LUAN MELO

COMPONENTES DE INTERFACES GRÁFICAS ACESSÍVEIS PARA APLICAÇÕES MÓVEIS: UMA ABORDAGEM SEMIAUTOMÁTICA

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná para obtenção do título de Mestre em Informática.

Curitiba
2019

LUAN MELO

COMPONENTES DE INTERFACES GRÁFICAS ACESSÍVEIS PARA APLICAÇÕES MÓVEIS: UMA ABORDAGEM SEMIAUTOMÁTICA

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná para obtenção do título de Mestre em Informática.
Área de concentração: Engenharia de Software
Orientadora: Profa. Dra Andreia Malucelli
Coorientador: Prof. Dr. André Menolli

Curitiba
2019

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor, com anuência de seu orientador.

Curitiba, dia 24 do mês de agosto do ano de 2021.

Assinatura do Autor

Assinatura do Orientador

FICHA CATALOGRÁFICA

Melo, Luan
Componentes de interfaces gráficas acessíveis par aplicações móveis: uma abordagem semiautomática, Curitiba, 2019, 83 p.

Dissertação (Mestrado) – Pontifícia Universidade Católica do Paraná. Curitiba. Programa de Pós-Graduação em Informática.

1. Accessibilities 2. Model-Driven Development 3. User Interfaces 4. Mobile devices
Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática d.

DEDICATÓRIAS

Aos meus pais Kátia e Edilson.

Aos meus amigos que sempre me motivaram a seguir em frente.

AGRADECIMENTOS

Aos meus pais *Kátia* e *Edilson* que sempre estiveram presentes dando apoio para que eu não desistisse dos meus sonhos e me incentivaram a sempre continuar.

À minha orientadora *Andreia*, pelo carinho, ensinamentos e conhecimento compartilhado, e pela inspiração, que me incentivou a querer buscar mais e mais conhecimento. Andreia, obrigado por ter acreditado em mim.

À professora *Sheila*, pelo carinho, amizade, apoio, conhecimento e incentivo que contribuíram para o desenvolvimento do trabalho.

Ao meu coorientador *André*, pela amizade, por ter acreditado em mim, pelos ensinamentos passados desde o início da minha graduação, pela prestatividade, dedicação e por ter me incentivado a continuar os estudos. André, muito obrigado.

Aos meus amigos, que sempre estiveram comigo durante todos estes anos, em especial à *Letícia e Larissa*, companheiras e irmãs. Obrigado, vocês são demais.

Ao meu colega *Elias*, pela amizade, conversas e altas emoções que passamos juntos para realizar nossas pesquisas do mestrado.

Aos amigos e colegas do grupo de pesquisa, por compartilhar momentos incríveis, muito conhecimento, aprendizados e motivações.

A todos os meus professores, tanto da UENP quanto do PPGIa, que pessoalmente e profissionalmente contribuíram, de alguma forma, para a realização deste trabalho.

Ao meu tio *Heitor*, por todas as coisas que fez por mim, desde a graduação até agora.

Aos meus tios *Adilton* e *Elizabeth*, por ter me hospedado em sua casa quando cheguei a Curitiba.

“Nós só podemos ver um pouco do futuro, mas o suficiente
para perceber que há muito a fazer.”
(Alan Turing)

RESUMO

Embora tenham sido propostas várias diretrizes para promover a acessibilidade, existem ainda vários aplicativos que não são acessíveis. Com o objetivo de contribuir para a mudança deste cenário, este artigo apresenta uma abordagem baseada em desenvolvimento dirigido a modelos e apoiada por diretrizes de desenvolvimento de interfaces gráficas para auxiliar os desenvolvedores na geração semiautomática de componentes de interfaces gráficas acessíveis para aplicativos móveis. A pesquisa foi desenvolvida em quatro etapas principais: (i) identificação de diretrizes e recomendações de acessibilidade, (ii) desenvolvimento da abordagem para geração semiautomática de componentes de interfaces gráficas acessíveis para dispositivos móveis, (iii) desenvolvimento de ambiente para geração de componentes de interfaces gráficas acessíveis, e (iv) avaliação da abordagem por meio de um estudo experimental. A avaliação contou com 28 participantes, sendo seis na fase do estudo piloto e 22 na avaliação final. Cada participante criou um projeto e ao final 125 componentes foram comparados com as recomendações de acessibilidade. Os resultados obtidos indicam que a abordagem apoia o desenvolvimento de componentes de interfaces gráficas acessíveis para aplicativos móveis. No melhor do nosso conhecimento, este é o primeiro estudo voltado para o contexto de aplicativos móveis e nativos que utiliza diretrizes de acessibilidade. A abordagem proposta pode ser utilizada tanto para o contexto de aplicativos nativos quanto para Web.

Palavras-chaves: Acessibilidade, Desenvolvimento Dirigido a Modelo, Dispositivos móveis, Interfaces gráficas.

ABSTRACT

Although several guidelines have been proposed to promote accessibility, there are still many applications that are not accessible. In order to contribute to changing this scenario, this article presents an approach based on model-driven development and supported by GUI development guidelines to assist developers in the semi-automatic generation of accessible GUI components for mobile applications. The research was carried out in four main stages: (i) identification of accessibility guidelines and recommendations, (ii) development of the approach for semi-automatic generation of accessible graphical user interface components for mobile devices, (iii) development of an environment for the generation of mobile components. accessible graphical interfaces, and (iv) evaluation of the approach through an experimental study. The evaluation had 28 participants, six in the pilot study phase and 22 in the final evaluation. Each participant created a project and in the end 125 components were compared with the accessibility recommendations. The results obtained indicate that the approach supports the development of accessible graphical user interface components for mobile applications. To the best of our knowledge, this is the first study focused on the context of mobile and native apps that uses accessibility guidelines. The proposed approach can be used both for the context of native applications and for the Web.

Keywords: Accessibility, Model Driven Development, Mobile Devices, User Interfaces.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

DSL	Domain Specific Language
e-MAG	Modelo de Acessibilidade do Governo Eletrônico
MDA	Model-Driven Architecture
MDD	Model-Driven Development
ONU	Organização das Nações Unidas
PIM	Platform Independent Model
PSM	Platform Specific Models
RENAPI	Rede de Pesquisa e Inovação em Tecnologias Digitais
SISP	Sistema de Administração dos Recursos de Informação e Informática
TIC	Tecnologia de Informação e Comunicação
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WCAG	Web Content Accessibility Guidelines 2.0

CAPÍTULO 1 - INTRODUÇÃO

“Lutar pela igualdade sempre que as diferenças nos discriminem e lutar pelas diferenças sempre que a igualdade nos descaracterize” (Boaventura de Souza Santos)

Segundo Kelm, Mello e Bernardino (2018), o mercado de dispositivos móveis vem evoluindo constantemente. Até 2022 o número de dispositivo móveis que estarão conectados à internet será de quase dois dispositivos por pessoa, ou seja, chegará a aproximadamente 12,3 bilhões de dispositivos conectados. Assim, os dispositivos móveis são promissores para produtos e serviços produzidos pela indústria de software, que terá o desafio de desenvolver aplicações móveis a todas as pessoas, incluindo as pessoas com deficiência (Serra et al., 2015).

No entanto, para Wasserman (2010) os aplicativos móveis têm que considerar várias limitações inerentes ao ecossistema móvel, tais como: o tamanho da tela, o tipo de aplicação, os desenvolvedores adotarem o uso de boas práticas de desenvolvimento, os desenvolvedores ainda estarem pouco organizados em relação aos esforços e métricas, e ausência de elementos interativos, como o teclado físico.

Assim, para a construção de aplicativos móveis que estejam acessíveis ao maior número de pessoas, incluindo as pessoas com deficiência, é necessário considerar aspectos diferentes durante o desenvolvimento da interface gráfica desses aplicativos (MEDEIROS, 2013).

Apesar das inúmeras pesquisas e esforços governamentais para fornecer leis e diretrizes que reforcem e promovam a criação de interfaces gráficas acessíveis, ainda existem vários aplicativos que não são totalmente acessíveis ao usuário com deficiência. Isto acontece devido à diferentes fatores: dificuldade de implementação das interfaces gráficas de forma acessível, evolução da tecnologia móvel, falta de ferramentas de apoio ao desenvolvimento acessível e, (4) segundo Power et al. (2012), há menos trabalhos, técnicas e diretrizes bem estabelecidas para dispositivos móveis do que para outro ambiente, como a Web.

Considerando esta realidade, várias iniciativas foram propostas para promover a acessibilidade por meio da aplicação de diretrizes, como a Android Accessibility Practices, a Age-centered Research-Based Design Guidelines e as Diretrizes de Acessibilidade para Conteúdo WEB 2.0 (WCAG 2.0), que é a mais utilizada.

Com isto, umas das iniciativas que visam promover a acessibilidade é de como aplicar, no contexto de dispositivos móveis, as Diretrizes de Acessibilidade para Conteúdo WEB 2.0 (WCAG 2.0, do inglês *Web Content Accessibility Guidelines* 2.0). Apesar da WCAG ser a mais utilizada, existem outras diretrizes, tais como *Android Accessibility Practies*, *Age-centered Research-Based Design Guidelines*, Modelo de Acessibilidade de Governo Eletrônico e *Mobile Web Best*.

Contudo, embora estas diretrizes estejam disponíveis para a criação de interfaces gráficas acessíveis para dispositivos móveis, ainda há necessidade de conscientizar os desenvolvedores e encorajá-los a implementar aplicativos acessíveis (OLIVEIRA; FILGUEIRAS, 2018). Assim, torna-se importante investigar abordagens e ferramentas que auxiliem na adoção de diretrizes de acessibilidade, como, por exemplo, para geração de componentes de interfaces gráficas.

Considerando o exposto, uma maneira de facilitar o desenvolvimento de interfaces gráficas acessíveis para dispositivos móveis é utilizando o Desenvolvimento Dirigido a Modelos (MDD). Brambilla, Cabot e Wimmer (2012) definem o MDD como um paradigma que utiliza modelos como artefatos primários no processo de desenvolvimento, e geralmente a implementação é gerada de forma automática ou semiautomática. O MDD se mostra um possível facilitador para este processo, pois desenvolvedores têm utilizado o MDD para criar aplicativos simples e confiáveis em menos tempo. Como exemplo, no trabalho de Antonelli, Silva e Fortes (2015), o MDD foi utilizado para gerar menus acessíveis para Web. Por meio do MDD é possível criar aplicativos baseadas em modelos, capazes de gerar códigos de forma total ou parcial. Os desenvolvedores, ao utilizar o MDD, podem se concentrar em conceitos importantes, ao invés de pensar em detalhes técnicos, facilitando o gerenciamento da interface do usuário. Com essa abordagem, o nível de complexidade de desenvolvimento é reduzido. Além disso, a arquitetura da interface também é simplificada, aumentando a compreensão e a rastreabilidade do sistema, tornando assim manutenções mais fáceis e menos custosas (AHAMED; ASHRAF, 2007).

Diante do exposto, este trabalho apresenta uma abordagem baseada em MDD para facilitar a criação de componentes de interfaces gráficas acessíveis para dispositivos móveis. A abordagem se apoia nas diretrizes de acessibilidade e foi desenvolvida para auxiliar os desenvolvedores na criação de interfaces gráficas acessíveis.

Por meio do MDD é possível criar uma padronização de códigos, pois é garantido que todas as soluções tenham as classes que respeitam o padrão que foi estabelecido no PIM. E, por meio das regras de transformação é possível aplicar as recomendações de acessibilidade diretamente nas classes, garantido que os componentes de interface gráfica fiquem acessíveis, sem a necessidade do desenvolvedor conhecer essas recomendações.

Neste cenário surge a seguinte questão de pesquisa: Como o MDD pode facilitar a criação de componentes de interfaces gráficas acessíveis para dispositivos móveis?

1.1 Objetivos

Devido ao fato de que muitas interfaces gráficas para dispositivos móveis não são desenvolvidas de acordo com as normas que garantem serem inclusivas à todas as pessoas, este trabalho tem o objetivo geral de: **propor uma abordagem baseada em MDD para geração semiautomática de componentes de interfaces gráficas acessíveis para aplicações móveis.**

Para que se possa atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

i. Conceber a abordagem.
ii. Desenvolver um ambiente para apoiar a abordagem.
iii. Avaliar a abordagem.

1.2 Delimitação de escopo

A abordagem não pretende substituir os desenvolvedores de interfaces, conhecidos como *front-end*, e nem retirar as práticas já adotadas pelas empresas desenvolvedoras de software. A proposta é de que a abordagem seja um meio para auxiliar os profissionais no desenvolvimento de componentes de interfaces gráficas de forma rápida e de acordo com as diretrizes de acessibilidade.

A abordagem proposta foi criada de forma a ser possível criar componentes de interfaces gráficas acessíveis para diferentes sistemas operacionais e para diferentes tipos de deficiência. Para avaliar a abordagem foi necessário implementar um ambiente e, como não seria possível abranger todos os sistemas operacionais e tipos de deficiência durante a avaliação, o escopo da pesquisa se restringiu ao desenvolvimento de um ambiente para apoiar a geração semiautomática de componentes de interfaces gráficas acessíveis para dispositivos móveis que possuem o sistema operacional android. Para este trabalho, foi escolhido o Design Universal para que se possa abranger uma maior variedade de pessoas com deficiência. No entanto, o experimento foi direcionado para pessoas com deficiência visual. Assim, para realizar uma expansão deste trabalho é necessário implementar outras regras para outros sistemas operacionais, bem como para outros tipos de deficiência.

