vez que se inserem no contexto da recente Norma ABNT NBR 17060 (2022), que trata especificamente da acessibilidade em aplicativos de dispositivos móveis. O quinto capítulo da norma estabelece uma série de requisitos de acessibilidade que convergem com o que foi abordado por este trabalho. Isso enfatiza a necessidade de a indústria de desenvolvimento de software móvel se preparar para atender a esses requisitos e cumprir as normas estabelecidas.

6.2 Limitações e trabalhos futuros

Durante o desenvolvimento desta pesquisa, foi mantida uma preocupação constante em seguir o rigor científico. No entanto, ocorreram limitações decorrentes do escopo da proposta, do tempo para realização de certas atividades e de mudanças tecnológicas, em parte devido à extensão atípica do período de pesquisa, que foi formalmente iniciado em 2019.

Nos últimos anos, foram observadas importantes mudanças no desenvolvimento de aplicativos móveis. O uso de *frameworks* para desenvolvimento *cross-platform* – ou seja, que geram aplicativos nativos para múltiplas plataformas – tem aumentado (DIAS, 2021), com destaque para o Flutter[1] e o React Native.[2] No desenvolvimento Android, a introdução da biblioteca Jetpack, e em especial o Jetpack Compose,[3] têm alterado a maneira como as interfaces de usuário são criadas para aplicativos desses dispositivos. Contudo, essas novas formas de desenvolvimento estavam fora do escopo deste trabalho.

É possível que existam mais possibilidades do que limitações com as novas abordagens. Por exemplo, a forma declarativa como as UIs são criadas com o Jetpack Compose podem facilitar a criação de testes de unidade para verificação de acessibilidade – o que pode ser melhor investigado em trabalhos futuros. No entanto, ferramentas atuais precisarão ser atualizadas. O Android Lint, por exemplo, não é totalmente compatível com as UIs criadas com o Compose.

Outro tópico que esteve fora do escopo deste trabalho foi o de artefatos de *design*. Como o objetivo, desde o início, concentrou-se em encontrar soluções práticas para a etapa de Construção do Software, não foram dedicados esforços para apresentar as soluções de *design* que auxiliam na produção de aplicativos móveis acessíveis. Os desenvolvedores e testadores não estão errados ao afirmar que um *design system* bem definido pode resolver muitos problemas de acessibilidade. No entanto, é importante ressaltar que isso não os isenta das responsabilidades de implementação

[1] <https://flutter.dev/>
[2] <https://reactnative.dev/>
[3] <https://developer.android.com/jetpack/compose>

e testes, como buscou-se demonstrar neste documento. Por isso, foi importante a inclusão dos *designers* entre os eixos de responsabilidades do AALT.

Como exemplo, um artefato do AALT incluído na Figura 12 e detalhado no Apêndice F é a "Ferramenta de Testes de Acessibilidade para *Designers*" (ATTD - *Accessibility Testing Tool for Designers*). Essa ferramenta é um projeto de aplicativo Android, pré-configurado com testes de acessibilidade, que pode ser usado por *designers* para validar a acessibilidade de componentes gerados a partir de ferramentas como o Figma[4] ou o Zeplin.[5] Entretanto, a melhor maneira de disponibilizar essa ferramenta ainda está sendo discutida com pesquisadores da área.

Uma outra limitação deste trabalho diz respeito ao escopo de problemas de acessibilidade abordados. A deficiência visual é atualmente mais abordada pela literatura - o que ficou demonstrado no mapeamento sistemático apresentado na Seção 3.2. Isso se correlaciona com o fato de os problemas enfrentados por esses usuários serem os mais cobertos pelas ferramentas existentes. Concentrar-se neste tipo de deficiência permitiu que este estudo focasse na elaboração de estratégias para organizar tais recursos em soluções para desenvolvedores. E sim, muitos dos requisitos de acessibilidade apresentados na Seção 5.2 compreendem também outros tipos de deficiência. Por exemplo, "R02 - Elementos de interação devem ter um tamanho mínimo de 48x48dp" e "R05 - Elementos de interação devem ter um espaçamento mínimo de 8dp da borda da tela e pelo menos 9dp de espaçamento entre eles" são requisitos importantes para pessoas com deficiência motora, assim como "R10 - Todos os possíveis erros devem ser claramente informados ao usuário" pode contribuir para pessoas com deficiência cognitiva. Ainda assim, faz-se urgente avançar em pesquisas e oferta de soluções abrangendo a ampla gama de necessidades, características e capacidades de usuários.

É importante salientar que o AALT foi concebido como uma proposta aberta, destinada a ser estendida e aprimorada. Contribuições de trabalhos relacionados podem ser incorporadas em futuras pesquisas. Por exemplo, Silva *et al.* (2020) investigaram a relação entre elementos de código e problemas de acessibilidade em aplicativos Android, e sugerem que aplicativos que adotam elementos de código de acessibilidade tendem a ter menos problemas de acessibilidade. Silva (2020), em seu "guia de automação de testes de acessibilidade para deficiências visuais", fornece orientações para a implementação de testes de acessibilidade identificados como automatizáveis. Com esse conteúdo, é possível aperfeiçoar o conjunto de ferramentas do AALT, ampliando a abrangência e a precisão dos testes de acessibilidade.

Por fim, deve-se considerar uma validação da aplicação do AALT em um projeto voltado ao desenvolvimento de um aplicativo Android nativo.

[4] <https://www.figma.com/>
[5] <https://zeplin.io/home/>

6.3 Considerações finais

Foi notável, na condução das pesquisas para este trabalho de mestrado, o fato de os problemas de acessibilidade prevalentes nos aplicativos mais populares serem justamente aqueles mais abordados por recomendações, mais fáceis de serem identificados por ferramentas, e ainda concentrarem-se em elementos de UI comuns.

Esse fato remete às perguntas iniciais que estes pesquisadores fizeram entre si, antes mesmo das definições do problema e do objetivo de pesquisa: "se existem recursos, por que os aplicativos móveis ainda apresentam tantos problemas de acessibilidade?"

Ao longo deste trabalho, foram explorados diferentes caminhos e hipóteses, principalmente sob uma perspectiva técnica. Embora tenham sido apresentadas algumas propostas e estratégias, a sensação ao final deste percurso é que as respostas talvez estejam mais relacionadas a questões políticas, organizacionais e culturais. É importante reconhecer que a acessibilidade em aplicativos móveis vai além das soluções técnicas e requer uma mudança de mentalidade, um engajamento das partes interessadas e a criação de políticas públicas adequadas.

Salienta-se aqui que a inclusão de recursos de acessibilidade em soluções computacionais não é apenas uma demonstração de empatia, mas também um requisito legal no Brasil. Assim, os profissionais que desenvolvem tecnologias precisam levar a implementação desse requisito a sério. A acessibilidade deve deixar de ser tratada como "*nice to have*". É preciso um esforço conjunto da indústria e da academia para, não só discutir sobre o tema e propor soluções pontuais, mas realizar ações práticas como incentivar disciplinas específicas de acessibilidade nas grades dos cursos da Computação e áreas afins, além de propor um ferramental que apoie os profissionais envolvidos, oferecendo formas mais práticas de aplicação das técnicas. Além disso, é necessário que o poder público esteja envolvido, promovendo agendas positivas relacionadas ao tema, criando programas de incentivo e realizando fiscalização efetiva.

REFERÊNCIAS

ABNT NBR 17060. **Acessibilidade em aplicativos de dispositivos móveis - Requisitos**. [S.l.], 2022. Citado na página 121.

ACOSTA-VARGAS, P.; SALVADOR-ULLAURI, L.; JADáN-GUERRERO, J.; GUEVARA, C.; SANCHEZ-GORDON, S.; CALLE-JIMENEZ, T.; LARA-ALVAREZ, P.; MEDINA, A.; NUNES, I. L. Accessibility assessment in mobile applications for android. **Advances in Intelligent Systems and Computing**, v. 959, p. 279–288, 2020. ISSN 21945365. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85067362341&doi=10.1007%2F978-3-030-20040-4_25&partnerID=40&md5=643772db0bc500fc6c53a734b18ef346>. Citado nas páginas 60 e 61.

AFB. **Screen Reading Technology and Refreshable Braille Displays**. 2017. Disponível em: <http://www.afb.org/info/living-with-vision-loss/using-technology/assistive-technology-videos/screen-reading-technology/1235>. Citado na página 44.

AHMAD, A.; LI, K.; FENG, C.; ASIM, S. M.; YOUSIF, A.; GE, S. An empirical study of investigating mobile applications development challenges. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 6, p. 17711–17728, 3 2018. ISSN 21693536. Disponível em: <https://ieeexplore.ieee.org/document/8326707>. Citado nas páginas 30 e 38.

ALSHAYBAN, A.; AHMED, I.; MALEK, S. Accessibility issues in android apps: State of affairs, sentiments, and ways forward. In: **Proc of the 42nd Intl Conf on Software Engineering**. New York, NY, USA: ACM, 2020. (ICSE '20), p. 1323–1334. ISBN 9781450371216. Disponível em: <https://doi.org/10.1145/3377811.3380392>. Citado nas páginas 29, 31, 60, 61, 63, 86, 89, 94 e 95.

ANICHE, M. **Real World Test-Driven Development**. São Paulo: Casa do Código, 2014. ISBN 9788566250572. Citado nas páginas 30, 40 e 41.

ANTONELLI, H. L.; RODRIGUES, S. S.; WATANABE, W. M.; FORTES, R. P. de M. A survey on accessibility awareness of brazilian web developers. In: **Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion**. New York, NY, USA: Association for Computing Machinery, 2018. (DSAI '18), p. 71–79. ISBN 9781450364676. Disponível em: <https://doi.org/10.1145/3218585.3218598>. Citado nas páginas 46, 77, 79 e 113.

APPLE. **Accessibility - Apple Developer**. 2022. Disponível em: <https://developer.apple.com/accessibility/>. Citado na página 47.

ARTEAGA, J. M.; RIVERA, D. I. P. A process model to develop educational applications for children with dyslexia. In: . IEEE, 2018. p. 79–87. ISBN 978-1-5386-6577-0. Disponível em: <https://ieeexplore.ieee.org/document/8645896/>. Citado nas páginas 73 e 74.

BALLANTYNE, M.; JHA, A.; JACOBSEN, A.; HAWKER, J. S.; EL-GLALY, Y. N. Study of accessibility guidelines of mobile applications. In: **Proc of the 17th Intl Conf on Mobile**

and Ubiquitous Multimedia. New York, NY, USA: ACM, 2018. (MUM 2018), p. 305–315. ISBN 9781450365949. Disponível em: <https://doi.org/10.1145/3282894.3282921>. Citado nas páginas 60 e 61.

BATISTA, M. H. d. S. **Uma abordagem para verificação de acessibilidade e usabilidade em aplicativos móveis**. Tese (Mestrado em Ciências de Computação e Matemática Computacional) — Universidade de São Paulo, São Carlos, jan. 2018. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-07012019-091622/>. Citado nas páginas 49 e 50.