1.3 Processo de trabalho

Para a execução deste trabalho foram definidas as seguintes fases, que serão descritas na seção 3.2:

Fase 1 – Identificação de diretrizes e recomendações de acessibilidade

Fase 2 – Abordagem para geração de componentes de interfaces gráficas para aplicativos nativos.

Fase 3 – Desenvolvimento do ambiente.

Fase 4 – Avaliação da abordagem proposta.

1.4 Estrutura do documento da dissertação

O Capítulo 1, aqui apresentado, visa oferecer ao leitor um panorama geral sobre o contexto no qual se insere este trabalho de pesquisa.

O Capítulo 2 aprofunda o referencial teórico inicial descrito no Capítulo 1, focando especialmente nos temas desenvolvimento dirigido a modelos, acessibilidade, diretrizes de acessibilidade e trabalhos relacionados.

O Capítulo 3 apresenta um posicionamento metodológico, define o tipo de pesquisa e quais são as estratégias para a sua execução.

O Capítulo 4 apresenta a abordagem proposta.

O Capítulo 5 apresenta os resultados e discussões.

O Capítulo 6 finaliza o trabalho abordando a relevância do estudo, as suas contribuições e limitações.

CAPÍTULO 2 - REVISÃO DA LITERATURA

“Eu acredito no trabalho, não dou a mínima para a sorte
(Harvey Specter)”

Neste capítulo serão apresentados os principais temas envolvidos neste projeto de pesquisa. Inicialmente discute-se o Desenvolvimento Dirigido a Modelos, na sequência são apresentados conceitos de Acessibilidade e as diretrizes de acessibilidade, com destaque para desenvolvimento acessível para aplicações móveis. Por fim, são apresentados os trabalhos relacionados que utilizam o MDD para geração de interfaces gráficas.

2.1 Desenvolvimento Dirigido a Modelos

O MDD vem ganhando cada vez mais popularidade durante os últimos anos, e um deus tópicos é o desenvolvimento de formulários WEB (EBEID; VALOV; JACOBSEN, 2016). O MDD é uma abordagem da engenharia de software em que os modelos, ao invés de programas, são as saídas principais do processo de desenvolvimento (KENT, 2002; SCHMIDT, 2006).

Para Sommerville (2011), um modelo é uma visão abstrata de uma aplicação que ignora alguns detalhes e que pode unir modelos complementares para demonstrar o contexto, interações, estrutura e comportamento da aplicação. France e Rumpe (2007), também definem modelo como uma abstração de alguns aspectos do sistema, que são criados para servir a um propósito específico de como apresentar o entendimento de uma pessoa descrevendo aspectos do sistema ou apresentar informações que podem ser analisadas por processamento computacional. Os diagramas são considerados como a materialização gráfica de modelos.

Assim, um dos principais objetivos do MDD é aumentar o nível de abstração de uma aplicação, ou seja, ao invés de pensar em código fonte, o analista passa a pensar no modelo da aplicação a ser desenvolvida.

Um outro objetivo, apontado por Vidyapeetham (2009), é compreender o uso de modelos no ciclo de vida em desenvolvimento de software e, além disto, o MDD

automatiza o desenvolvimento de software devido a três características desta abordagem: processamento dos modelos criados, transformação dos modelos para código de baixo nível e técnicas de geração de código.

Stahl, Voelter e Czarnecki (2006) elegeram cinco benefícios do MDD:

1. **Diminuir o prazo de desenvolvimento automatizando a implementação do sistema, ou aplicação:** com o aumento da abstração, usando modelos, ao invés de códigos de baixo nível, em conjunto com as regras de transformação, o prazo de desenvolvimento se reduz, diminuindo assim o custo de implementação.
2. **Software ou aplicação, com qualidade maior:** o aumento da qualidade da aplicação aumenta devido à arquitetura de software que, deverá reduzir, uniformemente, a alteração de códigos. Uma vez que o modelo é alterado, o código final é gerado por transformações.
3. **Aspectos de aplicações transversais especificadas nas regras de transformação:** estes aspectos podem ser especificados nas regras de transformação, que também podem ser aplicados para resolver erros que são apresentados nas saídas dos códigos. A separação de interesse aumenta a facilidade de manutenção do software, que é reduzido por meio da anulação de redundâncias de códigos. Esta facilidade traz como benefício um melhor gerenciamento de mudanças tecnológicas.
4. **Adaptação de modelos e tecnologias:** a facilidade de adaptação de modelos e tecnologia para vários tipos de software faz com que o nível de reusabilidade seja maior.
5. **Melhora no gerenciamento de complexidade:** por meio do aumento da abstração, o MDD melhora a capacidade de gerenciamento da complexidade.

Para melhorar a qualidade e controlar o os custos de desenvolvimento dos sistemas, o MDD defende o uso de modelos formalmente definidos como cidadãos de primeira classe durante o desenvolvimento de software, em vez de usar modelos apenas como meio informal para descrever sistemas de software ou facilitar a comunicação entre equipes. No MDD, um sistema é definido por diferentes modelos em diferentes camadas de abstração (HE; MUSSBACHER, 2016).

No entanto, além desses benefícios, o MDD tem estrutura suficiente para lidar, de maneira satisfatória, com variações de lógicas de negócio e tecnologias de baixo nível (MIZUNO; MATSUMOTO; MORI, 2010). O MDD reduz o custo do desenvolvimento e conflitos entre o design da aplicação e a implementação. Quando há necessidade de alteração em alguma parte da aplicação, basta criar um modelo com as novas especificações e executar a transformação que terá a parte do projeto alterado, evitando assim que o modelo e a implementação estejam diferentes um do outro.

Outro ponto importante apontado por Heitkötter, Majchrzak e Kuchen (2013), é que, por meio do MDD, pode-se criar aplicações prontas para o uso, por meio de modelos que são capazes de realizar transformações para gerar códigos de baixo nível. Em outras palavras, os desenvolvedores podem gerar código fonte usando modelos e gerando uma aplicação final. Assim, o MDD pode levar a um rápido ciclo de desenvolvimento.

Fernandes, Machado e Carvalho (2007), apontam quatro contribuições do MDD, quando usado no desenvolvimento de software:

1. **Produtividade:** a melhoria da produtividade dos esforços de desenvolvimento é uma das principais motivações do MDD. A produtividade é benéfica para a empresa de duas maneiras: aumento da produtividade de curto prazo e aumento da produtividade de longo prazo. A produtividade de curto prazo depende da quantidade de funções que podem ser derivadas de um artefato de um software, ou seja, quanto mais funcionalidade executável puder gerar, maior será a produtividade. A produtividade de longo prazo depende da longevidade de um artefato de software, ou seja, quanto mais tempo um artefato permanecer valioso, maior será o retorno do investimento.
2. **Conceitos de desenvolvimento próximo ao domínio:** para Selic (2008), a vantagem de utilizar o MDD é que os desenvolvedores expressam modelos usando conceitos. Esses conceitos estão fortemente atrelados ao domínio do problema. Além disso, a modelagem permite esconder e extirpar erros na passagem de um modelo abstrato para um produto, ou seja, realizar a passagem do modelo para o código de baixo nível.

3. **Automação:** uma das características do MDD é a transformação automática dos modelos de design de alto nível em sistemas em execução e a facilidade de uso e manutenção dos modelos, que são menos sensíveis a tecnologia escolhida.
4. **Captura de conhecimento específico e reutilização:** o MDD permite a captura de conhecimento especializado. Este conhecimento é alcançado por meio de funções de mapeamento que levam a informação para a transformação de um modelo para o outro, permitindo a reutilização, quando é necessário realizar alguma mudança na aplicação ou ter que implementar algo novo. Esta característica permite uma evolução independente dos modelos e conduz a uma maior longevidade dos modelos.

Essas contribuições do MDD fazem com que ele seja melhor que projetar a solução de forma visual ou codificar a solução baseando-se em requisitos funcionais. Isso ocorre porque com o MDD, os desenvolvedores são capazes de construir artefatos parciais que farão parte do produto, além de projetar uma solução específica para um determinado tipo de aplicação (VIDYAPEETHAM, 2009).

O MDD é basicamente composto por conceitos e os mais comuns entre eles estão (TUFAIL et al., 2018):

- Modelo Independente de Computação (CIM): O CIM é uma visão do sistema em que a compreensão independe do conhecimento em computação (LUCRÉDIO, 2009)
- Modelo Independente de plataforma (PIM): O PIM é uma diagrama de classe da UML, enriquecido com anotações que definem regras e camadas de persistência, que ajudam a descrever todos os aspectos do sistema, incluído as configurações e o comportamento (SILVA; ABREU, 2014 e EBEID; VALOV; JACOBSEN, 2016).
- Modelo específico de plataforma (PSM): É uma visão do sistema que considera detalhes específicos a plataforma de desenvolvimento (LUCRÉDIO, 2009).

Além do MDD, pode-se utilizar uma arquitetura, como a arquitetura dirigida a modelos (MDA, do inglês *Model-Driven Architecture*). O MDA é o componente que alavancou as pesquisas e aplicações de engenharia dirigida a modelos. O foco do

MDD está sobre o projeto e implementação, em que esta arquitetura prescreve certos tipos de modelos a serem utilizados, assim como detalha a forma como estes modelos devem ser elaborados e, também, define como deve ocorrer o relacionamento entre os tipos de modelos.

2.1.1 Arquitetura dirigida a modelos

A MDA utiliza modelos para melhorar o desempenho de planejamento, design e outros ciclos, que têm o intuito de aprimorar a qualidade e durabilidade dos produtos. Basicamente, a função do MDA é transformar um modelo visual de software em código executável, seja integral ou parcialmente, que pode variar de acordo com as necessidades da aplicação a ser criada. A abordagem MDA, além de efetuar a transformação do modelo em código, possui três vantagens importantes, que contribuem neste processo de código, sendo eles (OMG, 2000):

- Estabelece a definição de termos, ícones e notações que contribuem na compreensão;
- Disponibiliza a base para modelos de dados semânticos a serem gerenciados;
- Disponibiliza biblioteca de modelos reutilizáveis, tais como regras, modelos de objetos de negócios ou padrões de projetos de arquitetura.

Para OMG (2000), essa automatização gerada pela MDA reduz custos e o tempo da realização do sistema como um todo e, além disso, beneficia as manutenções futuras, garantindo a consistência entre o modelo e o código gerado. Assim, essa abordagem torna a geração de código mais rápida e confiável.

Segundo Stahl, Vöelter (2006), um dos conceitos da arquitetura e da engenharia orientada a modelos é a separação que existe entre modelos independente de plataforma (PIM, do inglês *Platform Independent Model)* e os modelos específicos de plataforma (PSM, do inglês *Platform Specific Model).*

Um modelo independente de plataforma representa uma visão da aplicação sem associação direta com as características da plataforma que abrigará a aplicação. Desta forma, é possível ter um modelo que seja adequado para diversas plataformas semelhantes (OMG, 2003).

Já os modelos específicos de plataforma contêm todas as informações requeridas sobre o comportamento e estrutura da aplicação, em sua plataforma

específica, mesmo que não seja executável por si só. Deve ser um modelo autocontido, de forma que os desenvolvedores possam utilizá-los para a implementação do código, ou geradores de aplicação irão conceber o código executável valendo-se de processos de transformação (BRAMBILLA; CABOT; WIMMER, 2012).

2.2 Acessibilidade

Os dispositivos móveis tornaram-se um dos meios mais utilizados para fornecer acesso às notícias, serviços e todos os tipos de informações usadas por pessoas no dia-a-dia. Assim, é importante que todas as pessoas possam usar esses serviços, inclusive pessoas com deficiência.

Pessoas com deficiência, possuem diferentes habilidades, e são prejudicadas por não poder usar estes serviços, devido as barreiras nas aplicações construídas. Assim, nesta seção, serão apresentados conceitos de acessibilidade, acessibilidade na tecnologia e a caracterização da deficiência.

2.2.1 Conceitos e definições da acessibilidade

De acordo com o Decreto n°5.296, de 2 de dezembro de 2004, a acessibilidade tem o objetivo de fornecer condição para a utilização, com segurança e autonomia, de forma total ou assistida, dos espaços, mobiliários e equipamentos urbanos, das edificações, dos serviços de transporte e dos dispositivos, sistemas e meios de comunicação e informação, por uma pessoa com deficiência ou por grupos que têm mobilidade reduzida. Este decreto também define que as barreiras que impendem a acessibilidade são quaisquer entraves ou obstáculos que limitem ou impeçam o acesso, a falta de liberdade de movimento, a falta de segurança com a circulação das pessoas e a dificuldade das pessoas se comunicarem ou terem acesso a lugares e/ou informações (BRASIL, 2004). De acordo com Artigo 3° deste decreto, serão aplicadas sanções administrativas, civis e penais cabíveis caso alguma de suas normas não forem atendidas.

Segundo Sarraf (2012), a acessibilidade é uma maneira de concepção de ambientes, que considera o uso de todos os indivíduos, independentemente de suas limitações físicas e sensoriais, desenvolvidas a partir dos conceitos de inclusão social. A acessibilidade possibilita a melhoria da qualidade de vida da população que possui,

ou não, deficiência, proporcionando liberdade de escolha e abertura de horizontes pessoais, profissionais e acadêmicos.