BBC. **Accessibility - BBC**. 2022. Disponível em: <https://www.bbc.co.uk/accessibility/>. Citado nas páginas 46, 47, 81, 101, 102 e 159.

BECK. **Test Driven Development: By Example**. USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0321146530. Citado na página 41.

BECK, K.; ANDRES, C. **Extreme Programming Explained: Embrace Change (2nd Edition)**. USA: Addison-Wesley Professional, 2004. ISBN 0321278658. Citado na página 41.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R. *et al.* **The Agile Manifesto**: Manifesto for agile software development. Agile Alliance, 2001. Disponível em: <http://www.agilemanifesto.org>. Citado nas páginas 41 e 95.

BENESTY, J.; CHEN, J.; HUANG, Y.; COHEN, I. Pearson correlation coefficient. In: ____. [S.l.: s.n.], 2009. v. 2. Citado na página 109.

BI, T.; XIA, X.; LO, D.; GRUNDY, J.; ZIMMERMANN, T.; FORD, D. Accessibility in software practice: A practitioner's perspective. **Trans. on Software Engineering and Methodology**, ACM, v. 31, p. 1–26, 10 2022. ISSN 1049-331X. Disponível em: <https://dl.acm.org/doi/10.1145/3503508>. Citado nas páginas 30, 32, 46, 56, 77, 78, 79, 94, 95 e 101.

BOSNIC, S.; PAPP, I.; NOVAK, S. The development of hybrid mobile applications with apache cordova. **24th Telecommunications Forum, TELFOR 2016**, Institute of Electrical and Electronics Engineers Inc., 1 2017. Citado nas páginas 29 e 39.

BOURAOUI, A.; GHARBI, I. Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. **Science of Computer Programming**, v. 172, p. 63–101, 2019. ISSN 01676423. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0167642318304301>. Citado nas páginas 44 e 75.

BRASIL. **Lei nº 13.146, de 6 de julho de 2015 (Lei Brasileira de Inclusão da Pessoa com Deficiência / Estatuto da Pessoa com Deficiência)**. 2015. Disponível em: <http://www2.camara.leg.br/legin/fed/lei/2015/lei-13146-6-julho-2015-781174-normaatualizada-pl.pdf>. Citado nas páginas 29, 44 e 59.

BROOKE, J. Sus - a quick and dirty usability scale. **Usability evaluation in industry**, London: Taylor & Francis, v. 189, p. 194, 1996. Citado nas páginas 107 e 109.

CALVO, R.; IGLESIAS, A.; MORENO, L. User-centered requirement engineering for accessible chats in m-learning. **Journal of Universal Computer Science**, v. 20, p. 964–985, 2014. ISSN 09486968. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84908113873&partnerID=40&md5=942325c1a5e78bebdf2e20f6cf63f9b1https://www.scopus.com/inward/record.uri?eid=2-s2.0-84908113873%5C&partnerID=40%5C&md5=942325c1a5e78bebdf2e20f6cf63f9b1>. Citado na página 75.

CARVALHO, L. P.; FREIRE, A. P. Native or web-hybrid apps? an analysis of the adequacy for accessibility of android interface components used with screen readers. **Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems**, ACM, 2017. Disponível em: <https://doi.org/10.1145/3160504.3160511>. Citado nas páginas 30 e 38.

CARVALHO, L. P.; PERUZZA, B. P. M.; SANTOS, F.; FERREIRA, L. P.; FREIRE, A. P. Accessible smart cities?: Inspecting the accessibility of brazilian municipalities' mobile applications. In: **Proc of the 15th Brazilian Symposium on Human Factors in Computing Systems Human Factors in Computing Systems**. New York, NY, USA: ACM, 2016. p. 17:1–17:10. ISBN 978-1-4503-5235-2. Disponível em: <http://doi.acm.org/10.1145/3033701.3033718>. Citado na página 29.

CARVALHO, M. C. N.; DIAS, F. S.; REIS, A. G. S.; FREIRE, A. P. Accessibility and usability problems encountered on websites and applications in mobile devices by blind and normal-vision users. In: **Proc of the 33rd Annual Symposium on Applied Computing**. New York, NY, USA: ACM, 2018. p. 2022–2029. ISBN 978-1-4503-5191-1. Disponível em: <http://doi.acm.org/10.1145/3167132.3167349>. Citado nas páginas 60 e 61.

CLEGG-VINELL, R.; BAILEY, C.; GKATZIDOU, V. Investigating the appropriateness and relevance of mobile web accessibility guidelines. In: . Association for Computing Machinery, 2014. p. 1–4. ISBN 978-1-4503-2651-3. Disponível em: <http://dl.acm.org/citation.cfm?doid=2596695.2596717>. Citado na página 29.

CONSORTIUM, I. G. L. **IMS Guidelines for Developing Accessible Learning Applications | IMS Global Learning Consortium**. 2018. Disponível em: <https://www.imsglobal.org/accessibility/accessiblevers/index.html>. Citado na página 75.

CORONEL, N. O. M.; HERNANDEZ, J. O.; SANCHEZ, S. S.; SERNA, J. G. G. Integration of Usability Into the Development of Mobile Computing Environments Applied to Contextual Location Based Services (LBS). In: **2011 IEEE Electronics, Robotics and Automotive Mechanics Conference**. Cuernavaca, Morelos, Mexico: IEEE, 2011. p. 409–414. ISBN 978-1-4577-1879-3. Disponível em: <http://ieeexplore.ieee.org/document/6125865/>. Citado na página 64.

CRISPIN, L.; GREGORY, J. **Agile testing: A practical guide for testers and agile teams**. Boston, MA: Pearson Education, 2009. ISBN 978-0-321-53446-0. Citado nas páginas 30, 31, 39, 40, 42, 43 e 94.

CRUZ, L.; ABREU, R.; LO, D. To the attention of mobile software developers: guess what, test your app! **Empirical Software Engineering 2019 24:4**, Springer, v. 24, p. 2438–2468, 4 2019. ISSN 1573-7616. Disponível em: <https://link.springer.com/article/10.1007/s10664-019-09701-0>. Citado na página 30.

DAMACENO, R. J. P.; BRAGA, J. C.; MENA-CHALCO, J. P. Mobile device accessibility for the visually impaired: problems mapping and recommendations. **Universal Access in the Information Society**, v. 17, p. 421–435, 2018. ISSN 1615-5297. Disponível em: <https://doi.org/10.1007/s10209-017-0540-1>. Citado na página 61.

DARDAILLER, D. **WAI early days**. 2009. Disponível em: <https://www.w3.org/WAI/history>. Citado nas páginas 44 e 67.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao teste de software**. Rio de Janeiro: Elsevier, 2016. ISBN 978-85-352-8352-5. Citado nas páginas 30, 39, 41, 42, 43 e 54.

DESIGN, M. **Material Design**. 2019. Disponível em: <https://material.io/>. Citado na página 50.

DIAS, A. L. **Um processo para sistemas web com foco em acessibilidade e usabilidade**. Tese (Doutorado) — Universidade de São Paulo, São Carlos, aug 2014. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-18032015-160137/>. Citado na página 64.

DIAS, F.; DUARTE, L.; FORTES, R. Accessmdd: An mdd approach for generating accessible mobile applications. **Proc of the 39th ACM Intl Conf on the Design of Communication - SIGDOC 2021**, ACM, v. 11, p. 85–95, 2021. Disponível em: <https://doi.org/10.1145/3472714.3473904>. Citado nas páginas 60, 86, 101, 116, 120 e 157.

DIAS, F. S. **AccessMDD: uma abordagem MDD para geração de aplicativos móveis acessíveis**. Tese (Doutorado) — Universidade de São Paulo, 6 2021. Disponível em: <https://www.teses.usp.br/teses/disponiveis/55/55134/tde-17082021-135910/>. Citado nas páginas 61, 62, 89, 102 e 121.

DIX, A.; FINLAY, J.; ABOWD, G. D.; BEALE, R. **Human Computer Interaction**. 3. ed. Harlow, England: Pearson Prentice Hall, 2004. ISBN 978-0-13-046109-4. Citado na página 48.

DUVALL, P.; MATYAS, S.; GLOVER, A. **Continuous integration: improving software quality and reducing risk**. [S.l.]: Addison-Wesley Professional, 2007. Citado na página 42.

DYBå, T.; DINGSøYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: An experience report. In: . [S.l.: s.n.], 2007. ISBN 0769528864. Citado na página 67.

ECKHARDT, J.; VOGELSANG, A.; FERNÁNDEZ, D. M. Are non-functional requirements really non-functional? an investigation of non-functional requirements in practice. **Proceedings - International Conference on Software Engineering**, IEEE Computer Society, v. 14-22-May-2016, p. 832–842, 5 2016. ISSN 02705257. Disponível em: <https://dl.acm.org/doi/10.1145/2884781.2884788>. Citado na página 95.

ELER, M. M.; ROJAS, J. M.; GE, Y.; FRASER, G. Automated accessibility testing of mobile apps. In: **2018 IEEE 11th Intl Conf on Software Testing, Verification and Validation (ICST)**. Västerås, Sweden: IEEE Inc., 2018. p. 116–126. ISBN 9781538650127. Citado nas páginas 31, 46, 55, 79, 80, 83, 85, 86, 87 e 89.

FAJARDO-FLORES, S. B.; GAYTáN-LUGO, L. S.; SANTANA-MANCILLA, P. C.; RODRÍGUEZ-ORTIZ, M. A. Mobile accessibility for people with combined visual and motor impairment: A case study. In: **Proceedings of the 8th Latin American Conference on Human-Computer Interaction**. New York, NY, USA: Association for Computing Machinery, 2017. (CLIHC '17). ISBN 9781450354295. Disponível em: <https://doi.org/10.1145/3151470.3151476>. Citado na página 60.

FEINER, J.; KRAINZ, E.; ANDREWS, K. A new approach to visualise accessibility problems of mobile apps in source code. In: . [s.n.], 2018. v. 2, p. 519–526. ISBN 9789897582981. Cited By 1. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85047731236&partnerID=40&md5=d47522bb6414f6702b939ae8da49175c>. Citado na página 74.

FELIZARDO, K. R.; NAKAGAWA, E. Y.; FABBRI, S. C. P. F.; FERRARI, F. C. **Revisão Sistemática da Literatura em Engenharia de Software**. [S.l.: s.n.], 2017. ISSN 01676687. ISBN 8535286411. Citado nas páginas 65, 67 e 69.

FERREIRA, F.; ALMEIDA, N.; ROSA, A. F.; OLIVEIRA, A.; CASIMIRO, J.; SILVA, S.; TEIXEIRA, A. Elderly centered design for interaction – the case of the s4s medication assistant. **Procedia Computer Science**, v. 27, p. 398–408, 2014. ISSN 18770509. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1877050914000465>. Citado na página 75.

FOLMER, E. **Interaction Design Patterns**. Interaction Design Foundation, 2019. Disponível em: <https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/interaction-design-patterns>. Citado na página 49.

FREIRE, A. P.; RUSSO, C. M.; FORTES, R. P. M. A survey on the accessibility awareness of people involved in web development projects in brazil. In: **Proceedings of the 2008 International Cross-Disciplinary Conference on Web Accessibility (W4A)**. New York, NY, USA: Association for Computing Machinery, 2008. (W4A '08), p. 87–96. ISBN 9781605581538. Disponível em: <https://doi.org/10.1145/1368044.1368064>. Citado na página 77.

GAMMA, E.; VLISSIDES, J.; HELM, R.; JOHNSON, R. **Design patterns: elements of reusable object-oriented software**. [S.l.]: Addison-Wesley, 1995. ISBN 978-0-201-63361-0. Citado na página 49.

GARCéS, L.; AMPATZOGLOU, A.; AVGERIOU, P.; NAKAGAWA, E. Y. Quality attributes and quality models for ambient assisted living software systems: {A} systematic mapping. **Information and Software Technology**, v. 82, p. 121–138, 2017. ISSN 09505849. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0950584916302932>. Citado na página 67.

GEMOU, M.; COLOMER, J. B. M.; CABRERA-UMPIERREZ, M. F.; RIOS, S. de los; ARREDONDO, M. T.; BEKIARIS, E. Validation of Toolkits for developing third-generation Android accessible mobile applications. **Universal Access in the Information Society**, v. 15, n. 1, p. 101–127, mar. 2016. ISSN 1615-5297. Disponível em: <https://doi.org/10.1007/s10209-014-0377-9>. Citado nas páginas 79 e 80.

GIL, A. C. **Métodos e técnicas de pesquisa social**. São Paulo: Atlas, 2008. v. 6. ed. ISBN 85-224-3169-8. Citado na página 92.

GITHUB. **GitHub**. 2023. Disponível em: <https://github.com>. Citado nas páginas 115 e 116.

GIUFFRIDA, R.; DITTRICH, Y. **Empirical studies on the use of social software in global software development-A systematic mapping study**. [S.l.]: Elsevier B.V., 2013. 1143-1164 p. Citado na página 66.

GOMES, B.; RIOS, J.; RODRIGUES, K. R. Challenges for the implementation of accessible web and mobile systems. In: SPRINGER. **Software Ecosystems, Sustainability and Human Values in the Social Web: 8th Workshop of Human-Computer Interaction Aspects to the Social Web, WAIHCWS 2017, Joinville, Brazil, October 23, 2017 and 9th Workshop, WAIHCWS 2018, Belém, Brazil, October 22, 2018, Revised Selected Papers 8**. [S.l.], 2020. p. 138–158. Citado nas páginas 30, 32 e 56.

GOMES, F. T.; SALGADO, A. de L.; DUARTE, L. M. C.; SANTOS, F. S.; FORTES, R. P. M. Um simulador visual de leitor de telas para aux{í}lio à interpretação de questões de acessibilidade por avaliadores videntes. **Revista de Sistemas e Computação-RSC**, v. 8, p. 114–134, 2018. Disponível em: <https://revistas.unifacs.br/index.php/rsc/article/view/5272>. Citado na página 29.

GOOGLE. **Accessibility Test Framework for Android**. [S.l.]: GitHub, 2022. <https://github.com/google/Accessibility-Test-Framework-for-Android>. Citado na página 51.

____. **Accessibility - Material Design**. 2023. Disponível em: <https://m3.material.io/foundations/accessible-design/overview>. Citado nas páginas 48 e 74.

____. **Conceitos básicos de testes**. 2023. Disponível em: <https://developer.android.com/training/testing/fundamentals?hl=pt-br>. Citado nas páginas 31, 40, 41 e 87.

____. **Developer Guides**. 2023. Disponível em: <https://developer.android.com/guide/>. Citado nas páginas 31 e 87.

____. **Espresso**. 2023. Disponível em: <https://developer.android.com/training/testing/espresso?hl=pt-br>. Citado na página 54.

____. **Roboletric**. [S.l.]: GitHub, 2023. <https://github.com/robolectric/robolectric>. Citado na página 55.

____. **Testar a acessibilidade do seu app**. 2023. Disponível em: <https://developer.android.com/guide/topics/ui/accessibility/testing?hl=pt-br>. Citado na página 50.

____. **Testar a UI de vários apps**. 2023. Disponível em: <https://developer.android.com/training/testing/ui-testing/uiautomator-testing?hl=pt-br>. Citado nas páginas 53 e 54.

____. **Tornar os apps mais acessíveis**. 2023. Disponível em: <https://developer.android.com/guide/topics/ui/accessibility/apps?hl=pt-br>. Citado nas páginas 46, 47 e 51.

HAO, S.; LIU, B.; NATH, S.; HALFOND, W. G.; GOVINDAN, R. Puma: Programmable ui-automation for large-scale dynamic analysis of mobile apps. In: **Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services**. New York, NY, USA: Association for Computing Machinery, 2014. (MobiSys '14), p. 204–217. ISBN 9781450327930. Disponível em: <https://doi.org/10.1145/2594368.2594390>. Citado nas páginas 79 e 80.

HARRISON, R.; FLOOD, D.; DUCE, D. Usability of mobile applications: literature review and rationale for a new usability model. **Journal of Interaction Science**, v. 1, p. 1, 2013. ISSN 2194-0827. Disponível em: <http://journalofinteractionscience.springeropen.com/articles/10.1186/2194-0827-1-1>. Citado na página 38.

HENRY, S. L.; ABOU-ZAHRA, S.; BREWER, J. The role of accessibility in a universal web. In: . ACM Press, 2014. p. 1–4. ISBN 978-1-4503-2651-3. Disponível em: <http://dl.acm.org/citation.cfm?doid=2596695.2596719>. Citado na página 29.

HESS, S.; KIEFER, F.; CARBON, R.; MAIER, A. nconcappt - a method for the conception of mobile business applications. **Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering**, v. 110 LNICST, p. 1–20, 2013. ISSN 18678211. Cited By 7. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84874811498&doi=10.1007%2F978-3-642-36632-1_1&partnerID=40&md5=14c3491311b517db96b05444d8b38c69>. Citado nas páginas 30 e 76.

HOLZINGER, A.; SAMMER, P.; HOFMANN-WELLENHOF, R. Mobile computing in medicine: Designing mobile questionnaires for elderly and partially sighted people. In: HUTCHISON, D.; KANADE, T.; KITTLER, J.; KLEINBERG, J. M.; MATTERN, F.; MITCHELL, J. C.; NAOR, M.; NIERSTRASZ, O.; RANGAN, C. P.; STEFFEN, B.; SUDAN, M.; TERZOPOULOS, D.; TYGAR, D.; VARDI, M. Y.; WEIKUM, G.; MIESENBERGER, K.; KLAUS, J.; ZAGLER, W. L.; KARSHMER, A. I. (Ed.). Springer Berlin Heidelberg, 2006. v. 4061 LNCS, p. 732–739. ISBN 3540360204. Disponível em: <http://link.springer.com/10.1007/11788713_107>. Citado nas páginas 73, 74 e 75.

IBM. **5 steps of test-driven development**. 2023. Disponível em: <https://developer.ibm.com/articles/5-steps-of-test-driven-development/>. Citado na página 42.

IDRI, A.; AMAZAL, F. A.; ABRAN, A. **Analogy-based software development effort estimation: A systematic mapping and review**. [S.l.]: Elsevier B.V., 2015. 206-230 p. Citado na página 66.

INAL, Y.; RıZVANOğLU, K.; YESILADA, Y. Web accessibility in Turkey: awareness, understanding and practices of user experience professionals. **Universal Access in the Information Society**, v. 18, n. 2, p. 387–398, jun. 2019. ISSN 1615-5297. Disponível em: <https://doi.org/10.1007/s10209-017-0603-3>. Citado nas páginas 77, 79 e 113.

ISO 9241-11. **Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts**. [S.l.], 2018. Citado na página 43.

ISO 9241-171. **ISO 9241-171:2008 Ergonomics of human-system interaction-Part 171 : Guidance on software accessibility**. [S.l.], 2008. Citado na página 64.

ISO 9241-210. **ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems**. [S.l.], 2019. Citado na página 48.

JAEGER, P. T.; XIE, B. Developing online community accessibility guidelines for persons with disabilities and older adults. **Journal of Disability Policy Studies**, 2009. ISSN 10442073. Citado na página 75.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele University,UK and National ICT Australia**, 2004. ISSN 13537776. Citado na página 69.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.], 2007. Http://www.dur.ac.uk/ebse/resources/Systematic-reviews-5-8.pdf. Citado na página 65.

____. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.], 2007. Disponível em: <http://www.dur.ac.uk/ebse/resources/Systematic-reviews-5-8.pdf>. Citado na página 67.

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, O. P. The value of mapping studies – a participant-observer case study. In: . [S.l.]: BCS Learning and Development, 2010. Citado na página 65.

KOCHHAR, P. S.; THUNG, F.; NAGAPPAN, N.; ZIMMERMANN, T.; LO, D. Understanding the test automation culture of app developers. In: **8th Intl Conf on Software Testing, Verification and Validation (ICST)**. Graz, Austria: IEEE, 2015. p. 1–10. ISBN 978-1-4799-7125-1. Citado na página 87.

KRAINZ, E.; FEINER, J.; FRUHMANN, M. Accelerated development for accessible apps – model driven development of transportation apps for visually impaired people. In: BOGDAN, C.; GULLIKSEN, J.; SAUER, S.; FORBRIG, P.; WINCKLER, M.; JOHNSON, C.; PALANQUE, P.; BERNHAUPT, R.; KIS, F. (Ed.). Springer International Publishing, 2016. v. 9856, p. 374–381. ISBN 978-3-319-44901-2 978-3-319-44902-9. Disponível em: <http://link.springer.com/10.1007/978-3-319-44902-9_25>. Citado na página 74.

KRAJNC, E.; FEINER, J.; SCHMIDT, S. User centered interaction design for mobile applications focused on visually impaired and blind people. In: LEITNER, G.; HITZ, M.; HOLZINGER, A. (Ed.). **{HCI} in {Work} and {Learning}, {Life} and {Leisure}**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. v. 6389, p. 195–202. ISBN 978-3-642-16606-8 978-3-642-16607-5. Disponível em: <http://link.springer.com/10.1007/978-3-642-16607-5_12>. Citado nas páginas 75 e 76.

LAZAR, J. Web accessibility policy and law. In: ____. **Web Accessibility: A Foundation for Research**. London: Springer London, 2019. p. 247–261. ISBN 978-1-4471-7440-0. Disponível em: <https://doi.org/10.1007/978-1-4471-7440-0_14>. Citado na página 59.

LAZAR, J.; FENG, J. H.; HOCHHEISER, H. **Research Methods in Human-Computer Interaction**. [S.l.: s.n.], 2017. Citado na página 92.

LEITE, M. V. R.; SCATALON, L. P.; FREIRE, A. P.; ELER, M. M. Accessibility in the mobile development industry in brazil: Awareness, knowledge, adoption, motivations and barriers. **Journal of Systems and Software**, Elsevier, v. 177, p. 110942, 7 2021. ISSN 0164-1212. Citado nas páginas 30, 32, 56, 59, 77, 78, 79, 94, 101 e 113.