Assim, 9é importante se atentar com as acessibilidades que dizem respeito a tecnologia, como por exemplo a acessibilidade comunicacional, pois, atualmente vivemos uma era digital e é essencial que a tecnologia esteja acessível para todos. Para que isso aconteça é preciso que haja uma preocupação com a acessibilidade na tecnologia.

2.2.2 Acessibilidade na tecnologia

A W3C (*Word Wide Web* Consortium) é uma organização que se propõe a tornar toda a informação e serviços disponíveis na Web acessíveis ao maior número de pessoas possíveis, inclusive as pessoas com deficiências.

Assim, em parceria com outras organizações internacionais, a W3C promoveu a criação da Iniciativa da Acessibilidade na WEB (WAI, do inglês *Web Accessibility Initiative)*, que são vários documentos que contêm guias e informações de acessibilidade e que procura desenvolver a acessibilidade na *Web,* nas áreas de tecnologia, diretrizes, ferramentas, educação, pesquisa e desenvolvimento. Em 1999 foi publicado, pelos responsáveis pelos padrões mundiais relacionados com a Web, o primeiro documento sobre a acessibilidade na internet, tornando-se uma referência mundial nesse assunto. Um dos documentos, pertencentes ao WAI, chama-se *Web Content Acessibility Guidelines* 1.0, 1999 (WCAG 1.0). Este documento contém informações de como tornar o conteúdo Web acessível às pessoas com deficiência e quatorze recomendações básicas e dezenas de pontos de verificações que podem ser traduzidos em diversos idiomas (WCAG 2.0, 2008).

Alguns países europeus e os Estados Unidos, desenvolveram projetos e legislações específicos no que se refere à acessibilidade à Tecnologia de Informação e Comunicação (TIC). Porém, apenas os Estados Unidos, Canadá, Austrália e Portugal possuem regulamentação sobre acessibilidade à Web e todos adotaram a WCAG do WAI/W3C como referência. Os Estados Unidos possuem o maior número de regulamentações sobre Acessibilidade Tecnológica. A “*section 508,* Padrões de Acessibilidade para Tecnologia Eletrônica e de Informação”, emenda de 1998, da Lei Federal *Rehabilitation Act,* 1973, determina que todos os sites, softwares, sistemas operacionais, produtos de telecomunicações, quiosques eletrônicos, entre outras TIC

para uso, contratação e aquisição pelas agências governamentais, devem estar em conformidade com determinados padrões publicados na norma, que foram baseadas na WCAG 1.0 (ESTADOS UNIDOS, 2002).

Em 26 de agosto de 1999, Portugal tornou-se o primeiro país europeu e o quarto país do mundo a criar legislação sobre a adoção de regras de acessibilidade na concepção da informação disponibilizada na Internet pela administração pública, com o objetivo de facilitar o seu acesso às pessoas com deficiência. Essa regulamentação constitui a Resolução de Conselho de Ministros n° 97/99 (PORTUGAL, 1999).

No Brasil também não é diferente, uma das principais tarefas do Governo Federal é promover a inclusão social, com distribuição de renda e diminuição das desigualdades. Dentre as diversas iniciativas que visam atingir esse objetivo, o governo investe na inclusão digital, para que pessoas com deficiência tenham o uso adequado e coordenado da tecnologia (GOVERNO ELETRÔNICO, 2014).

2.2.3 Deficiência

O conceito de deficiência teve diversos tratamentos ao longo da história. Este tratamento e mudanças de definições acontece porque a deficiência não se trata de um conceito abstrato e sim de como a pessoa com deficiência é encarada e incluída na sociedade (ARAUJO; FERRAZ, 2010).

Para Montoya (2000), deficiência é a perda ou anormalidade de uma estrutura ou função psicológica, fisiológica ou anatômica; incapacidade restrição ou ausência da capacidade de realizar uma atividade na forma, ou dentro da margem que se considera normal para o ser humano; é a desvantagem definida como a situação desvantajosa que se encontra um indivíduo, em consequência de uma deficiência ou de uma incapacidade, que lhe limita e impede de executar uma série de atividades que seria considerada normal para pessoas da mesma idade, sexo e nível sociocultural.

Algumas mudanças aconteceram no decorrer dos anos e a Convenção sobre os Direitos da Pessoa com Deficiência, que é patrocinada pela Organização das Nações Unidas (ONU), aderiu a um novo conceito sobre o que é deficiência:

> A “Deficiência resulta da interação entre pessoas com deficiência e as barreiras devido às atitudes e ao ambiente que impedem a plena e efetiva participação dessas pessoas na sociedade em igualdade de oportunidades com as demais pessoas” (BRASIL, 2009).

Além disso, nesta mesma convenção, é definido que:

> “Pessoas com deficiência são aquelas que têm impedimentos de longo prazo de natureza física, mental, intelectual ou sensorial, as quais, em interação com diversas barreiras, podem obstruir sua participação plena e efetiva na sociedade em igualdade de condições com as demais pessoas” (BRASIL, 2009).

2.2.4 Diretrizes para acessibilidade

Existem várias diretrizes ou guias com recomendações para acessibilidade, incluindo a parte móvel (aplicações nativas e aplicativos web). Uma dessas iniciativas, como citado anteriormente, é a WCAG, criado pela W3C, que está trabalhando na adaptação de suas diretrizes genéricas existentes para o contexto móvel.

2.2.4.1 WCAG

A segunda versão da WCAG foi publicada como uma recomendação da W3C em dezembro de 2008. A WCAG define um conjunto de recomendações para tornar o conteúdo Web mais acessível. Essas recomendações se destinam a vários tipos de deficiência, tais como visuais, auditivas, físicas, de fala, cognitiva, de linguagem, de aprendizagem e neurológica, e também as pessoas idosas as quais adquirem algumas deficiências devido ao processo de envelhecimento (WCAG 2.0, 2008).

A elaboração da WCAG contou com a colaboração de organizações e pessoas do mundo todo, com o objetivo de fornecer um padrão comum para a acessibilidade do conteúdo Web, para atender as necessidades dos indivíduos das organizações e dos governos, à nível internacional (WCAG 2.0, 2008).

A WCAG possui dois objetivos principais: a aplicação da WCAG em diferentes tecnologias Web, a partir do W3C e de outras fontes; e a capacidade de ser testável por meio de avaliações realizadas por seres humanos, e um subconjunto das recomendações feitas por testes automáticos (ALONSO et al., 2010).

Tendo em vista sua utilização, “as pessoas e organizações que utilizam a WCAG é muito distinta e inclui desenvolvedores e Web designers, legisladores, agentes de compras, professores e alunos” (WCAG 2.0, 2008).

De forma a atender esse público, foram elaborados níveis de abordagem, que incluem (WCAG 2.0, 2008):

- **Os princípios:** contém quatro princípios que constituem a fundação da acessibilidade da Web: perceptível, operável, compreensível e robusto;
- **As diretrizes:** as 12 diretrizes fornecem os objetivos básicos que os autores devem atingir para produzir conteúdo mais acessível a utilizadores com diferentes incapacidades.
- **Os critérios de sucesso:** para cada diretriz são fornecidos critérios de sucesso testáveis de forma a permitir que a WCAG 2.0 seja usada onde os requisitos e os testes de conformidade sejam necessários, nomeadamente na especificação das concepções, nas compras, nas regulamentações e nos acordos contratuais.
- **As técnicas de tipo suficiente e aconselhada:** para cada uma das diretrizes e critérios existentes foram criadas técnicas. As técnicas têm caráter informativo e se enquadram em dois tipos: As que são de tipo suficiente satisfazem os critérios de sucesso e as que são de tipo aconselhada. As do tipo aconselhada vão muito além do que é requerido em cada um dos critérios de sucesso e permite aos autores um melhor cumprimento das diretrizes.

Assim, o documento WCAG 2.0 foi concebido para satisfazer as necessidades das pessoas que precisavam de um padrão técnico estável e passível, que possa ser utilizado como referência. Outros documentos, chamados documentos de apoio, baseiam-se no documento WCAG 2.0 e respondem a outros objetivos importantes (WCAG 2.0, 2008). Entretanto, a WCAG 2.0 não trata de acessibilidade de interfaces gráficas para dispositivos móveis.

2.2.4.2 WCAG para dispositivos móveis

Em fevereiro de 2015 foi publicado pela W3C um documento que descreve como a WCAG 2.0 e seus princípios, recomendações e critérios de sucesso podem ser aplicados nos dispositivos móveis, tanto para aplicações nativas, quanto para aplicações híbridas. A WCAG 2.0 para mobile fornece uma orientação informativa, porém, não define requisitos de acessibilidade. Ele também destaca a relevância da WCAG 2.0 no contexto móvel (W3C, 2017).

O documento faz referência as recomendações da WCAG 2.0 existentes que se aplicam à plataforma móvel e fornece novas práticas recomendadas, que podem no futuro ser recomendações da WCAG 2.0. O documento descreve, em quatro tópicos, os desafios emergentes de acessibilidade móvel (W3C, 2015):

1. **Perceptível:** a percepção dos dispositivos móveis é dividida em 3 pontos principais. O primeiro é em relação ao tamanho da tela dos dispositivos móveis, que são pequenas. O segundo está relacionado ao controle do usuário em ampliar o conteúdo que é mostrado na interface. O terceiro ponto é que os dispositivos móveis são mais propensos a serem usados em ambientes variados, incluindo ambientes externos, onde o reflexo do sol ou outras fontes de iluminação podem afetar o contraste das aplicações.
2. **Operável:** é considerado na operacionalidade dos dispositivos móveis cinco desafios: uso do teclado, toque no componente específico, gestos *touchscreen,* gestos como tremor e inclinação, e, botões com acesso fácil.
3. **Compreensível:** a compreensão está relacionada à visão da interface, contendo seis desafios: mudança da orientação da tela; layout consistente, ou seja, padronização das interfaces; posicionamento dos elementos importantes da página antes da rolagem da página; agrupamento de elementos operáveis que executam a mesma ação; fornecer uma indicação clara de que os elementos são acionáveis; fornece instruções para *touchscreen* personalizado.
4. **Robusto:** há três desafios: definir teclado virtual diferente para determinados tipos de entrada de dados; fornecer métodos fáceis para entrada de dados; e, suportar as propriedades características da plataforma.

Assim, o documento da WCAG para dispositivos móveis é usado por pessoas que presam por um padrão para seguir durante o desenvolvimento. Ao conseguir atingir os tópicos os desenvolvedores garantem a acessibilidade em suas aplicações móveis.

2.3 Criação de interfaces gráficas por meio de MDD

Foi encontrado uma revisão sistemática da literatura sobre o desenvolvimento de interfaces gráficas acessíveis para pessoas com deficiência visual e sete trabalhos que utilizam o MDD para geração de códigos por meio de modelos. Cinco destes trabalhos utilizaram o MDD para geração de interfaces gráficas acessíveis para Web, outros trabalhos utilizaram o MDD para geração de códigos multiplataforma.

Oliveira e Filgueiras (2018), apresentam em seu trabalho uma revisão sistemática de literatura, que identifica trabalhos que abordam o uso, ou a criação de ferramentas que podem apoiar o desenvolvimento de aplicativos móveis acessíveis a usuários com deficiência visual. A pesquisa mostrou que, apesar de já existirem várias ferramentas que apoiam os desenvolvedores, nenhuma é integrada às ferramentas de desenvolvimento de aplicativos móveis. Oliveira e Filgueiras (2018) também identificaram que as ferramentas não cobrem todas as fases do desenvolvimento, e a maioria dos trabalhos baseia-se nas diretrizes do WCAG 2.0.

O trabalho de Antonelli, Silva e Fortes (2015) tem o objetivo de abordar o problema relacionado com a falta de acessibilidade dos usuários, ajudando os desenvolvedores a criar menus Web com base no MDD. Neste trabalho, os autores estudaram diferentes tipos de menus e suas estruturas, bem como suas diretrizes de acessibilidade que envolvam a criação de menus acessíveis. Posteriormente, desenvolveram um Modelo Independente de Plataforma (Platform Independent Model - PIM) que originou o Domain Specific Language (DSL) chamado AMenu. Com isso, eles inseriram detalhes técnicos sobre acessibilidade nas transformações desses modelos. Os autores apresentaram uma avaliação exploratória da DSL, discutindo a eficiência e as limitações da abordagem.

Ahamed e Ashraf (2007), apresentam uma maneira de combinar várias abordagens baseadas em MDD, para construir uma estrutura para o desenvolvimento de interfaces gráficas. Além disso, os autores apresentam uma clara definição dos vários modelos que são utilizados para o desenvolvimento de interfaces e, com isto, é fornecido um esboço do processo de derivação da interface gráfica para Web. Os autores apresentam um caso para avaliar a aplicabilidade da abordagem.

No trabalho de Panach, Aquino e Pastor (2015), é apresentada uma maneira de incluir recurso de usabilidade funcional em MDD, por meio de primitivas conceituais. A usabilidade funcional é quando a usabilidade está fortemente ligada

com as funcionalidades do sistema. O trabalho foi aplicado no âmbito de aplicativos para Web. Os autores estudaram diretrizes de usabilidade para identificar propriedades de usabilidade que podem ser representadas em um modelo conceitual. Assim, essas diretrizes se tornam entradas para um compilador modelo, que gera código de acordo com as características expressas neles. Um estudo empírico com 66 indivíduos foi realizado para estudar o efeito de incluir funcionalidades relativas à satisfação e tempo dos usuários finais para completar tarefas.