LINARES-VÁSQUEZ, M.; BERNAL-CARDENAS, C.; MORAN, K.; POSHYVANYK, D. How do developers test android applications? In: **2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. [S.l.: s.n.], 2017. p. 613–622. Citado na página 55.

LOCKWOOD, C.; MUNN, Z.; PORRITT, K. Qualitative research synthesis: Methodological guidance for systematic reviewers utilizing meta-aggregation. **International Journal of Evidence-Based Healthcare**, v. 13, p. 179–187, 9 2015. ISSN 17441609. Disponível em: <http://content.wkhealth.com/linkback/openurl?sid=WKPTLP:landingpage&an=01787381-201509000-00010>. Citado na página 65.

LOSADA, B.; FERNáNDEZ-CASTRO, I.; LóPEZ-GIL, J.-M.; URRETAVIZCAYA, M. Applying usability engineering in intermod agile development methodology. a case study in a mobile application. **JUCS - Journal of Universal Computer Science**, v. 19, n. 8, p. 1046–1065, 2013. ISSN 0948695X. Place: Austria Publisher: Verlag der Technischen Universität Graz. Disponível em: <https://lib.jucs.org/article/23400>. Citado nas páginas 30 e 76.

MAIA, L. S.; TURINE, M. A. S.; PAIVA, D. M. B. **Um Modelo para o Desenvolvimento de Aplicações Web Acessíveis 1**. [S.l.], 2010. 235-242 p. Disponível em: <www.dasilva.org.br>. Citado na página 64.

MANKOFF, J.; HOFMANN, M.; CHEN, X. A.; HUDSON, S. E.; HURST, A.; KIM, J. Consumer-grade fabrication and its potential to revolutionize accessibility. **Communications of the ACM**, Association for Computing Machinery, v. 62, p. 64–75, 2019. ISSN 00010782. Disponível em: <http://dl.acm.org/citation.cfm?doid=3363418.3339824>. Citado na página 29.

MIRANDA, D.; ARAUJO, J. Studying industry practices of accessibility requirements in agile development. In: . Association for Computing Machinery, 2022. v. 10, p. 1309–1317. ISBN 9781450387132. Disponível em: <https://doi.org/10.1145/3477314.3507041>. Citado nas páginas 78, 79, 94, 95 e 113.

NERIS, V. P. d. A. Estudo e proposta de um framework para o design de interfaces de usuario ajustaveis. [s.n.], 6 2010. Disponível em: <http://acervus.unicamp.br/index.asp?codigo_sophia=768432>. Citado na página 99.

NIELSEN, J. **Usability engineering**. Academic Press, 1993. 358 p. ISBN 0125184050. Disponível em: <https://dl.acm.org/citation.cfm?id=529793>. Citado na página 43.

NORMAN, D. A. **The Design of Everyday Things**. 2. ed. [S.l.]: Basic Books, 2013. 2002 Reprint Paperback ISBN 978-0-465-06710-7. ISBN 978-0-465-05065-9. Citado na página 48.

NUDELMAN, G. **Android Design Patterns: Interaction Design Solutions for Developers**. [S.l.]: Wiley, 2013. v. 1. 456 p. OCLC: ocn788247284. ISSN 1098-6596. ISBN 978-1-118-39415-1. Citado na página 50.

NUNKESSER, R. Beyond web/native/hybrid: A new taxonomy for mobile app development. **Proceedings - International Conference on Software Engineering**, IEEE Computer Society, p. 214–218, 5 2018. ISSN 02705257. Disponível em: <https://doi.org/10.1145/3197231.3197260>. Citado na página 38.

OLIVEIRA, A. D. A.; ELER, M. M. Strategies and challenges on the accessibility and interoperability of e-government web portals: A case study on brazilian federal universities. In: **2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)**. [S.l.: s.n.], 2017. v. 1, p. 737–742. Citado nas páginas 77 e 94.

OLIVEIRA, A. D. A.; SANTOS, P. S. H. D.; JúNIOR, W. E. M.; ALJEDAANI, W. M.; ELER, D. M.; ELER, M. M. Analyzing accessibility reviews associated with visual disabilities or eye conditions. In: **Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2023. (CHI '23). ISBN 9781450394215. Disponível em: <https://doi.org/10.1145/3544548.3581315>. Citado nas páginas 30, 31, 60, 63, 86, 89, 94 e 95.

OLIVEIRA, A. F. B. A. de; FILGUEIRAS, L. V. L. Developer assistance tools for creating native mobile applications accessible to visually impaired people: A systematic review. In: **Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2018. (IHC 2018). ISBN 9781450366014. Disponível em: <https://doi.org/10.1145/3274192.3274208>. Citado nas páginas 79 e 80.

____. Accessibilint: A tool for early accessibility verification for android native applications. In: . Association for Computing Machinery, 2019. ISBN 9781450369718. Disponível em: <https://doi.org/10.1145/3357155.3360474>. Citado nas páginas 79, 80, 82, 83, 95, 101 e 102.

OLIVEIRA, C. de; FIORAVANTI, M.; FORTES, R.; BARBOSA, E. Accessibility in mobile applications for elderly users: A systematic mapping. In: . [S.l.: s.n.], 2019. v. 2018-Octob. ISBN 9781538611739. ISSN 15394565. Citado nas páginas 30 e 37.

OLIVEIRA, R.; FRANÇA, C. Agile practices and motivation: A quantitative study with brazilian software developers. In: **Proc of the Evaluation and Assessment on Software Engineering**.

New York, NY, USA: ACM, 2019. (EASE '19), p. 365–368. ISBN 9781450371452. Disponível em: <https://doi.org/10.1145/3319008.3319714>. Citado nas páginas 30, 31 e 102.

PAI, M.; MCCULLOCH, M.; GORMAN, J. D.; PAI, N.; ENANORIA, W.; KENNEDY, G.; THARYAN, P.; COLFORD, J. M. **Systematic reviews and meta-analyses: An illustrated, step-by-step guide**. 2004. 86-95 p. Citado na página 65.

PAIVA, D. M. B.; FREIRE, A. P.; FORTES, R. P. M. Accessibility and software engineering processes: A systematic literature review. **Journal of Systems and Software**, Elsevier, v. 171, p. 1–17, 2021. Citado na página 29.

PANDEY, Y.; LEE, J.; BANDA, D. R.; GRIFFIN-SHIRLEY, N.; NGUYEN, T.; OTHUON, V. A survey of mobile app use among university students with visual impairment in india. **British Journal of Visual Impairment**, 2022. Disponível em: <https://doi.org/10.1177/02646196211067358>. Citado na página 60.

PARK, E.; HAN, S.; BAE, H.; KIM, R.; LEE, S.; LIM, D.; LIM, H. Development of automatic evaluation tool for mobile accessibility for android application. In: **2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS)**. [S.l.: s.n.], 2019. p. 1–6. Citado nas páginas 79, 80, 82 e 83.

PARK, K.; GOH, T.; SO, H.-J. Toward accessible mobile application design: Developing mobile application accessibility guidelines for people with visual impairment. In: **Proceedings of HCI Korea**. Seoul, KOR: Hanbit Media, Inc., 2014. (HCIK '15), p. 31–38. ISBN 9788968487521. Citado na página 60.

PEISCHL, B.; FERK, M.; HOLZINGER, A. The fine art of user-centered software development. **Software Quality Journal**, v. 23, p. 509–536, 2015. ISSN 15731367. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84928708378&doi=10.1007\http://link.springer.com/10.1007/s11219-014-9239-1>. Citado na página 64.

PETRIE, H.; BEVAN, N. The Evaluation of Accessibility, Usability, and User Experience. In: STEPHANIDIS, C. (Ed.). **The Universal Access Handbook**. CRC Press, 2009. v. 20091047, p. 1–16. ISBN 978-1-4200-6499-5. Disponível em: <http://www.crcpress.com/product/isbn/9780805862805>. Citado nas páginas 43, 48 e 49.

PETRIE, H.; KHEIR, O. The relationship between accessibility and usability of websites. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2007. (CHI '07), p. 397–406. ISBN 9781595935939. Disponível em: <https://doi.org/10.1145/1240624.1240688>. Citado na página 43.

PETRIE, H.; SAVVA, A.; POWER, C. Towards a unified definition of web accessibility. In: . ACM, 2015. p. 35:1–35:13. ISBN 978-1-4503-3342-9. Disponível em: <http://doi.acm.org/10.1145/2745555.2746653>. Citado nas páginas 29 e 44.

PINHEIRO, V.; MARQUES, A. B. Accessibility-oriented design with a focus on autism aspects: Designing a mobile application for autistic children's daily routine. In: **Proceedings of the XIX Brazilian Symposium on Software Quality**. New York, NY, USA: Association for Computing Machinery, 2021. (SBQS '20). ISBN 9781450389235. Disponível em: <https://doi.org/10.1145/3439961.3439988>. Citado na página 48.

PRESSMAN, R. S. **Engenharia de Software: Uma Abordagem Profissional**. 7. ed. Porto Alegre: AMGH, 2011. 780 p. ISBN 978-85-63308-33-7. Citado nas páginas 41, 42 e 95.

REITHINGER, N.; RUSS, A.; SCHUMACHER, K. User-centered interaction design of a mobile learning platform for the generation 60 +. In: . ACM, 2015. p. 924–927. ISBN 9781450336536. Disponível em: <http://doi.acm.org/10.1145/2786567.2794305>. Citado na página 75.

RIEGER, C.; LUCRéDIO, D.; FORTES, R. P. M.; KUCHEN, H.; DIAS, F.; DUARTE, L. A model-driven approach to cross-platform development of accessible business apps. In: **Proc of the 35th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2020. p. 984–993. ISBN 9781450368667. Disponível em: <https://doi.org/10.1145/3341105.3375765>. Citado na página 29.

SAKAMOTO, S. G.; SILVA, L. F. da; MIRANDA, L. C. de. Identificando barreiras de acessibilidade web em dispositivos móveis: resultados de um estudo de caso orientado pela engenharia de requisitos. In: . Brazilian Computer Society, 2012. p. 23–32. ISBN 978-85-7669-262-1. Disponível em: <http://dl.acm.org/citation.cfm?id=2393536.2393540>. Citado nas páginas 37 e 75.

SALEHNAMADI, N.; ALSHAYBAN, A.; LIN, J.-W.; AHMED, I.; BRANHAM, S.; MALEK, S. Latte: Use-case and assistive-service driven automated accessibility testing framework for android. In: **Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2021. (CHI '21). ISBN 9781450380966. Disponível em: <https://doi.org/10.1145/3411764.3445455>. Citado nas páginas 79, 80, 82, 84 e 87.

SALEHNAMADI, N.; HE, Z.; MALEK, S. Assistive-technology aided manual accessibility testing in mobile apps, powered by record-and-replay. In: **Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2023. (CHI '23). ISBN 9781450394215. Disponível em: <https://doi.org/10.1145/3544548.3580679>. Citado nas páginas 60 e 63.