Em outro trabalho, Panach et al. (2014), definem um processo para incluir o modelo de usabilidade em qualquer método MDD sem afetar o modelo conceitual existente. A proposta baseia-se em transformações de modelo para modelo e de modelo para código de interface gráfica para Web. Para validar o processo criado no trabalho, os autores aplicaram a proposta a um método MDD existente chamado OO-method e mediram a eficiência do processo.

Gamecho et al. (2014) desenvolveram um gerador automático de interfaces acessíveis, chamado Egoki, que foi projetado para permitir que as pessoas com deficiência acessem os serviços de apoio. O gerador foi elaborado com base em uma abordagem baseada em modelos. Para avaliar o gerador, um protótipo de prova de conceito foi conduzido com experiência em acessibilidade e com participantes de dois diferentes cenários: pessoas com deficiência visual e pessoas com deficiência cognitiva. Para isso, os autores criaram dois serviços para cada cenário, sendo seleção de serviço de atualização e atendimento. As interfaces gráficas geradas automaticamente foram avaliadas por especialistas em acessibilidade.

Vaupel et al. (2018), apresentam em seu trabalho uma linguagem de modelagem e uma infraestrutura de aplicativos móveis nativos para o MDD, que rodam tanto nos sistemas Android quanto em IOS. A abordagem criada pelos autores permite o desenvolvimento de aplicativos flexíveis em diferentes níveis de abstração de modelagem compacta de elementos de aplicativos padrões, como gerenciamento de dados padrão e modelagem cada vez mais detalhada de elementos individuais para cobrir, por exemplo, um comportamento específico. Além disso, a abordagem criada suporta um tipo de modelagem de variabilidade, isto é, a maneira que aplicativos móveis com variantes podem ser desenvolvidos. Para testar a abordagem, os autores utilizaram em vários aplicativos, sendo um deles um guia de museu com funcionalidade de realidade aumentada.

No trabalho de Usman, Iqbal e Khan (2017) é proposta uma nova aplicação de engenharia orientada por modelo de linhas de produtos de software, que é utilizada para o desenvolvimento de aplicações para dispositivos móveis. Neste trabalho também são abordados os principais desafios das variantes no desenvolvimento de aplicações móveis nativas, que são baseadas em recursos para múltiplas plataformas. Os autores utilizaram três tipos de variação de aplicações móveis: variação devido a sistemas de operação e suas versões, softwares e recurso de hardware de dispositivos móveis e funcionalidades oferecidas pelo aplicativo móvel. Para automatizar a proposta, os autores desenvolveram uma ferramenta, chamada MOPPET. Para validar o trabalho os autores realizaram dois casos de estudos industriais, que mostraram que a abordagem proposta é aplicável às aplicações móveis industriais. Houve a utilização do MDD para criação de aplicações multiplataforma, isto é, aplicações que rodam, por exemplo, em *Android e IOS.*

	Acessibilidade	**Diretrizes**	**Dispositivo Móvel**	**Web**	**PIM**	**PSM**
AHAMED; ASHRAF, 2007		X		X		
PANACH; AQUNO; PASTOR, 2015				X		
PANACH et al., 2014				X		
USMAN; IQBAL; KHAN, 2017			X			
VAUPEL et al., 2017			X		X	X
ANTONELLI; SILVA; FORTES, 2015	X	X		X	X	X
GAMECHO et al., 2014	X		X			

Quadro 2.1 - Resumo comparativo das características encontradas nos trabalhos relacionados. FONTE: o autor (2019)

O Quadro 2.1 apresenta um resumo comparativo entre algumas características envolvidas nos trabalhos relacionados. Como pode ser observado, três trabalhos tentaram resolver dificuldades de implementação de aplicativos móveis, como por exemplo, problemas relacionados ao desenvolvimento de aplicativos móveis multiplataformas e problemas de acessibilidade de aplicativos móveis que rodam na Web. No entanto, os trabalhos não apresentam nenhuma evidência da possibilidade de gerar interfaces gráficas para aplicativos móveis nativos. Além disso, poucos trabalhos utilizaram as diretrizes de acessibilidade para gerar interfaces gráficas, e

dentre estes trabalhos, nenhum é focado em aplicativos móveis nativos. Também é possível observar que somente dois trabalhos descrevem ter utilizado um PIM. Assim, diferente dos trabalhos apresentados, este trabalho visa fornecer uma abordagem que integre o desenvolvimento dirigido à modelos com diretrizes de acessibilidade, para gerar de forma semiautomática componentes de interfaces gráficas acessíveis para dispositivos móveis.

2.4 Considerações sobre o capítulo

Neste capítulo foram abordados conceitos relevantes ao trabalho, tais como: desenvolvimento dirigido a modelos, acessibilidade, diretrizes de acessibilidade e, por fim, os trabalhos relacionados. Foram encontrados nove trabalhos que se relacionam com a proposta deste estudo. Todos os trabalhos encontrados utilizam o MDD para geração de códigos. Pode-se observar também que há quatro trabalhos que não foram desenvolvidos para dispositivos móveis, no entanto, ambos trabalhos são para geração de interfaces gráficas para Web, sendo um deles com foco em acessibilidade.

Assim o diferencial deste trabalho é criar uma abordagem para gerar interfaces gráficas para aplicativos móveis nativos, ou seja, aplicações que rodam diretamente no dispositivo móvel. Além disto, a proposta visa gerar as interfaces gráficas acessíveis baseando-se em diretrizes de acessibilidade.

CAPÍTULO 3 - ESTRUTURAÇÃO DA PESQUISA

“Você nunca ganhará muito se ficar apenas se preocupando em minimizar as perdas (Harvey Specter)”

Neste capítulo serão apresentadas as estratégias de pesquisa a serem adotadas para atingir o objetivo geral deste estudo.

3.1 Caracterização da pesquisa

Esta é uma pesquisa exploratória (GIL, 2007) sob o ponto de vista dos objetivos, pois, pretende explorar um novo modo para geração de interfaces gráficas acessíveis para dispositivos móveis. Quanto a natureza, é caracterizada como pesquisa aplicada (GIL, 2007), pois, tem como objetivo aplicar o conhecimento para gerar interfaces gráficas acessíveis. Este trabalho foi organizado para ser operacionalizado em quatro etapas principais, descritas a seguir.

3.2 Estratégia de pesquisa

A Figura 3.1 apresenta a estrutura da pesquisa, composta por quatro fases principais: mapeamento de diretrizes de acessibilidade; proposta de abordagem para geração semiautomática de interfaces gráficas acessíveis para dispositivos móveis; desenvolvimento de ambiente para a geração de interfaces gráficas acessíveis para dispositivos móveis; e, avaliação da abordagem proposta.

Figura 3.1 - Estrutura da Pesquisa. FONTE: o autor (2019).

3.2.1 Identificação de diretrizes e recomendações de acessibilidade

Para a identificação das diretrizes de acessibilidade foi realizada uma pesquisa exploratória nas bases científicas IEEE e ACM. A busca por artigos que citassem diretrizes foi realizada utilizando os termos “accessibility” AND “guidelines” AND “mobile”.

No Quadro 3.1 é apresentado o total de artigos que retornaram em cada base científica e a quantidade de artigos que fazem menção a alguma diretriz de acessibilidade. Com a pesquisa foram identificadas quatro diretrizes (Quadro 3.2): WCAG, *Android User Interface Guidelines*, *The Accessibility Developer Checklist* e *iOS Human Interface Guidelines for Accessibility*. No entanto, somente a WCAG foi considerada, pois além de ser a mais completa e conter um documento atualizado de como utilizar as recomendações em dispositivos móveis, também foi a mais encontrada nos artigos. Um estudo completo da WCAG foi realizado para a identificação das recomendações para acessibilidade, assim como, para quais delas seriam viáveis a implementação por desenvolvedores de aplicativos.

	N. de artigos encontrados	***N. de artigos que usam alguma diretriz***
IEEE	*49*	*9*
ACM	*319*	*230*

Quadro 3.1 – Artigos que usam a WCAG

	Número de artigos
WCAG	*239*
Android User Interface Guidelines	*1*
The Accessibility Developer Checklist	*2*
iOS Human Interface Guidelines for Accessibility	*1*

Quadro 3.2 - Diretrizes encontradas

3.2.2 Abordagem para geração de componentes de interfaces gráficas para aplicativos nativos

Para a elaboração da abordagem foi necessária a criação de um PIM, que é um modelo com elementos como classes, atributos, associações e agregações. No entanto, o PIM é uma abstração da solução, que apresenta elementos e regras gerais que um modelo final deve conter, que neste trabalho significa os elementos de uma interface gráfica. A partir do PIM, um Modelo Específico para Plataforma (Platform Specific Model (PSM)) foi gerado. O PSM é um modelo que apresenta todos os elementos a serem implementados na plataforma específica da solução final, e sua criação deve seguir as regras definidas no PIM.

Para a criação, tanto do PIM quanto do PSM, pode-se encontrar ferramentas como o Eclipse Model Framework, que ajuda a criar estes itens. Contudo, esta ferramenta para a proposta apresenta dois problemas: tem seu foco em modelos e é complexa para usuários finais. Sendo assim, optou-se por desenvolver em conjunto com a abordagem proposta, um ambiente que abstrai o PIM e auxilia na criação do Modelo (PSM).

Além disso, para a criação da abordagem também é necessária a utilização de regras de transformação. Estas regras são utilizadas para descrever o “como”, isto é, como um modelo PSM pode ser transformado em código. No caso da abordagem proposta, as regras de transformação definem como o modelo PSM pode ser transformado em códigos de componentes de interfaces gráficas de aplicativos móveis nativos. Para a criação das regras de transformação, este estudo utilizou como base as recomendações de acessibilidade que foram identificadas na primeira etapa deste trabalho.

3.2.3 Desenvolvimento do ambiente

Como apresentado na seção anterior, o uso de ferramentas como Eclipse Model Framework, utilizado para criação do PSM, é complexo para usuários finais e teria impacto no uso da abordagem pelos desenvolvedores que precisariam estudar sobre modelagem e sobre as ferramentas de modelagem. Assim, neste trabalho foi desenvolvido um ambiente que abstrai o PIM, focando no desenvolvimento do modelo de forma mais simples. Dessa forma, o desenvolvedor não precisa compreender o

modelo PIM, ou aprender uma ferramenta de modelagem para criar a estrutura da interface gráfica acessível.

O ambiente possui interface gráfica visual, não sendo necessário os usuários utilizarem ferramentas de modelagem para criar o PSM e o código final.

3.2.4 Avaliação da abordagem proposta

Para a avaliação da abordagem foi utilizado o método experimental controlado, proposto por Basili (1996). Dentre os métodos controlados, optou-se pelo sintético. Neste método o pesquisador monitora um projeto em profundidade.

O experimento foi planejado de acordo com o seguinte objetivo: “desenvolver uma abordagem para geração de componentes de interfaces gráficas acessíveis para aplicativos móveis” e foi dividido em cinco etapas, descritas a seguir:

Definição das hipóteses. Para responder à questão principal desta pesquisa, a seguinte hipótese foi definida: desenvolvedores com pouco ou sem nenhum conhecimento em modelagem e implementação de interfaces, conseguem desenvolver interfaces gráficas acessíveis utilizando a abordagem.

Seleção dos Participantes. Como o objetivo da proposta é auxiliar que desenvolvedores consigam gerar interfaces gráficas acessíveis, optamos por utilizar diferentes perfis de usuários no experimento. A avaliação foi realizada com estudantes de graduação dos cursos de Sistemas de Informação e de Ciência da Computação, e de mestrado e doutorado na área de Informática. Além disso, foi investigado se a qualidade final da solução teria diferença entre desenvolvedores com conhecimento e sem conhecimento em Interface Humano Computador (IHC). Assim, foi aplicado um questionário para 33 estudantes com o objetivo de selecionar aqueles que já tinham tido alguma experiência com interfaces gráficas e/ou conhecimento em IHC. A avaliação contou com 28 participantes, sendo seis na fase do estudo piloto e 22 na avaliação final. Dos 22 participantes, somente seis tinham conhecimento em IHC e apenas um, dos 22 participantes, não tinha conhecimento em desenvolvimento de interfaces gráficas.

Descrição do Experimento. O experimento teve duas avaliações: a avaliação da abordagem proposta e do ambiente. O principal objetivo da avaliação da abordagem foi verificar se o seu uso auxilia os desenvolvedores na criação de interfaces gráficas acessíveis para aplicativos móveis nativos para o sistema

operacional Android. Assim, neste trabalho não foi necessário realizar um experimento focado para o público final da aplicação móvel.

Foi elaborado um contexto, simples, porém suficiente nesta fase do trabalho, pois tem-se a necessidade de avaliar somente se é possível ou não criar interfaces gráficas acessíveis de forma semiautomatizadas. O contexto envolve a implementação de uma aplicação de gestão de compras e reservas de voos de uma companhia aérea e deste contexto foram elaboradas 3 tarefas (Quadro 3.3). A partir dos contextos, os participantes tiveram que fazer o uso da abordagem para criar as interfaces gráficas.

	DESCRIÇÃO
TAREFA 1	O cliente seleciona uma cidade de origem, uma cidade de destino e uma data de partida para o voo. Caso o cliente selecione um voo de ida/volta, também deve selecionar a data de volta. Por fim, o cliente deve selecionar o número de passagens que deseja comprar e se deseja apenas voos diretos.
TAREFA 2	O cliente insere o número do cartão de crédito, o nome do dono do cartão, o número de segurança, e data de validade do cartão. Será informado o valor da compra para que o cliente possa selecionar se deseja, ou não, parcelar a compra.
TAREFA 3	O cliente insere o nome, CPF, RG, gênero e a data de nascimento do passageiro.

Quadro 3.3 - Tarefas implementadas pelos participantes

Os participantes receberam a descrição da abordagem, a descrição da tarefa, o termo de consentimento, e o manual do ambiente, que apesar de ser intuitivo, os participantes poderiam ter dúvidas quanto ao seu uso.

Já na avaliação da ferramenta, o objetivo principal foi verificar se o seu uso auxilia os desenvolvedores a criar as interfaces gráficas, mesmo para quem não tenha conhecimento sobre modelagem.

Para a avaliação da ferramenta, após a realização do experimento foi solicitado aos participantes que eles respondessem um questionário com perguntas referentes ao uso da ferramenta na criação de componentes de interfaces gráficas.

O experimento foi realizado com um participante de cada vez. Cada participante recebeu a descrição de uma das tarefas e o manual da ferramenta, que continha a explicação do passo a passo de como utilizá-la.

Execução do Experimento. Como dito anteriormente, o experimento foi realizado com 22 participantes que tiveram como tarefa criar uma interface gráfica contendo componentes de interfaces gráficas acessíveis.

O experimento foi realizado em um laboratório e individualmente. Cada participante teve uma hora para realizar o experimento, com intervalo de 30 minutos entre os participantes.

Durante o experimento os participantes tiveram acesso a todos os materiais que foram apresentados na etapa de descrição do experimento. Cada participante tinha uma tarefa específica. O Quadro 3.4 apresenta a distribuição de tarefas por participantes.

Tarefa	Número de alunos
TAREFA 1	6 alunos
TAREFA 2	9 alunos
TAREFA 3	7 alunos

Quadro 3.4 - Número de participantes por tarefa. FONTE: O autor (2019)

3.3 Considerações sobre o capítulo

Este capítulo apresentou a caracterização da pesquisa, a estratégia e as suas etapas. Para cada etapa detalharam-se os respectivos procedimentos e premissas para execução. As etapas definidas foram: identificação das recomendações para acessibilidade; proposta de abordagem para geração semiautomática de interfaces gráficas acessíveis para dispositivos móveis; desenvolvimento de ambiente para a geração de interfaces gráficas acessíveis para dispositivos móveis; e, avaliação da abordagem proposta. Após a definição das etapas da pesquisa, a abordagem deste trabalho foi definida e é apresentada no Capítulo 4.

CAPÍTULO 4 - ABORDAGEM PROPOSTA

“Nunca julgue um homem até estar no lugar dele (Harvey Specter)”

Este capítulo apresenta o mapeamento das diretrizes e a abordagem proposta para a geração semiautomática de componentes de interfaces gráficas acessíveis.

4.1 Identificação das Recomendações para Acessibilidade

Neste trabalho foi realizado um levantamento de recomendações de acessibilidade que são necessárias para que os componentes de interface sejam acessíveis.

Foram encontradas 26 recomendações, retiradas da WCAG, e separadas em 2 grupos: implementáveis (19 recomendações) e não implementáveis (7 recomendações). As recomendações não implementáveis são recomendações de acessibilidade que não são de responsabilidade do desenvolvedor da aplicação, mas sim do desenvolvedor do dispositivo móvel e do desenvolvedor do sistema operacional deste dispositivo.

Dentre as 20 recomendações que são implementáveis, foram selecionadas 6 para avaliar este trabalho, pois apesar da WCAG ser uma diretriz que garante designer universal, foi necessário limitar o escopo desta pesquisa devido ao tempo para a sua realização. Assim, foram selecionadas as recomendações que tinham mais similaridade com a deficiência visual.

No Quadro 4.1 são apresentadas, tanto as diretrizes selecionadas, quanto a sua respectiva categoria no documento WCAG.

4.2 Abordagem Proposta – visão geral

A abordagem proposta (Figura 4.1**Erro! Fonte de referência não encontrada.**) é derivada do conceito de MDD e é composta pelos elementos: PIM, PSM, Regras de Transformação, Gerador de Códigos e Diretrizes de Acessibilidade. Esses elementos são necessários para que seja possível transformar um modelo em componentes de interface gráfica.

Recomendação	Tarefa	Número de alunos
Campos de formulário um abaixo do outro, ao invés de ser ao lado.	Percepção	Tamanho da Tela
Contraste mínimo: ter um contraste de pelo menos 4,5 pontos por 1 (4,5:1) (ou 3:1 para textos em grande escala).	Percepção	Contraste
Ter um contraste de 7:1 (ou 4.5:1 para textos em larga escala).	Percepção	Contraste
Fornecer uma indicação clara que os elementos são acionáveis.	Compreensão	Elementos Acionáveis
Definir um teclado virtual adequado para cada tipo de entrada de dados.	Robustez	Teclado virtual
Os dispositivos móveis fornecem recursos para ajudar os usuários a interagir com o conteúdo.	Robustez	Suportar propriedades da plataforma

Quadro 4.1 - Recomendações selecionadas

Primeiramente, conforme apresentado na Figura 1, é necessário definir o PIM (1), que é uma abstração do modelo final da aplicação, e este possui a visão geral da aplicação. Em paralelo, é preciso identificar quais diretrizes (2) são utilizadas na criação dos componentes de interface gráfica.

Posteriormente, deve-se criar as regras de transformação (3), que são utilizadas para transformar um modelo em componentes de interfaces gráficas de aplicativos móveis acessíveis. No entanto, para a criação das regras de transformação esta abordagem se baseia no PIM e nas diretrizes de acessibilidade.

Após a criação do PIM, definição das diretrizes e criação das regras de transformação, é possível criar um modelo, também conhecido como PSM (4), que contém as informações dos componentes de interface gráfica da aplicação que se pretende desenvolver.

Após ter o modelo definido, deve-se acionar o Gerador de Códigos (5) que usará as regras de transformação em conjunto com o modelo para criar interfaces gráficas contendo componentes acessíveis.

Em relação aos papeis desta abordagem, o desenvolvedor da aplicação está envolvido somente na etapa (4), pois é nesta etapa que se deve informar as características que o componente deve ter. As outras etapas são transparentes ao desenvolvedor da aplicação, pois todo o restante já foi implementado anteriormente.

O PIM usa um diagrama de classes para descrever todos os aspectos de uma aplicação ou, como neste caso, de componentes de interface gráfica.

Neste trabalho foi criado um PIM para identificar quais componentes e atributos são necessários para compor uma interface gráfica. Esses componentes e atributos, posteriormente, serão tratados por meio das regras de transformação, a fim de gerar componentes acessíveis em uma interface gráfica.

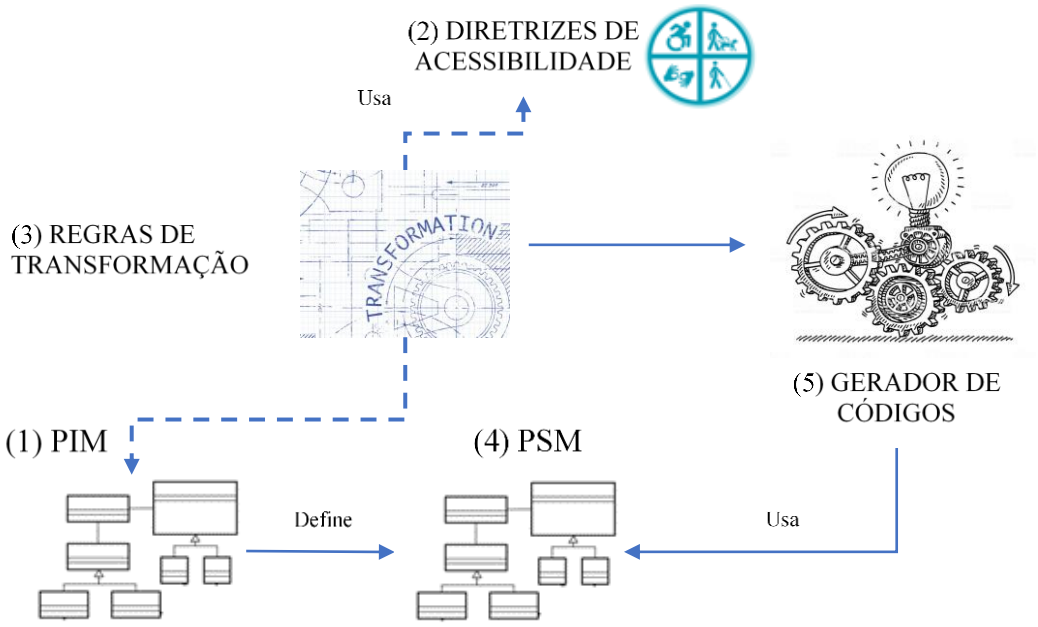


Figura 4.1 - Abordagem Proposta. FONTE: o autor (2019)

4.2.1 PIM – Platform Independent Model

A Figura 4.2 apresenta o PIM criado para validar a abordagem e também para auxiliar os desenvolvedores na padronização de um código final. O PIM desta proposta (Figura 4.2) é composto por:

- **Application**: nó inicial da construção de um modelo. A partir dele é possível configurar a criação de interfaces gráficas que farão parte de uma aplicação. Este componente possui atributos, que no caso deste trabalho, ajudarão a identificar onde os artefatos que foram gerados pelo gerador de código serão armazenados, bem como o nome da aplicação e outros atributos necessários.
- **Window**: nó que representa a interface gráfica única de uma aplicação móvel. Por meio dele é possível configurar atributos que serão usados para definir o tipo de layout da interface, o nome da interface e a orientação que os componentes ficarão dispostos na tela. Dentro da Window serão acrescentados os componentes de interface gráfica, que neste trabalho são chamados de Widget.
- **Widget**: nó final de uma interface, ou seja, é o componente que irá compor a interface gráfica. Por meio dele é possível informar o identificador daquele componente, características como margem, tamanho e alguns atributos que serão usados para fazer com que o componente seja acessível. O Widget é herdado pelos seus respectivos tipos de componentes, que podem ser: TextOutput, Button, TextInput, Checkbox, RadioGroup, RadioButton, SeekBar e Switch.

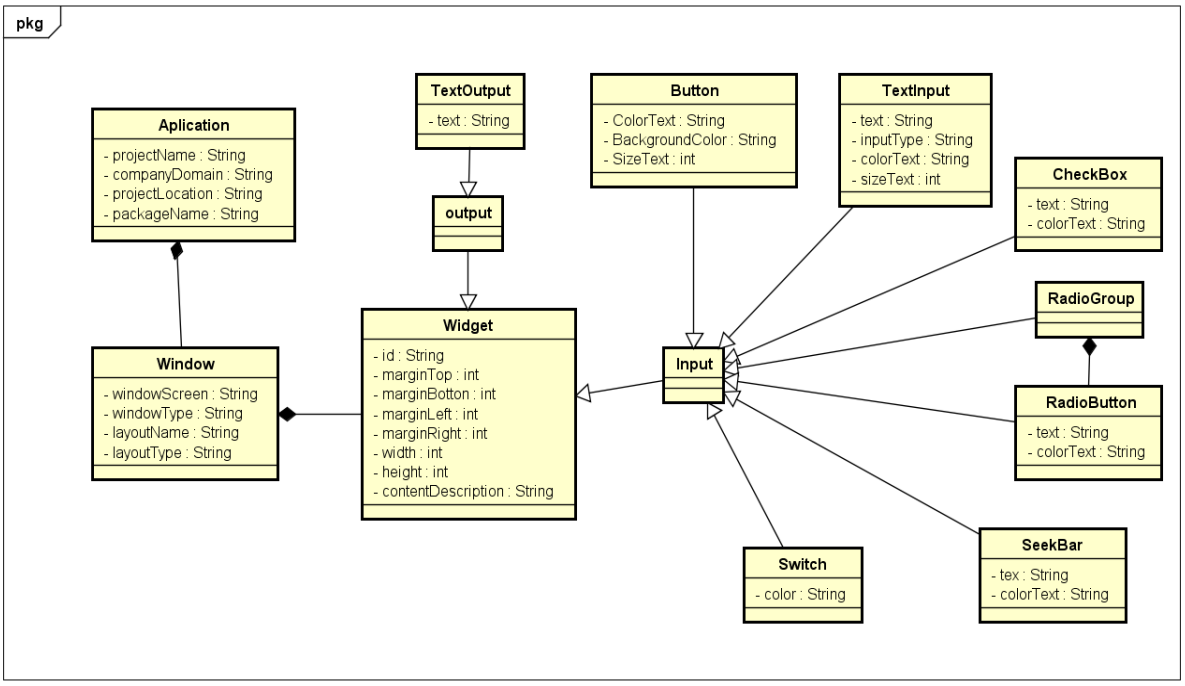


Figura 4.2 – PIM. FONTE: o autor (2019)

4.2.2 Regras de Transformação

As Regras de transformação são utilizadas para criação de códigos executáveis e criação de regras de negócios. Nas regras de transformação que os atributos colocados nos modelos são tratados, corrigindo erros nas saídas dos códigos e padronizando os códigos.