SALMAN, F. A.; DERAMAN, A. A model for incorporating suitable methods of usability evaluation into agile software development. **Bulletin of Electrical Engineering and Informatics**, Institute of Advanced Engineering and Science, v. 11, p. 3433–3440, 12 2022. ISSN 2302-9285. Disponível em: <https://beei.org/index.php/EEI/article/view/4277>. Citado na página 30.

SHNEIDERMAN, B. Universal usability. **Commun. ACM**, Association for Computing Machinery, v. 43, p. 84–91, 2000. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/332833.332843>. Citado na página 43.

SIDI. **Guia para o Desenvolvimento de Aplicações Móveis Acessíveis**. 2017. Samsung Instituto de Desenvolvimento para Informática (SIDI). Citado nas páginas 46, 47 e 159.

SIEBRA, C.; GOUVEIA, T.; MACEDO, J.; CORREIA, W.; PENHA, M.; ANJOS, M.; FLORENTIN, F.; SILVA, F. Q. B.; SANTOS, A. L. M. Observation based analysis on the use of mobile applications for visually impaired users. In: **Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct**. New York, NY, USA: Association for Computing Machinery, 2016. (MobileHCI '16), p. 807–814. ISBN 9781450344135. Disponível em: <https://doi.org/10.1145/2957265.2961848>. Citado na página 60.

SIEBRA, C.; GOUVEIA, T. B.; MACEDO, J.; SILVA, F. Q. B. D.; SANTOS, A. L. M.; CORREIA, W.; PENHA, M.; FLORENTIN, F.; ANJOS, M. Toward accessibility with usability: Understanding the requirements of impaired uses in the mobile context. In: **Proc of the 11th International Conference on Ubiquitous Information Management and Communication, IMCOM 2017**. New York, NY, USA: ACM, 2017. p. 6:1–6:8. ISBN 9781450348881. Disponível em: <http://doi.acm.org/10.1145/3022227.3022233>. Citado na página 47.

SILVA, C.; ELER, M. M.; FRASER, G. A survey on the tool support for the automatic evaluation of mobile accessibility. In: **Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion**. New York, NY, USA: Association for Computing Machinery, 2018. (DSAI '18), p. 286–293. ISBN 9781450364676. Disponível em: <https://doi.org/10.1145/3218585.3218673>. Citado nas páginas 59, 79, 80 e 83.

SILVA, C. F. P. Um estudo sobre a automação de testes de acessibilidade em dispositivos móveis: uma análise de recomendações para deficiência visual da bbc. Biblioteca Digital de Teses e Dissertações da Universidade de São Paulo, 3 2020. Disponível em: <https://www.teses.usp.br/teses/disponiveis/100/100131/tde-16052020-111716/>. Citado nas páginas 81, 95, 101, 102 e 122.

SILVA, H. N. da; ENDO, A. T.; ELER, M. M.; VERGILIO, S. R.; DURELLI, V. H. S. On the relation between code elements and accessibility issues in android apps. In: **Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing**. New York, NY, USA: Association for Computing Machinery, 2020. (SAST 20), p. 40–49. ISBN 9781450387552. Disponível em: <https://doi.org/10.1145/3425174.3425209>. Citado nas páginas 37 e 122.

SILVA, L. F. da; JUNIOR, P. A. P.; FREIRE, A. P. Mobile user interaction design patterns: A systematic mapping study. **Information**, v. 13, n. 5, 2022. ISSN 2078-2489. Disponível em: <https://www.mdpi.com/2078-2489/13/5/236>. Citado na página 37.

STATISTA. **Number of available applications in the Google Play Store from December 2009 to June 2023**. 2023. Disponível em: <https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>. Citado na página 62.

SWALLOW, D.; PETRIE, H.; POWER, C. Understanding and supporting web developers: Design and evaluation of a web accessibility information resource (webair). **Studies in Health Technology and Informatics**, IOS Press, v. 229, p. 482–491, 2016. ISSN 18798365. Citado nas páginas 32, 46, 56 e 77.

THATCHER, J.; BOHMAN, P.; BURKS, M.; HENRY, S. L.; REGAN, B.; SWIERENGA, S.; URBAN, M. D.; WADDELL, C. D. **Constructing Accessible Web Sites**. Berkeley, CA: Apress, 2002. ISBN 978-1-59059-148-2 978-1-4302-1116-7. Disponível em: <http://link.springer.com/10.1007/978-1-4302-1116-7>. Citado na página 43.

TURINE, M. A. S.; MASIERO, P. C. **Especificacao de requisitos: uma introducao**. [S.l.]: Icmsc-Usp, 1996. <https://repositorio.usp.br/item/000906321>. Citado na página 95.

TUUNANEN, T.; PEFFERS, K.; HEBLER, S. A requirements engineering method designed for the blind. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 6105 LNCS, p. 475–489, 2010. ISSN 03029743. Cited By 1. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-79955114027&doi=10.1007%

2F978-3-642-13335-0_33&partnerID=40&md5=850fa865b72688d028115ee8df1de311>. Citado na página 74.

USABILITY.GOV. **System Usability Scale (SUS)**. 2023. Disponível em: <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>. Citado na página 109.

VÄÄTÄJÄ, H.; KOPONEN, T.; ROTO, V. Developing practical tools for user experience evaluation - A case from mobile news journalism. In: Norros, L and Koskinen, H and Salo, L and Savioja, P. (Ed.). **VTT Symposium (Valtion Teknillinen Tutkimuskeskus)**. VTT, Finland, Finland: VTT Technical Research Centre of Finland, 2009. (VTT Symposia, 258), p. 240–247. ISBN 9789513863395. ISSN 03579387. Disponível em: <http://dl.acm.org/citation.cfm?id=1690508.1690539>. Citado na página 64.

VENDOME, C.; SOLANO, D.; LIÑÁN, S.; LINARES-VÁSQUEZ, M. Can everyone use my app? an empirical study on accessibility in android apps. In: **2019 IEEE Intl Conf on Software Maintenance and Evolution (ICSME)**. Cleveland, OH, USA: IEEE, 2019. p. 41–52. Citado na página 29.

VONTELL, A. R. **Bility : Automated Accessibility Testing for Mobile Applications**. Tese (Doutorado) — Massachusetts Institute of Technology, 2019. Disponível em: <https://dspace.mit.edu/handle/1721.1/121685>. Citado nas páginas 31, 79, 80, 81, 82, 83, 85, 86 e 87.

W3C. **Web Content Accessibility Guidelines 1.0**. 1999. Disponível em: <https://www.w3.org/TR/WAI-WEBCONTENT/>. Citado na página 44.

____. **Mobile Accessibility at W3C**. 2008. Web Accessibility Initiative (WAI). Disponível em: <https://www.w3.org/WAI/standards-guidelines/mobile/>. Citado na página 46.

____. **Web Content Accessibility Guidelines (WCAG) 2.1**. 2018. Disponível em: <https://www.w3.org/TR/WCAG21/#identify-input-purpose>. Citado nas páginas 44, 45 e 101.

WASSERMAN, A. I. Software engineering issues for mobile application development. In: **Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10**. Santa Fe, New Mexico, USA: ACM Press, 2010. p. 397. ISBN 978-1-4503-0427-6. Disponível em: <http://portal.acm.org/citation.cfm?doid=1882362.1882443>. Citado na página 38.

WILLIAMS, A. User-centered design, activity-centered design, and goal-directed design: a review of three methods for designing web applications. In: **Proceedings of the 27th ACM international conference on Design of communication - SIGDOC '09**. Bloomington, Indiana, USA: ACM Press, 2009. p. 1. ISBN 978-1-60558-559-8. Disponível em: <http://portal.acm.org/citation.cfm?doid=1621995.1621997>. Citado nas páginas 48 e 70.

WINKELMANN, H.; TROOST, L.; KUCHEN, H. Constraint-logic object-oriented programming for test case generation. In: **Proc of the 37th ACM/SIGAPP Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2022. p. 1499–1508. ISBN 9781450387132. Disponível em: <https://doi.org/10.1145/3477314.3507015>. Citado na página 39.

WOLKERSTORFER, P.; TSCHELIGI, M.; SEFELIN, R.; MILCHRAHM, H.; HUSSAIN, Z.; LECHNER, M.; SHAHZAD, S. Probing an agile usability process. In: **CHI '08 Extended Abstracts on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2008. (CHI EA '08), p. 2151–2158. ISBN 9781605580128. Disponível em: <https://doi.org/10.1145/1358628.1358648>. Citado nas páginas 30, 76 e 88.

YAN, S.; RAMACHANDRAN, P. G. The current status of accessibility in mobile apps. **ACM Transactions on Accessible Computing**, Association for Computing Machinery, v. 12, n. 1, 2 2019. ISSN 19367228. Disponível em: <https://doi.org/10.1145/3300176>. Citado nas páginas 30, 31, 60, 62, 63, 85, 86, 89, 94 e 95.

YIN, R. K. **Pesquisa qualitativa do início ao fim**. [S.l.]: Penso Editora, 2016. Citado na página 92.

ZHANG, D.; ADIPAT, B. Challenges, methodologies, and issues in the usability testing of mobile applications. **Intl Journal of HCI**, v. 18, p. 293–308, 7 2005. ISSN 10447318. Disponível em: <http://www.tandfonline.com/doi/abs/10.1207/s15327590ijhc1803_3>. Citado na página 37.

ZHOU, X. C.; XIA, X. Design and research for mobile web learning platform accessibility. In: . [S.l.: s.n.], 2010. p. 478–481. ISBN 9781424469338. ISSN null. Citado na página 75.

GLOSSÁRIO

Android Studio: é um ambiente de desenvolvimento integrado (IDE), disponibilizado gratuitamente sob Licença Apache 2.0, para desenvolver para a plataforma Android.

API: *Application Programming Interface*, termo utilizado para representar o "protocolo" definido em um programa que permite que outros programas interajam com ele. Por exemplo, o sistema operacional de um computador expõe várias APIs para os programas nele instalados executarem diversas funcionalidades em comum, como apresentar janelas, botões, mensagens ao usuário, mecanismos de entrada (mouse, teclado), etc..

Framework: é uma abstração que une códigos comuns entre vários projetos de *software* provendo uma funcionalidade genérica. *Frameworks* são projetados com a intenção de facilitar o desenvolvimento de *software*, habilitando designers e programadores a gastarem mais tempo determinando as exigências do *software* do que com detalhes de baixo nível do sistema.

HTML: *HyperText MarkUp Language*, a linguagem de marcação utilizada para a criação de páginas web. Esta linguagem provê marcas estruturais e semânticas, a qual são colocadas ao redor do conteúdo, com o objetivo tanto de dar uma aparência visual a este conteúdo (estruturando o seu layout), quanto um significado semântico (marcando paragráfos, ênfases, tabelas, formulários, etc.).

Kit: Conjunto de ferramentas, instrumentos ou equipamentos para fins específicos.

Nuvem de *tags*: é uma representação visual em texto de uma lista de etiquetas hierarquizadas, onde a quantidade de repetições das etiquetas são representadas por cores diferentes e pelo tamanho da fonte.

Personas: são arquétipos hipotéticos que representam usuários reais durante o processo de *design*.