As regras de transformação atuam no PSM, pois conhecem as classes existentes, que foram geradas a partir do PIM. Com isso, as regras foram utilizadas em conjunto com o PSM e as recomendações de acessibilidade da WCAG. Essas recomendações serviram para validar e formatar os atributos dos componentes da interface gráfica. Sem o uso das recomendações seria possível criar componentes de interface gráfica, porém, esses componentes poderiam, ou não, ser acessíveis, uma vez que não teria uma validação de acessibilidade.

As recomendações da WCAG contêm regras que ao serem aplicadas fazem com que interfaces ou componentes de interfaces sejam acessíveis. Assim, foram criadas as regras de transformação respeitando as seis recomendações de acessibilidade da WCAG.

A primeira recomendação é a identificação da posição dos componentes, que determina onde os componentes devem estar posicionados na interface. O WCAG 2.0

descreve que é importante os componentes ficarem um abaixo do outro, para maior facilidade de identificação dos componentes.

A segunda e terceira recomendação, que correspondem ao contraste do componente, foram agregadas nas regras de transformação. Essas recomendações determinam o grau de luminância dos componentes, ou seja, qual é o valor de contraste que a cor de um componente tem em relação a cor de fundo.

A WCAG apresenta várias técnicas para verificar o contraste de um componente, porém, a mais usada é a de luminância relativa das cores, entre a cor de um componente e a cor que está no fundo dele. Assim, as duas recomendações de contraste utilizam a mesma técnica variando somente no valor mínimo que o resultado da luminância deve ter, que varia dependendo do tamanho do componente, principalmente pelo tamanho de uma fonte usada.

A luminância relativa é descrita pela fórmula abaixo, que foi retirada do WCAG:

$$L = 0.2126 * \boldsymbol{R} + 0.7152 * \boldsymbol{G} + 0.0722 * \boldsymbol{G} \quad (1)$$

Onde **R, G** e **B** são definidos como (WCAG 2.0):

$$se\ R_{sRGB} \leq 0.03928, então\ \boldsymbol{R} = \frac{R_{sRGB}}{12.92}\ se\ não\ \boldsymbol{R} = \left(\frac{R_{sRGB}+0.055}{1.055}\right)^{2.4}$$

$$se\ G_{sRGB} \leq 0.03928, então\ \boldsymbol{G} = \frac{G_{sRGB}}{12.92}\ se\ não\ \boldsymbol{G} = \left(\frac{G_{sRGB}+0.055}{1.055}\right)^{2.4}$$

$$se\ B_{sRGB} \leq 0.03928, então\ \boldsymbol{B} = \frac{B_{sRGB}}{12.92}\ se\ não\ \boldsymbol{B} = \left(\frac{B_{sRGB}+0.055}{1.055}\right)^{2.4}$$

Onde R_{sRGB},G_{sRGB} e B_{sRGB} são definidos como:

$$R_{sRGB} = \frac{R_{8bit}}{255}, G_{sRGB} = \frac{G_{8bit}}{255}\ e\ B_{sRGB} = \frac{G_{8bit}}{255}$$

Os valores R8bit, G8bit e B8bit, são obtidos por meio dos números RGB que uma cor representa.

A quarta recomendação, utilizada para criar as regras de transformação, é em relação a aparência e a posição dos componentes na interface gráfica. A WCAG diz que todos os componentes que podem receber interação do usuário devem ter o mesmo formato e, além disso, os componentes deverão estar um abaixo do outro.

A quinta recomendação está relacionada ao teclado que aparece para o usuário quando ele deseja inserir alguma informação. Ao inserir o tipo correto de entrada de

dados, o sistema operacional do dispositivo móvel consegue adequar o teclado para que seja exibido ao usuário. Um exemplo disso é que, se o campo tiver somente entrada de dados no valor numérico, não há necessidade de exibir um teclado com alfanumérico, garantindo assim que o teclado seja exibido de forma correta.

Por fim, a última recomendação é de que todos os componentes da interface devem ser configurados para serem lidos por aplicativos de leitura de tela. Normalmente, os sistemas operacionais possuem funções específicas para pessoas com deficiência visual. Uma dessas funções, por exemplo, é usada para realizar a leitura de componentes que estão dispostos na interface gráfica. No entanto, para que funcione 100% é preciso que os componentes sejam configurados para serem lidos por esta funcionalidade.

As regras de transformação geram códigos de interfaces gráficas para dispositivos móveis para o sistema operacional Android. No entanto, é possível criar regras para que sejam gerados códigos para vários outros sistemas, como iOS, Web e outros dispositivos móveis que possuam interfaces gráficas.

4.2.3 Ferramenta para geração do PSM

As ferramentas existentes, que auxiliam no processo do MDD, são de difícil uso por programadores que não tenham conhecimento específico para sua utilização. Assim sendo, mesmo podendo avaliar a abordagem com as ferramentas pré-existentes, a adoção de alguma delas teria impacto no uso da abordagem, pois, além do desenvolvedor ter que entender o funcionamento da abordagem, seria necessário treinar os desenvolvedores para utilização da ferramenta e, além disso, os desenvolvedores necessitariam ter conhecimento sobre modelagem.

Por esta razão, neste trabalho optou-se pelo desenvolvimento de um ambiente para apoiar a criação das interfaces gráficas. A ideia principal é abstrair o PIM que foi construído e tornar a geração do modelo da aplicação mais fácil para os desenvolvedores de software.

A execução do ambiente é guiada pelo PIM, e com isto, um PSM é gerado, contendo todos os componentes e atributos definidos no PIM. Como exemplo, a Figura 4.3 apresenta a representação da classe Application no PIM. Nesta tela, o desenvolvedor da aplicação deverá informar todos os campos que futuramente serão usados nas regras de transformação.

As validações das regras de transformação também foram aplicadas no ambiente. Dessa forma, caso o desenvolvedor não utilize, ou insira uma informação importante para gerar a interface, ao tentar avançar ou finalizar a criação de um componente, será exibida uma mensagem de erro, impossibilitando a criação da interface. A Figura 4.4 apresenta um exemplo de exibição de um erro após o desenvolvedor tentar criar uma interface onde os valores do contentDescription não foram configurados, campo usado para fazer a leitura de tela.

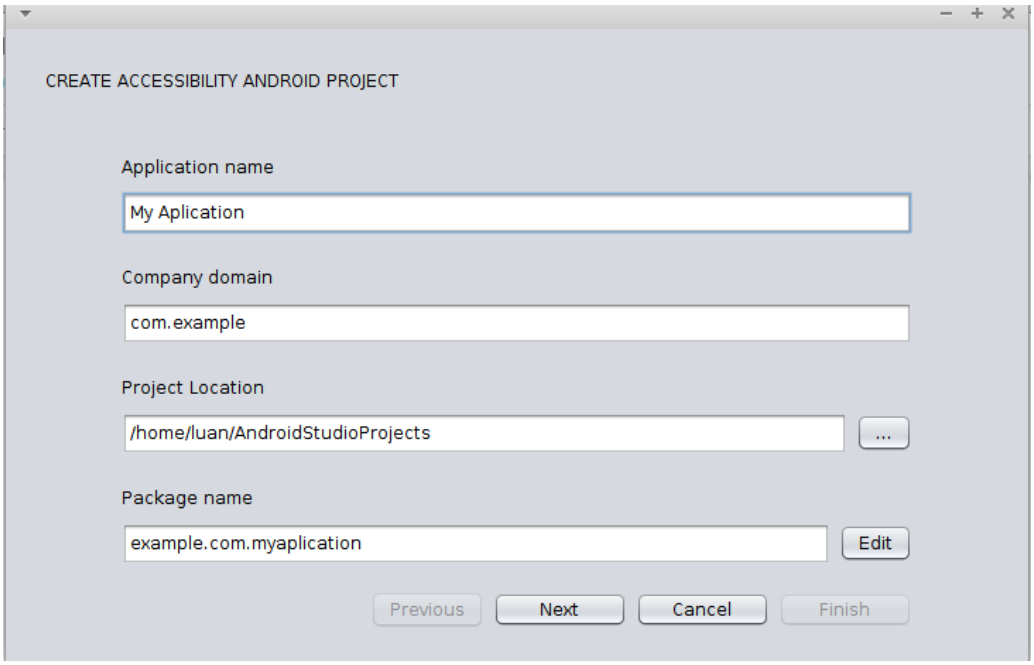


Figura 4.3 - Demonstração da classe *Application* . FONTE: O autor (2019)

Por fim, após o ambiente implementado, com todas as telas auxiliando a definir diferentes componentes descritos no PIM, é possível gerar o projeto da aplicação, que contém todo o código necessário para a aplicação acessível ser executada em um dispositivo móvel que utilize sistema operacional Android.

A geração do código é realizada por meio das regras de transformação. A Figura 4.5 apresenta parte do código de uma interface gráfica gerada pela abordagem, e a Figura 4.6 apresenta um exemplo de uma interface gráfica gerada usando as regras de transformação.

Por fim, é importante ressaltar que as regras de transformação criadas neste trabalho não se preocupam com os aspectos estéticos da interface gráfica, mas sim com a acessibilidade dos componentes desta interface.