Web: Sinônimo mais conhecido de *World Wide Web* (WWW). É a interface gráfica da Internet que torna os serviços disponíveis totalmente transparentes para o usuário e ainda possibilita a manipulação multimídia da informação.

APÊNDICE

A

CHECAGENS DE ACESSIBILIDADE PREDEFINIDAS PELA ATF

ClassNameCheck

Os leitores de tela e outros serviços de acessibilidade podem apresentar informações sobre o tipo de determinados elementos da interface. Exemplo: um leitor de tela pode falar "caixa de seleção"após o marcador associado a uma classe CheckBox.

- *WARNING* quando um elemento é visível e importante para acessibilidade, e se o accessibilityClassName estiver vazio, ou se o nome pertencer a um dos pacotes de interface do usuário padrão suportados

DuplicateClickableBoundsCheck

Às vezes, os desenvolvedores deixam *containers* marcados como sem nenhum evento de clique. Esse erro é difícil de detectar, mas quando um *container* compartilha seus limites com uma View filha, esse é um erro claro.

- *ERROR* se Views compartilham mesmo espaço e ambos são clicáveis.

DuplicateSpeakableTextCheck

Se duas Views em uma hierarquia tiverem o mesmo texto falado, isso pode ser confuso para os usuários.

- *WARNING* se duas Views com o mesmo texto, e pelo menos uma delas é clicável.
- *INFO* se houver duas Views não clicáveis com o mesmo texto.
- ***NOT_RUN*** se nenhuma View na hierarquia tiver qualquer texto falado.

EditableContentDescCheck

- *ERROR* se contentDescription de TextView não estiver vazio.

ImageContrastCheck

– Verifica se *foreground* de ImageView têm contraste suficiente em relação ao *background*.

LinkPurposeUnclearCheck

– *WARNING* se um *link* (ClickableSpan) não tem finalidade clara.

RedundantDescriptionCheck

Verifica se há texto falado que pode conter informações redundantes ou inadequadas. Exemplo: os leitores de tela podem acrescentar "botão" ao texto falado de um Button. Então não faz sentindo incluir "botão" na descrição do conteúdo de um Button.

– Verifica se tem conteúdo em contentDescription, se contentDescription contém o tipo do *widget*, se contentDescription contém indicação de estado ou se contentDescription contém a ação que a View responde.

SpeakableTextPresentCheck

– *ERROR* se os itens que exigem texto falado não possuem algum. Não verifica se o texto faz sentido.

TextContrastCheck

– Verifica se foreground de TextView têm contraste suficiente em relação ao background.

TextSizeCheck

O tipo de dimensão recomendado para texto é o **sp** (*scale-independent pixels*), pois permite que o usuário altere o tamanho do texto usando os recursos de acessibilidade do Android.

Quando um TextView é visível e contém um texto visível, e quando o textSize e o textSizeUnit estão disponíveis, verifica três condições:

1) Se a unidade de dimensão de texto não é sp.
2) Se a unidade de dimensão de texto é sp e o TextView tem largura ou altura fixa.
3) Se a unidade de dimensão de texto é sp e o *container* que contém o TextView tem largura ou altura fixa.

– Pode reportar ERROR, *WARNING* ou *INFO*, dependendo da combinação de condições e tamanho da fonte.

TouchTargetSizeCheck

– Verifica se alvo de toque possui tamanho mínimo (48x48dp por padrão).

TraversalOrderCheck

Detecta problemas na ordenação transversal de acessibilidade especificada pelo desenvolvedor. A ordem de travessia de acessibilidade não deve incluir *loops* ou armadilhas.

- WARNING caso a View esteja envolvida em um *loop* ou se está inacessível por outras definições de accessibilityTraversalBefore e accessibilityTraversalAfter.

APÊNDICE

B

RELATÓRIO DO MAPEAMENTO SISTEMÁTICO GERADO PELA FERRAMENTA PARSIF.AL

Essa pesquisa visa identificar como requisitos de acessibilidade são considerados nos processos de desenvolvimento para *mobile*.

Planejamento

Investigar processos de desenvolvimento para aplicativos *mobile* que considerem acessibilidade e usabilidade.

PICOC

- População: Publicações sobre desenvolvimento *mobile* que considere acessibilidade e usabilidade.
- Intervenção: Processos de desenvolvimento para *mobile*
- Comparação:
- Resultado: Exemplos de processos de desenvolvimento para *mobile* que compreendam requisitos de acessibilidade/usabilidade.
- Contexto:

Questões de Pesquisa

1. Como requisitos de acessibilidade e usabilidade são contemplados durante o processo de desenvolvimento mobile?

Palavras-chave e Sinônimos

Palavra-chave	Sinônimos
accessibility	usability
development process	requirement engineering, requirement specification, software development, software engineering, software process*, software quality,software testing
mobile	

String de busca

("mobile") AND ("development process"OR "requirement engineering"OR "requirement specification"OR "software development"OR "software engineering"OR "software process*"OR "software quality"OR "software testing") AND ("accessibility"OR "usability") NOT ("networks"OR "wireless")

Fontes

- ACM Digital Library (http://portal.acm.org)
- IEEE Digital Library (http://ieeexplore.ieee.org)
- ISI Web of Science (http://www.isiknowledge.com)
- Science@Direct (http://www.sciencedirect.com)
- Scopus (http://www.scopus.com)
- Springer Link (http://link.springer.com)

Critérios de seleção

Critérios de inclusão:

- Estudo primário, escrito em inglês ou português, com texto completo disponível, que descreve processo de desenvolvimento para *mobile* e considera aspectos de acessibilidade.

Critérios de exclusão:

- Estudo não primário.
- Estudo primário não escrito em inglês ou português.
- Estudo primário que aborda acessibilidade ou usabilidade fora do contexto de processo de desenvolvimento.
- Estudo primário que não aborda acessibilidade ou usabilidade.
- Estudo primário que não aborda desenvolvimento *mobile*.
- Estudo primário que não descreve o processo de desenvolvimento
- Estudo primário sem o texto completo disponível.
- Estudo que não corresponda ao critério de inclusão.

- Versão resumida de um mesmo estudo.

Checklist de avaliação da qualidade

Questões:

- A técnica é nova ou a aplicação das técnicas a esse tipo de problema é nova? (possui um diferencial??)
- O problema a ser resolvido pela técnica está claramente explicado?
- A técnica está suficientemente bem descrita para que o autor ou outros possam validá-la em pesquisas posteriores?
- A principal relevância da proposta é demonstrada?
- Existe discussão suficiente sobre trabalhos relacionados? Em outras palavras, as técnicas concorrentes são discutidas e comparadas com esta?

Respostas:

- Sim
- Parcialmente
- Não

Formulário de extração de dados

- Base de busca
- Tipo de estudo
- Metodologia de pesquisa utilizada
- Identificação/nome da técnica
- Breve descrição da proposta
- Denominação(ões) é(são) utilizada(s) para a proposta apresentada
- Processo(s) da norma ISO 12207 apoiado(s) pela técnica
- Filosofias, paradigmas, metodologias e/ou processos abordados
- Que outras técnicas são citadas?
- Quanto à acessibilidade/usabilidade, aborda
- Quanto à acessibilidade, é direcionada para
- Quanto à acessibilidade, é direcionada para (especifique)
- Como acessibilidade é considerada na etapa do desenvolvimento *mobile*?
- A acessibilidade tratada foi validada ?
- Existe alguma implementação na abordagem descrita?
- Foi feita alguma validação com desenvolvedores?
- Domínio no qual a proposta foi apresentada/experimentada
- Plataforma de desenvolvimento

Condução

Strings de pesquisa de bibliotecas digitais

ACM Digital Library: +mobile AND +(usability accessibility) AND +("development process"OR "requirement engineering"OR "requirement specification"OR "software development"OR "software engineering"OR "software process*"OR "software quality"OR "software testing")) AND -("networkswireless")

IEEE Digital Library: ((("All Metadata":"development process") OR ("All Metadata":"requirement engineering") OR ("All Metadata":"requirement specification") OR ("All Metadata":"software development") OR ("All Metadata":"software engineering") OR ("All Metadata":"software process*") OR ("All Metadata":"software quality") OR ("All Metadata":"software testing") OR ("All Metadata":"software architecture")) AND (("All Metadata":accessibility) OR ("All Metadata":usability)) AND "All Metadata":mobile)

Science@Direct: ("mobile") AND ("development process"OR "requirement engineering"OR "requirement specification"OR "software development"OR "software engineering"OR "software process") AND ("accessibility"OR "usability") AND NOT ("networks"OR "wireless")

Scopus: (TITLE-ABS-KEY (mobile) AND TITLE-ABS-KEY (accessibility OR usability) AND ((TITLE-ABS-KEY ("development process"OR "requirement engineering"OR "requirement specification"OR "software development"OR "software engineering"OR "software process*"OR "software quality"OR "software testing"))) AND NOT ("networks"OR "wireless")) AND PUBYEAR > 1999 AND (LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (LANGUAGE , "English") OR LIMIT-TO (LANGUAGE , "Portuguese"))

Estudos importados

- ACM Digital Library: 420
- IEEE Digital Library: 239
- ISI Web of Science: 246
- Science@Direct: 11
- Scopus: 328
- Springer Link: 369

APÊNDICE

C

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO - ENTREVISTA COM DESENVOLVEDORES

Você está sendo convidado (a) para participar da pesquisa “Uma abordagem dirigida a testes para o desenvolvimento de aplicativos móveis acessíveis”.

O objetivo deste estudo é propor um método para apoiar desenvolvedores, que seja mais pragmático (ou seja, auxiliando durante o desenvolvimento, com orientação prática sobre as soluções tecnológicas relacionadas à acessibilidade) na produção de aplicativos móveis acessíveis. Você foi selecionado (a) por trabalhar em empresa no ramo de Tecnologia da Informação, na função de designer, desenvolvedor e/ou testador. Sua participação é voluntária, isto é, a qualquer momento você pode desistir de participar e retirar seu consentimento. A sua recusa não trará nenhum prejuízo na sua relação com o pesquisador ou com a instituição que forneceu os dados.

A coleta de dados será composta por um questionário para coleta do seu perfil acadêmico e profissional, e uma entrevista não estruturada em que você será convidado a relatar a sua experiência no tema de acessibilidade. O tempo utilizado no estudo será de aproximadamente uma 1 hora.

Suas respostas serão tratadas de forma anônima e confidencial, ou seja, em nenhum momento será divulgado seu nome em qualquer fase do estudo. Quando for necessário exemplificar determinada situação, sua privacidade será assegurada. Os dados coletados poderão ter seus resultados divulgados em eventos, revistas e/ou trabalhos científicos.

O preenchimento destes questionários não oferece risco imediato a você, porém considera-se a possibilidade de um risco subjetivo, pois algumas perguntas podem remeter à algum desconforto, evocar sentimentos ou lembranças desagradáveis ou levar à um leve cansaço após responder os questionários. Caso algumas dessas possibilidades ocorram, você poderá optar pela

suspensão imediata da entrevista.

Você não terá nenhum custo ou compensação financeira ao participar do estudo, pois toda ela acontecerá de forma virtual. No entanto, você terá direito a indenização por qualquer tipo de dano resultante da sua participação na pesquisa.

Este trabalho poderá contribuir de forma direta na ampliação do seu conhecimento sobre o tema da acessibilidade em soluções computacionais e incentivá-lo (a) a discutir o tema com outros profissionais.

Você receberá uma via deste termo, rubricada em todas as páginas por você e pelos pesquisadores, onde consta o telefone e o endereço do pesquisador principal com quem você poderá tirar suas dúvidas sobre a pesquisa e sua participação agora ou a qualquer momento.

Este projeto de pesquisa foi aprovado por um Comitê de Ética em Pesquisa (CEP) que é um órgão que protege o bem-estar dos participantes de pesquisas. O CEP é responsável pela avaliação e acompanhamento dos aspectos éticos de todas as pesquisas envolvendo seres humanos, visando garantir a dignidade, os direitos, a segurança e o bem-estar dos participantes de pesquisas. Caso você tenha dúvidas e/ou perguntas sobre seus direitos como participante deste estudo, entre em contato com o Comitê de Ética em Pesquisa em Seres Humanos (CEP) da UFSCar que está vinculado à Pró-Reitoria de Pesquisa da universidade, localizado no prédio da reitoria (área sul do campus São Carlos). Endereço: Rodovia Washington Luís km 235 - CEP: 13.565-905 - São Carlos-SP. Telefone: (16) 3351-9685. E-mail: cephumanos@ufscar.br. Horário de atendimento: das 08:30 às 11:30.

O CEP está vinculado à Comissão Nacional de Ética em Pesquisa (CONEP) do Conselho Nacional de Saúde (CNS), e o seu funcionamento e atuação são regidos pelas normativas do CNS/Conep. A CONEP tem a função de implementar as normas e diretrizes regulamentadoras de pesquisas envolvendo seres humanos, aprovadas pelo CNS, também atuando conjuntamente com uma rede de Comitês de Ética em Pesquisa (CEP) organizados nas instituições onde as pesquisas se realizam. Endereço: SRTV 701, Via W 5 Norte, lote D - Edifício PO 700, 3º andar - Asa Norte - CEP: 70719-040 - Brasília-DF. Telefone: (61) 3315-5877 E-mail: conep@saude.gov.br.

Dados para contato (24 horas por dia e sete dias por semana):

Pesquisador Responsável: Anderson Canale Garcia Endereço: Av. Trab. São Carlense, 400 - Centro, São Carlos - SP, 13566-590 E-mail: andersongarcia@usp.br

Declaro que entendi os objetivos, riscos e benefícios de minha participação na pesquisa e concordo em participar.

Local e data: _____

Nome do participante	Assinatura do participante
Anderson Canale Garcia Mestrando em Computação (ICMC/USP)	Profa. Dra. Kamila Rios da Hora Rodrigues Professora e Pesquisadora do ICMC/USP

APÊNDICE

D

MENSAGEM CONVITE AOS PROFISSIONAIS DE EMPRESA PARA ENTREVISTAS

Prezado(a)

Você está sendo convidado a participar da nossa pesquisa, cujo objetivo é entender como o requisito acessibilidade tem sido projetado e implementado na empresa onde você trabalha e nas suas práticas profissionais.

Gostaríamos de entender as principais questões em torno do tema e as possíveis dificuldades.

Caso aceite participar, você receberá um formulário online, no qual deverá consentir sua participação e terá acesso a mais detalhes sobre o projeto. Em seguida, neste mesmo formulário, você deverá responder sobre informações do seu perfil de formação e profissional. Ao fim desta etapa conduziremos uma entrevista não estruturada com você para entender sobre sua experiência e considerações sobre o tema. Toda a sessão deverá levar em média 1 hora.

Reforçamos que os seus dados serão anonimizados e que você pode desistir de participar a qualquer momento sem prejuízos envolvidos. Reforçamos ainda que não há custos financeiros envolvidos.

Agradecemos desde já e aguardamos o seu retorno.

Atenciosamente,

Anderson Canale Garcia Mestrando em Computação (ICMC/USP) andersongarcia@usp.br

Dra. Kamila Rios da Hora Rodrigues Professora e Pesquisadora do ICMC/USP kamila.rios@icmc.usp.br

APÊNDICE

E

ROTEIRO DA ENTREVISTA COM DESENVOLVEDORES

Propósito do estudo:

- Compreender como a acessibilidade é abordada nos projetos em que os profissionais atuam.
- Compreender como são atribuídas responsabilidades para verificações de acessibilidade em projetos de desenvolvimento *mobile*.
- Obter a percepção dos desenvolvedores sobre a viabilidade e adequação de uma proposta dirigida por testes para inserção de acessibilidade.

Tópicos a serem abordados:

- Qual seu papel atualmente nos projetos de desenvolvimento?
- Você atua em algum projeto de desenvolvimento mobile atualmente? Com Android?
- Como a acessibilidade tem sido considerada nos projetos em que atua?
- A quem tem sido atribuída a responsabilidade pelas verificações de acessibilidade? E de quem você entende que deveria ser essa responsabilidade?
- Qual o suporte recebido dos líderes e dirigentes para considerar a acessibilidade?
- Qual sua perspectiva sobre uma proposta para inserção de requisitos de acessibilidade com TDD?
- Que outras estratégias você imagina que possam ser utilizadas?

APÊNDICE

F

RELAÇÃO DE ARTEFATOS DO AALT

A seguinte relação apresenta os artefatos que compõem o *Framework* AALT. Para cada artefato serão apresentados seu nome, sua descrição em relação ao conteúdo oferecido, sua categoria (eixo horizontal) e sua responsabilidade (eixo vertical).

Requisitos de acessibilidade

Descrição: Conjunto de requisitos de acessibilidade, derivados das Recomendações de Acessibilidade Móvel (maRs) (DIAS; DUARTE; FORTES, 2021), escritos em forma de declarações testáveis, e ordenados por ordem de prioridade para automação em uma estratégia de testes em pirâmide.

Categoria: Recursos.

Responsabilidade: Os requisitos podem ser usados por ***designers***, **desenvolvedores** e **testadores**. Porém, sugere-se que os testes pequenos sejam implementados por desenvolvedores, e os testes grandes por testadores.

AATK - Automated Accessibility Testing Kit for Android Apps

Descrição: Biblioteca Android escrita em Java para ajudar desenvolvedores na criação de testes de acessibilidade para projetos Android. Contém 5 testes escritos para serem executados com o Roboletric, e portanto não requerem uso de dispositivo físico ou emulado. Os testes disponíveis são:

- Constraste de cor
- Tamanho do alvo de toque
- Espaçamento

- Rótulo de componente
- Texto alternativo

O AATK está disponível em:
<https://github.com/AALT-Framework/android-accessibility-test-kit>.

Categoria: Ferramentas.

Responsabilidade: O AATK é principalmente dirigido aos **desenvolvedores**, mas também podem ser utilizados pelos **testadores**, dependendo da estratégia da equipe de desenvolvimento.

ATTD - Accessibility Testing Tool for Designers (em desenvolvimento)

Descrição: Projeto de aplicativo Android, pré-configurado com testes de acessibilidade, para ser usado por designers para validar a acessibilidade de componentes de UI.

Categoria: Ferramentas.

Responsabilidade: O ATTD é dirigido aos **designers**.

Guia de Ferramentas

Descrição: Uma relação das ferramentas atualmente disponíveis para testes de acessibilidade no Android, classificados por tipo de teste, execução (local ou instrumentado), sugestões de uso e como acessar.

Categoria: Ferramentas.

Responsabilidade: As ferramentas podem ser usadas por ***designers***, **desenvolvedores** e **testadores**, dependendo da proposta de cada uma delas.

Material Design

Descrição: Link para recursos do Material Design dedicados a acessibilidade, acompanhados de considerações de pesquisa relacionadas a esses itens.

Categoria: Recursos.

Responsabilidade: Este recurso é principalmente útil para ***designers***.

UIDPs (em desenvolvimento)

Descrição: Biblioteca de UIDPs com acessibilidae.

Categoria: Recursos.

Responsabilidade: Este recurso é principalmente útil para ***designers***.

Exemplos de código

Descrição: Arquivos com código de UI ilustrando exemplos de elementos bem definidos quanto à acessibilidade. Os exemplos são inspirados no que existe nos guias BBC (BBC, 2022) e SIDI (SIDI, 2017), porém, aproveitando os recursos do GitHub para compartilhar os exemplos em código (arquivos XML, inicialmente).

Categoria: Recursos.

Responsabilidade: Este recurso é principalmente útil para **desenvolvedores**.

Estratégia de Testes

Descrição: Seleção de ferramentas e orientações para diferentes estratégias de testes de acessibilidade. Disponibilizado dentro do site público do *Framework* AALT.

Categoria: Recursos.

Responsabilidade: Este recurso é principalmente útil para **testadores**.

Políticas

Descrição: Página Web com um resumo das leis e de outras políticas públicas voltadas à garantia de acessibilidade digital. Porém o principal objetivo deste artefato é fornecer um *slot* no AALT para que as organizações criem e implemente suas próprias políticas internas, envolvendo formação, conscientização e fiscalização.

Categoria: Recursos.

Responsabilidade: Este recurso é destinado aos ***stakeholders***.

Diretrizes e Recomendações

Descrição: Página Web com as recomendações *maRs*, contendo exemplos, e uma lista de links para os principais conjuntos de diretrizes e recomendações apresentados neste trabalho.

Categoria: Aprendizado.

Responsabilidade: Este artefato é aplicável a todos os atores, inclusive gestores e clientes.

Codelabs

Descrição: Coleção de treinamentos em formato de *codelabs*. Inclui treinamentos próprios e treinamentos do Google Codelabs com foco em acessibilidade.

Categoria: Aprendizado.

Responsabilidade: Os *codelabs* atendem inicialmente a designers, desenvolvedores e testadores. Porém, podem ser elaborados também treinamentos para outros atores.

Apresentações

Descrição: Conjunto de materiais para serem usados por gestores na divulgação e promoção de conteúdo sobre acessibilidade digital.

Categoria: Aprendizado.

Responsabilidade: Este artefato é destinado aos **stakeholders**.

Conscientização

Descrição: Todo o conteúdo disponibilizado para ampliar a conscientização de todos sobre a acessibilidade digital. O AALT oferecerá isso em forma de um site público, em publicações, apresentações e aulas. Porém, este deve ser um compromisso de todos.

Categoria: Aprendizado.

Responsabilidade: Este artefato é aplicável a todos os atores, internos e externos da organização.

APÊNDICE

G

FORMULÁRIO DE AVALIAÇÃO DO AATK

Testes de acessibilidade em *apps* Android

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO - TCLE

Olá! Você está sendo convidado a participar de nossa pesquisa, cujo objetivo é registar a sua experiência com a utilização de uma ferramenta para testes de acessibilidade em aplicativos Android, e está dividida em três etapas:

1. **Identificação de perfil** (2 minutos): serão 14 perguntas simples para conhecermos um pouco da sua experiência com ferramentas e conceitos envolvidos.
2. **Codelab** (20 minutos): você será guiado a identificar e corrigir problemas de acessibilidade em um projeto de aplicativo Android, usando uma de três opções de ferramentas disponíveis.
3. **Registro de experiência** (5 minutos): faremos algumas perguntas sobre sua experiência com a ferramenta utilizada e os resultados obtidos.