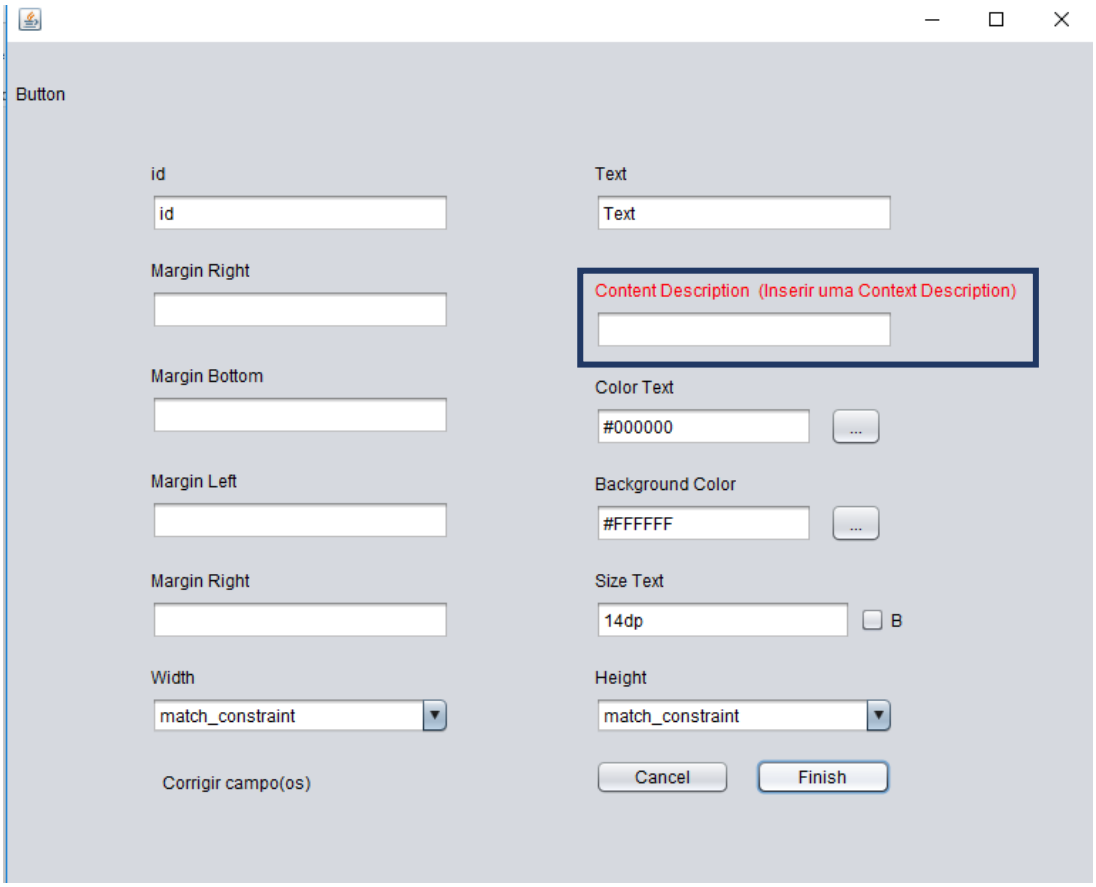


Figura 4.4 - Validação com erro. FONTE: O autor (2019)

```
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.andrc
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TelaInicial">
<android.support.design.widget.TextInputLayout
        android:id="@+id/nome_passageiro_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:layout_editor_absoluteX="0dp"
        tools:layout_editor_absoluteY="0dp">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/nome_passageiro_idtet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="Digite o nome do passageiro"
        android:hint="Nome Passageiro"
        android:inputType="text"
        android:text="Nome Passageiro"
        android:textColor="#000000"
        android:textSize="14sp" />
    </android.support.design.widget.TextInputLayout>


<android.support.design.widget.TextInputLayout
        android:id="@+id/cpf_passageiro"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

Figura 4.5 - Exemplo de código de interface gerado. FONTE: O autor (2019)

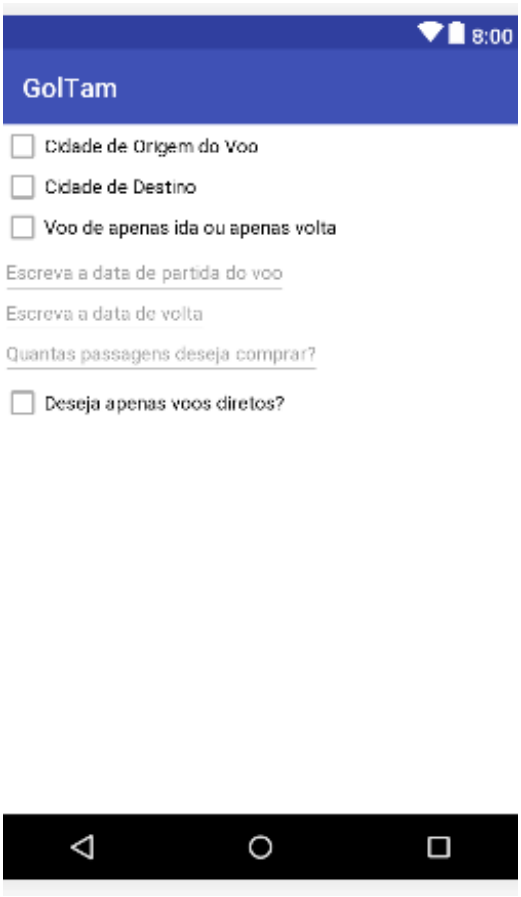


Figura 4.6 - Interface gerada pela ferramenta. FONTE: O autor (2019)

4.3 Considerações sobre o capítulo

Neste capítulo foi apresentada a abordagem que foi desenvolvida, detalhando cada um dos seus componentes: PIM, o gerador de código e o papel do desenvolvedor da aplicação, que é responsável pelo uso da abordagem.

CAPÍTULO 5 - AVALIAÇÃO DA ABORDAGEM

"O único lugar que o sucesso vem antes do trabalho é no dicionário (Harvey Specter)"

Neste capítulo é apresentada a avaliação e são discutidos os resultados da abordagem, a partir do experimento realizado.

5.1 Perfil dos Participantes

Conforme explicado na seção **Erro! Fonte de referência não encontrada.**, foi aplicado um questionário para identificar o perfil dos participantes, o nível do conhecimento sobre a criação de interfaces gráficas e/ou conhecimento de IHC (Interface Humano-computador).

Como é possível visualizar na Figura 5.1, um participante (4%) possuía somente graduação, três (14%) estavam em alguma pós-graduação e vinte e quatro (82%) ainda estavam cursando alguma graduação. Entre os participantes que estavam cursando alguma graduação, como mostrado na Figura 5.2, oito (44%) estavam no primeiro ano, um (6%) estava no segundo ano, cinco (28%) estavam no terceiro ano e quatro (22%) estavam no quarto ano.

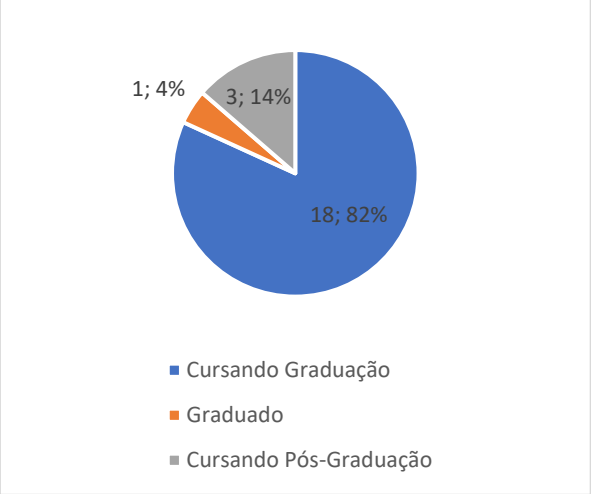


Figura 5.1 - Nível de Escolaridade

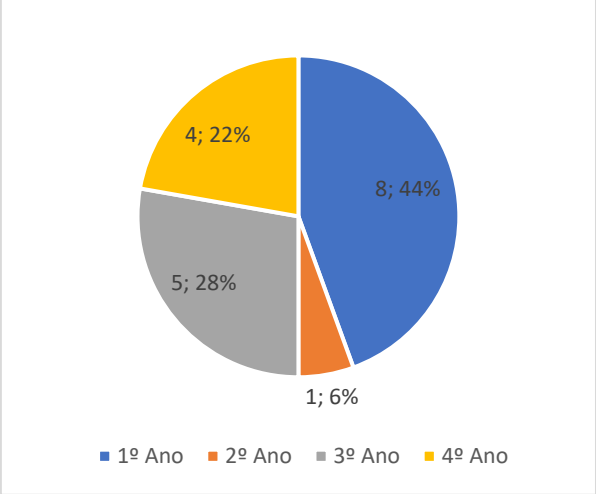


Figura 5.2 - Período dos alunos de Graduação

A Figura 5.3 mostra que seis (27%) participantes já tinham cursado a disciplina de IHC, sendo que quatro destes participantes já haviam finalizado a graduação e quatro estavam no último ano da graduação. A Figura 5.4, mostra quantos participantes tiveram contato com o desenvolvimento de interfaces gráficas, um (4%) respondeu que não havia desenvolvido nenhuma interface, três

14% ficaram com dúvidas e dezoito (82%) responderam que já haviam desenvolvido alguma interface gráfica.

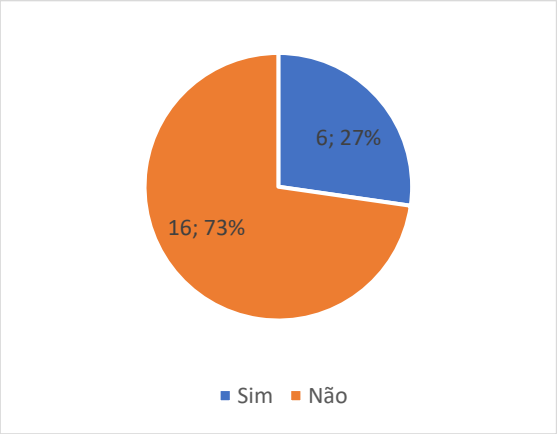


Figura 5.3 - Participantes que tiveram a disciplina de IHC

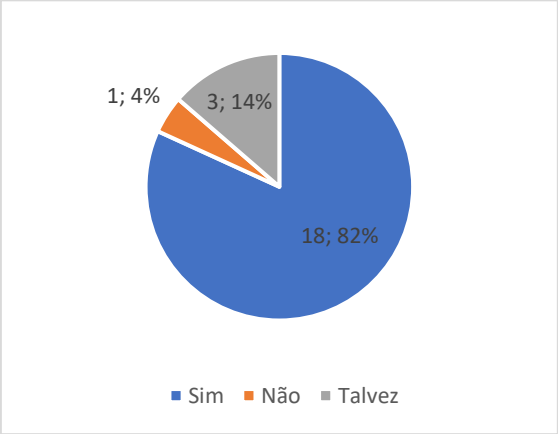


Figura 5.4 - Participantes que já desenvolveram alguma interface gráfica

5.2 Acessibilidade das aplicações

Nesta seção, os principais resultados após a análise das interfaces gráficas geradas no experimento são apresentados. A avaliação da acessibilidade foi realizada através da verificação de cada componente implementado, isto é, para cada componente utilizado nas interfaces foi analisado se ele atendia as recomendações da diretriz de acessibilidade.

Antes da análise dos componentes utilizados é importante ressaltar que cada participante desenvolveu um projeto e, com isso, foram criados 22 projetos. No entanto, dois desses 22 projetos foram descartados, pois os participantes não colocaram nada na interface e concluíram que haviam terminado. Antes destes participantes entregarem o experimento foi perguntado se realmente eles tinham terminado e foi perguntado se os participantes não iriam colocar nada na interface, e a resposta foi que eles já haviam terminado.

i) Recomendação de contraste

A primeira recomendação analisada foi a de contraste. O objetivo desta análise era verificar se os componentes das interfaces desenvolvidas pelos participantes tinham um contraste adequado (maior que 4.5), de acordo com a norma da WCAG. Para cada componente inserido na interface foi realizado o cálculo de luminância relativa[1]. Para isso, foram coletadas as cores de fundo e do texto de cada componente. Apenas dois, dos vinte e dois projetos, não puderam ser analisados, pois os participantes não chegaram a implementar uma interface. Eles abriram a ferramenta, utilizaram, porém, não colocaram nenhum componente e concluíram que haviam feito o que foi solicitado.

Foram analisados 125 componentes, comparando cada um deles com as recomendações de acessibilidade. Avaliando o resultado do cálculo de luminância relativa de cada componente foi possível verificar que todos os componentes atendiam a norma da WCAG sobre contraste.

A Figura 5.5 mostra o exemplo de um código que foi gerado pela ferramenta. É possível observar que o atributo *textColor* (A), que identifica a cor do texto, está com o valor “#000000” que representa a cor preta e o atributo *background_material_light*, que identifica a cor de fundo, está com o valor “#ffffff” que representa a cor branca (B). Assim, o valor de luminância relativa neste caso é de 21 pontos, ou seja, acima da recomendação das diretrizes de acessibilidade que é de 4,5 pontos.

(A) (B)

Figura 5.5 – Cor do Texto (A) comparado com a cor de fundo (B)

ii) Recomendação de entrada de dados

A segunda recomendação analisada foi a do tipo de teclado virtual. O objetivo desta análise foi verificar se os componentes possuíam uma entrada adequada para cada tipo de dado.

Do mesmo modo que a diretriz de contraste, foi necessário analisar cada interface gráfica. Para avaliar a interface foi necessário instalar cada uma delas no

[1] Para realizar o cálculo de luminância relativa foram coletadas as cores de fundo e a cor do texto de cada componente. Com estas cores utilizou a formula que foi apresentada no capitulo 3.

celular. Após a sua instalação, com o aplicativo aberto, o autor posicionava o foco em cada um dos componentes de entrada de dados para verificar se ele exibia o teclado virtual adequado para cada tipo de dado.

Dos 125 componentes, 85 eram de entrada de dados e destes foi observado que 60 exibiam um teclado virtual adequado (Figura 5.6). Os outros 25 componentes foram implementados de maneira incorreta, pois, teriam que ser definidos com um tipo de entrada numérico ou data, no entanto, os desenvolvedores definiam como entrada de texto. A Figura 5.7 apresenta a parte de um código que foi gerado. O código apresenta que o tipo de dados de entrada é do tipo texto.

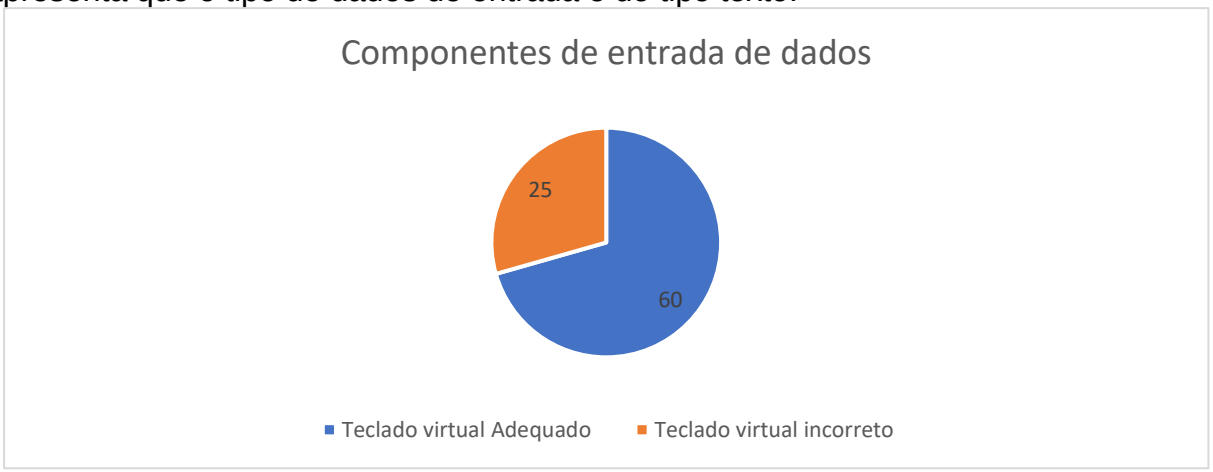


Figura 5.6 - Componentes de entrada de dados implemetnados de forma correta