DESCONFORTOS, RISCOS E BENEFÍCIOS

Sua participação na pesquisa pode envolver algum desconforto relacionado ao tempo despendido nas atividades previstas, porém, tudo foi planejado para acontecer de forma rápida. Estimamos que você leve em torno de 30 minutos entre a execução do codelab e o preenchimento dos formulários.

Você pode interromper a sua participação a qualquer momento, sem qualquer prejuízo em sua relação com as instituições ou com os pesquisadores envolvidos.

Os benefícios para a sua participação nesta pesquisa incluem: a contribuição científica gerada e o aprendizado acerca de ferramentas para teste de acessibilidade em dispositivos Android.

CUSTOS DA PARTICIPAÇÃO, RESSARCIMENTO E INDENIZAÇÃO POR EVENTUAIS DANOS:

A sua participação é voluntária e, em decorrência dela, você não receberá qualquer valor em dinheiro. Você não terá nenhum gasto por participar desse estudo.

As informações obtidas nesta pesquisa não serão associadas com sua identidade, permanecendo anônimas e garantindo sua privacidade. Garantimos que seus dados pessoais serão mantidos em sigilo e segurança.

Você poderá entrar em contato com qualquer uma das pesquisadoras responsáveis pela pesquisa no ICMC/USP, caso tenha dúvidas.

Desde já agradecemos sua participação.

Anderson Canale Garcia
Mestrando em Computação (ICMC/USP)
andersongarcia@usp.br

Dra. Kamila Rios da Hora Rodrigues
Professora e Pesquisadora do ICMC/USP
kamila.rios@icmc.usp.br

* Indica uma pergunta obrigatória

1. E-mail *

2. Eu li e/ou ouvi o esclarecimento acima e compreendi para que serve o estudo e a quais procedimentos serei submetido. A explicação que recebi esclarece os riscos e benefícios do estudo. Eu entendi que sou livre para interromper a autorização a qualquer momento, sem justificar minha decisão e que isso não afetará o questionário que estou recebendo. Sei que meu nome não será divulgado, que não terei despesas e não receberei dinheiro para participar do estudo. *

   Marcar apenas uma oval.

   - ◯ Eu concordo em participar.
   - ◯ Eu não concordo em participar.

3. Data *

   Exemplo: 7 de janeiro de 2019

Perfil do avaliador

Essa seção contém perguntas sobre você e a sua experiência com os conceitos e ferramentas envolvidos neste estudo.

4. Data de nascimento *

_____

Exemplo: 7 de janeiro de 2019

5. Qual a sua formação acadêmica? *
Ex.: Ciência da Computação, Sistemas de Informação, Design Digital, Design de Jogos, etc.

_____

6. Qual é o seu nível de formação? *

Marcar apenas uma oval.

- ◯ Graduação (em andamento)
- ◯ Graduação
- ◯ Pós-graduação Lato-Sensu
- ◯ Mestrado
- ◯ Doutorado
- ◯ Pós-doutorado
- ◯ Outro: _____

7. Caso esteja estudando, qual é o período em que está no curso?

_____

8. Sobre sua atividade *

Marcar apenas uma oval.

- ◯ Sou estudante
- ◯ Trabalho tempo parcial
- ◯ Trabalho tempo integral
- ◯ Desempregada(o)
- ◯ Aposentada(o)
- ◯ Outro: _____

9. Caso esteja trabalhando, como você descreveria sua atribuição?
Ex.: desenvolvedor back end, estagiário front end, analista de QA, etc.

_____

10. Há quanto tempo você trabalha nesta função?
OBS: Apenas para aqueles que trabalham tempo parcial ou integral.

_____

11. Como você classifica sua experiência em programação para Android? *

Marcar apenas uma oval.

Nenhuma experiência

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Muita experiência

12. Em relação ao desenvolvimento de *apps* Android, classifique seu nível de experiência com cada um dos tópicos abaixo. *

Selecione de 1 a 5, sendo 1 para "Nenhuma experiência" e 5 para "Muita experiência"

Marcar apenas uma oval por linha.

	1	2	3	4	5
Android Studio	◯	◯	◯	◯	◯
Java	◯	◯	◯	◯	◯
Kotlin	◯	◯	◯	◯	◯
XML Layout	◯	◯	◯	◯	◯
Jetpack Compose	◯	◯	◯	◯	◯
Flutter	◯	◯	◯	◯	◯
React Native	◯	◯	◯	◯	◯

13. Como você classifica seu conhecimento em acessibilidade? *

Marcar apenas uma oval.

Nenhum conhecimento	
1	◯
2	◯
3	◯
4	◯
5	◯
Conhecimento avançado	

14. Em relação a acessibilidade em dispositivos móveis, classifique seu nível de conhecimento sobre cada um dos tópicos abaixo *

Marcar apenas uma oval por linha.

	Desconheço	Conheço, mas NÃO SEI utilizar	Conheço, e SEI utilizar
WCAG - Web Content Accessibility Guidelines	◯	◯	◯
Guia de acessibilidade para desenvolvedores Android	◯	◯	◯
Guia de acessibilidade para desenvolvedores iOS	◯	◯	◯
Outro(s) guia(s)	◯	◯	◯
TalkBack	◯	◯	◯
Switch Access	◯	◯	◯
Voice Access	◯	◯	◯

15. Que outros guias/ferramentas de apoio à acessibilidade para dispositivos móveis você conhece? *

Se não conhece, diga "Não" neste campo.

_____

16. Como você classifica seu conhecimento em testes automatizados? *

Marcar apenas uma oval.

Nenhum conhecimento	
1	◯
2	◯
3	◯
4	◯
5	◯
Conhecimento avançado	

17. Classifique seu nível de conhecimento sobre cada um dos tópicos abaixo *

Marcar apenas uma oval por linha.

	Desconheço	Conheço, mas NÃO SEI utilizar	Conheço, e SEI utilizar
Android Lint	◯	◯	◯
Scanner de Acessibilidade	◯	◯	◯
UI Automator	◯	◯	◯
Espresso	◯	◯	◯
Roboletric	◯	◯	◯
JUnit	◯	◯	◯

Codelab

Aplicativos *mobile* devem ser utilizáveis por todos, incluindo pessoas com deficiência. Testes de acessibilidade ajudam a orientar os desenvolvedores quanto a melhorias a serem realizadas no projeto do app.

A plataforma Android disponibiliza uma série de ferramentas que podem ser utilizadas nas diferentes estratégias de testes de acessibilidade.

Neste estudo, estamos registrando a experiência de desenvolvedores com três ferramentas diferentes para resolver problemas de acessibilidade em aplicativos de exemplo.

1. Accessibility Scanner: *app* disponibilizado pelo Google que, quando habilitado, é capaz de analisar a tela de qualquer aplicativo e apresentar ao usuário uma visão que destaca os problemas de acessibilidade diretamente no interface de usuário. Requer uso de dispositivo físico ou emulado, com *app* Scanner de Acessibilidade;
2. Espresso:

   biblioteca de testes do Android criada para fazer testes de IU de forma automatizada, de forma integrada ao Android Studio. Requer uso de dispositivo físico ou emulado;
3. Automated accessibility tests kit for Android apps (AATK): biblioteca com conjunto de testes de acessibilidade automatizados projetados para serem executados como testes locais, sem a necessidade de um dispositivo físico ou emulado.

Escolha, de acordo com seu interesse, com qual ferramenta deseja realizar os testes.

18. Com qual ferramenta você deseja realizar o codelab? *

Marcar apenas uma oval.

- ◯ Accessibility Scanner
  *Pular para a seção 4 (**Codelab com Accessibility Scanner**)*
- ◯ Espresso *Pular para a seção 5 (**Codelab com Espresso (em inglês)**)*
- ◯ AATK *Pular para a seção 6 (**Codelab com AATK**)*

Codelab com Accessibility Scanner

Este é um codelab elaborado pela Google para desenvolvedores Android que querem entender como tornar os apps acessíveis para usuários com deficiência.

Clique no link abaixo para abrir o codelab. Execute todos os passos, e RETORNE, por favor, a este questionário para registrar a sua experiência.

Como iniciar a acessibilidade no Android

Pular para a pergunta 19

Codelab com Espresso (em inglês)

Este é um codelab elaborado pela Google e consiste em escrever testes automatizados para acessibilidade. Em particular, ele demonstra como integrar testes de acessibilidade em um conjunto de testes que utiliza o framework de testes Espresso.

Clique no link abaixo para abrir o codelab. Execute todos os passos, e RETORNE, por favor, a este questionário para registrar sua experiência.

Aplicando Testes de UI com o Espresso para o projeto do app Contador

Pular para a pergunta 19

Codelab com AATK

Este é um codelab elaborado pelos autores consiste em escrever testes automatizados para acessibilidade utilizando a biblioteca AATK, que consiste em uma coleção de testes de acessibilidade automatizados projetados para serem executados com o Robolectric. Isso permite que sejam executados como testes locais, sem a necessidade de um dispositivo físico ou emulado.

Clique no link abaixo para abrir o codelab. Execute todos os passos, e RETORNE, por favor, a este questionário para registrar sua experiência.

Aplicando o Kit de Testes de Acessibilidade Automatizados para Aplicativos Android (AATK) para o projeto do app Counter

Pular para a pergunta 19

Registro de Experiência

19. Por favor, nos diga qual a razão que te levou a escolher a ferramenta usada neste teste. *

20. Você conseguiu realizar as tarefas utilizando a ferramenta de testes automatizados de acessibilidade para aplicativos Android? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Parcialmente
- ◯ Outro: _____

Para cada uma das afirmações abaixo selecione a alternativa que melhor descreve a sua reação à ferramenta que acabou de utilizar

21. Eu acho que gostaria de usar essa ferramenta frequentemente *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

22. Eu achei a ferramenta desnecessariamente complexa *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

23. Eu achei a ferramenta fácil de usar *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

24. Eu acho que precisaria de um suporte técnico para poder usar a ferramenta *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

25. Eu achei que as várias funções da ferramenta estão bem integradas *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

26. Eu acho que a ferramenta apresenta muitas inconsistências *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

27. Eu imagino que as pessoas aprenderão a usar essa ferramenta rapidamente *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

28. Eu achei a ferramenta muito complicada de usar *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

29. Eu me senti muito confiante usando a ferramenta *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

30. Eu precisava aprender muitas coisas antes de poder usar essa ferramenta *

Marcar apenas uma oval.

Discordo completamente

1 ◯

2 ◯

3 ◯

4 ◯

5 ◯

Concordo completamente

31. Caso deseje, utilize o campo abaixo para quaisquer outras observações que deseje fazer sobre a atividade realizada, a ferramenta testada ou qualquer outra observação que deseje realizar.

Encerramos o teste.
Agradecemos pela sua participação.


ICMC
USP
SÃO CARLOS