```
android:inputType="text"
```

Figura 5.7 - Tipo de entrada de texto gerado pela ferramenta. FONTE: O Autor (2019)

Isto aconteceu porque, apesar de ser um campo obrigatório de entrada, a ferramenta pré-definia este campo como sendo do tipo *text,* e assim, os participantes não se preocupavam em trocar o tipo de entrada ocasionando o erro da ferramenta. Uma solução para resolver este problema é não deixar esta opção pré-definida e caso o usuário venha a deixar de selecionar esta opção a ferramenta exibirá uma mensagem dizendo que o campo é obrigatório.

A terceira diretriz analisada foi a de leitura de tela. O objetivo desta análise era verificar se os componentes de interface gráfica estavam todos configurados para que o leitor de tela do sistema operacional conseguisse utilizar o aplicativo gerado pela abordagem.

Do mesmo modo que a diretriz de entrada de dados, foi necessário instalar cada um dos aplicativos gerados no celular e, com o aplicativo aberto e com o leitor de tela do sistema operacional ativado, o autor posicionava o foco em cada um dos componentes e observava se ao realizar este passo o leitor de tela realizava a leitura do componente. Todos os 125 componentes implementados estavam configurados de forma correta e foram lidos pelo sistema operacional.

Também foi analisado o conteúdo que os participantes colocavam no atributo de leitura e, apesar da ferramenta não validar a semântica do que era escrito, pode-se afirmar que todos os participantes utilizaram o componente. No entanto, neste primeiro momento, não foi possível validar se o conteúdo semântico do atributo de leitura era válido, pois, é necessário um estudo mais aprofundado sobre como validar se o texto que foi digitado no componente é um texto válido, e se ele realmente faz sentido no contexto da aplicação.

Nesta etapa foi o uso dos componentes corretos para o que era solicitado na tarefa apresentada. O Quadro 5.1 apresenta os componentes corretos para cada interface e no Quadro 5.2, contém se os participantes incluíram corretamente ou não o componente. Os componentes que possuem um x, significa que não foi implementado.

Tarefa 1		Tarefa 2		Tarefa 3	
Descrição	**Componente**	Descrição	**Componente**	Descrição	**Componente**
Seleção de partida do voo	**Combo box ou Input Text**	Número do cartão	**Input Text**	Nome	**Input Text**
Seleção de destino do voo	**Combo box ou Input Text**	Nome do cartão	**Input Text**	Cpf	**Input Text**
Inserção da data de partida	**Input Text**	Número de segurança	**Input Text**	rg	**Input Text**
Inserção da data de retorno	**Check box e/ou Input Text**	Data de Validade	**Input Text**	genero	**Input Text**
Número de participantes	**Input Text**	Valor da compra	**Output Text**	Data de nascimento do passageiro	**Input Text**
Se deseja voo direto	**Check box**	Quantidade de parcelas	**Input Text**		**Botão**
	Botão		**Botão**		

Quadro 5.1 - Tipo correto de componente a ser utilizado nas implementações.

FONTE: O autor (2019)

Participante	**Qual problema você resolveu?**	n tela	n do componente	**componente correto**
1	Problema 1	1	1	Não
			2	Não
			3	Sim
			4	Sim
			5	Sim
			6	Sim
			7	Sim
2	Problema 2	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Não
			6	Sim
			b	x
3	Problema 2	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Não
			6	Sim
			b	x
4	Problema 1	ERRO		
5	Problema 1	1	1	Não
			2	Não
			3	Não
			4	Sim
			5	Sim
			6	Não
			7	Sim
6	Problema 2	1	1	Sim

Participante	**Qual problema você resolveu?**	n tela	n do componente	**componente correto**
			2	Sim
			3	Sim
			4	Sim
			5	Não
			6	Sim
	Problema 3	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
8	Problema 3	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			b	x
9	Problema 2	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Não
			6	Sim
10	Problema 3	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			6	Sim
11	Problema 2	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			6	Sim

Participante	**Qual problema você resolveu?**	n tela	n do componente	**componente correto**
12	Problema 1	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			6	Sim
			b	Sim
13	Problema 2	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Não
			6	Sim
14	Problema 3	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			6	Sim
15	Problema 2	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			6	Sim
16	Problema 1	1	1	Sim
			2	Não
			3	Sim
			4	Não
			5	Sim
			6	Sim
			b	Sim
17	Problema 2	1	1	Sim
			2	Sim

Participante	**Qual problema você resolveu?**	n tela	n do componente	**componente correto**
			3	Sim
			4	Sim
			5	Sim
			6	Sim
18	Problema 1	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			6	Sim
			7	Sim
19	Problema 3	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
20	Problema 2	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			6	Sim
21	Problema 1	ERRO		
22	Problema 3	1	1	Sim
			2	Sim
			3	Sim
			4	Sim
			5	Sim
			b	Sim

Quadro 5.2 - Implementações corretas e incorretas. FONTE: o autor (2019)

Dos 20 projetos, 125 componentes foram analisados. Destes, 109 foram utilizados conforme o Quadro 5.1. Os outros 13 componentes foram implementados errados, como por exemplo: ao invés de *input text*, os participantes colocaram check

box; alguns componentes já tinham componentes de output Text integrado, e mesmo assim os participantes colocaram outro componente. Problemas como inserir um componente a mais de Output Text aconteceu, pois, o manual não apresentava esta informação.

E, por fim, também foi verificado que dos 20 projetos analisados, apenas 6 possuíam um botão de navegação. Estes componentes foram identificados com a letra b, na coluna “n do componente” no Quadro 5.2.

5.3 Discussão

Durante o experimento o pesquisador atuou como observador, sem interferir na atividade, realizando anotações sobre o comportamento dos participantes.

Com base nos dados coletados no experimento é possível afirmar que a hipótese “desenvolvedores conseguem, utilizando a abordagem, criar componentes de interfaces gráficas acessíveis, mesmo como pouco conhecimento em modelagem e criação de interfaces gráficas” foi confirmada. Alguns benefícios da abordagem são:

- Capacidade do transformador de código garantir as saídas adequadas de uma aplicação: o contraste foi garantido em todos os componentes e todos os componentes estavam com os seus atributos de leitura de tela configurados;
- Acessibilidade dos aplicativos: apesar do experimento não focar na estética da aplicação, ao utilizar a abordagem é possível garantir que as recomendações de acessibilidade estarão implementadas nos componentes de interface gráfica e, assim, os desenvolvedores não terão a necessidade de ter que se dedicar a esta etapa em um desenvolvimento de aplicação, aumentando a garantia de que pessoas com deficiência não serão excluídas;
- Multiplataforma: a abordagem proposta pode ser utilizada tanto para dispositivos móveis nativos, quanto para aplicativos que rodam na Web, dependendo apenas do PIM e das regras de transformação.

Os resultados obtidos indicam que a abordagem apoia o desenvolvimento de interfaces gráficas acessíveis para aplicativos móveis, uma vez que todos os participantes conseguiram criar interfaces gráficas semelhantes, tendo o mesmo resultado em relação ao contraste e componentes de leitura de tela. Com isto, pode-se concluir que o MDD além de apoiar os programadores no desenvolvimento de aplicativos acessíveis, também apoia a padronização de códigos, pois, por meio do

PIM, é garantido que todas as soluções têm as classes que respeitam o padrão estabelecido.

No entanto, o uso do PIM em outras soluções que usam o MDD é de difícil uso, pois os usuários precisam entender o modelo descrito no PIM, e abstraí-lo para definir um modelo PSM para sua solução. Esta implementação, geralmente, se faz no nível do modelo. Na solução proposta, a criação de um PSM a partir de um PIM é feita de forma transparente, por meio de uma ferramenta de interfaces gráficas. Assim, tem-se o benefício do uso do MDD, sem necessariamente ser especialista em modelos.

Além de interfaces que tenham classes e regras definidas no padrão estabelecido pelo PIM, as regras de transformação garantem que as diretrizes de acessibilidade sejam aplicadas ao código final. Estas regras são aplicadas também de forma automática pela ferramenta desenvolvida, tirando a responsabilidade da geração dessas diretrizes do programador.

Contudo, o PIM desenvolvido neste trabalho ainda se encontra em fase inicial e em trabalhos futuros poderia ser definido um PIM que suporte mais componentes e diferentes tipos de telas. O mesmo se aplica às regras de transformação para acessibilidade, que podem ser estendidas para suportar outras diretrizes de acessibilidade, como adicionar atalhos via teclado externo; criar componentes de ajuda e interfaces de ajuda automaticamente; garantir o tamanho mínimo dos componentes; criar o layout automaticamente para ambas as orientações de tela (retrato e paisagem); componentes repetidos sempre no mesmo local; e agrupar componentes que podem ser operacionalizados.

Também é possível criar regras de transformação para criar interfaces gráficas completas, não somente os componentes de interface.

Em relação aos trabalhos relacionados, não foi possível realizar comparações porque dos sete trabalhos relacionados, apenas dois deles fazem menção às diretrizes de acessibilidade e destes dois nenhum apresenta uma proposta para dispositivos móveis. Além disso, dos sete trabalhos encontrados, somente três deles são voltados para dispositivos móveis e desses, nenhum é voltado para aplicativos móveis nativos de dispositivos móveis.

A avaliação do ambiente foi realizada por meio de um questionário que foi aplicado aos 22 participantes que realizaram o experimento. Foi questionado aos

participantes se eles tiveram algum problema com o uso da ferramenta e se sim, quais eram estes problemas.

Dos 22 participantes, 7 responderam que tiveram problemas ao usar a ferramenta. O problema relatado não foi em relação ao seu uso, mas em relação à uma falha que a ferramenta apresentou de duplicidade de componentes, quando a necessidade era somente de alterar um componente, e assim o participante teve que remover um dos componentes.

Um dos participantes apresentou uma sugestão para trabalhos futuros que é a alteração dos termos nas legendas dos campos da ferramenta, para que o usuário não precise usar o manual o tempo todo. O ambiente se mostrou adequado para auxiliar os desenvolvedores na criação de interfaces gráficas acessíveis sem a necessidade que os usuários conheçam de ferramentas de modelagem.

5.4 Considerações sobre o capítulo

Este capítulo apresentou análise dos resultados obtidos com o experimento, concluindo-se que a hipótese H1 é verdadeira, ou seja, a abordagem baseada em MDD pode auxiliar no desenvolvimento de interfaces gráficas acessíveis para aplicações móveis.

CAPÍTULO 6 - CONSIDERAÇÕES FINAIS

"A vida tem duas regras: (1) nunca desista; (2) não esqueça da regra número um (Harvey Specter)"

Este capítulo apresenta as considerações finais deste trabalho, incluindo questões pertinentes à relevância, contribuição, limitações da pesquisa e trabalhos futuros.

6.1 Relevância do estudo

Mesmo com iniciativas para apoiar o desenvolvimento de interfaces gráficas acessíveis, os desenvolvedores, em sua maioria, ainda não desenvolvem as interfaces gráficas acessíveis, fazendo com que pessoas com deficiência deixem de usar esses aplicativos. Visando minimizar este problema, este trabalho propôs uma abordagem para a criação de componentes de interfaces gráficas acessíveis para aplicativos móveis.

6.2 Contribuições da pesquisa

Este trabalho possui três contribuições principais:

- Abordagem para geração de interfaces gráficas acessíveis para dispositivos móveis, podendo ser adaptada a outros tipos de dispositivos, desde que seja criado um PIM e suas respectivas regras de transformação.
- Um ambiente para facilitar a criação de aplicativos e/ou interfaces gráficas sem a necessidade de os desenvolvedores entenderem de ferramentas de modelagem para MDD.

6.3 Limitações da Pesquisa

A principal limitação deste trabalho está relacionada a confiabilidade do experimento. Devido à dificuldade de se ter um envolvimento de especialistas em desenvolvimento de interfaces gráficas, o experimento foi realizado com alunos de

graduação. Assim, não se pode dizer se a abordagem, ao ser utilizada por especialistas, também ajuda no desenvolvimento de interfaces gráficas acessíveis.

Outra limitação encontrada é que a ferramenta foi desenvolvida para criar interfaces gráficas para o sistema operacional *android.*

6.4 Trabalhos Futuros

Algumas perspectivas de trabalhos futuros são:

- Validar a abordagem com profissionais que já atuam na área de desenvolvimento de software.
- Implementar outras recomendações de acessibilidade que foram identificadas neste trabalho. Também se faz necessário criar um estudo para adicionar usabilidade nas interfaces criadas.
- Desenvolver um estudo para poder verificar e corrigir a acessibilidade de aplicativos que foram desenvolvidos sem acessibilidade.
- Outro ponto a melhorar em trabalhos futuros é que o ambiente desenvolvido também precisaria evoluir para ser possível criar interfaces gráficas por meio de “clica e arrasta”, fazendo com que os desenvolvedores consigam criar essas interfaces de modo visual. Também se faz necessário realizar mais experimentos, incluindo especialistas em IHC ou em desenvolvimento de interfaces gráficas acessíveis.

REFERÊNCIAS BIBLIOGRÁFICAS

(AHAMED; ASHRAF, 2007) AHAMED, S.; ASHRAF, G. **Model-based user interface engineering with design patterns**. The Journal of Systems and Software. v. 80, p 1408-1422, Dezembro 2006.

(ALONSO. et al., 2010) ALONSO, F.; FUERTES, J. L.; GONZÁLES, Á. L.; MARTÍNEZ, L. **On the testability of WCAG 2.0 for beginners**. Proceedings of the 2010 Internacional Cross Disciplinary Conference on Web Accessibility (W4A), Raleigh, Carolina do Norte, abr. 2010.

(ANTONELLI; SILVA; FORTES, 2015) ANTONELLI, H. L.; SILVA, E. A. N.; FORTES, R. P. M. **A model-driven development for creating accessible web menus.** Procedia Computer Science, v. 67, p. 95-104, 2015.

(ARAUJO; FERRAZ, 2010) ARAUJO, E. A. B. S.; FERRAZ, F. B. **O conceito de pessoa com deficiência e seu impacto nas ações afirmativas brasileiras no mercado de trabalho.** XIX Encontro Nacional do CONPEDI. Fortaleza – CE, 2010

(BASILI, 1996) BASILI V. **The Role of Experimentation in Software Engineering**: Past, Present, Future. IN : PROC. OF THE 18TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 1(2), 1996, p. 133-164.

(BRAMBILLA; CABOT; WIMMER, 2012) BRAMBILLA, M; CABOT, JORDI; WIMMER, M. **Model-Driven Software Engineering in Practice**. Morgan & Claypool, 2012. 165p.

(BRASIL, 2004) BRASIL. Decreto nº 5.292, de 27 de Agosto de 1904. **Regulamenta as Leis nos 10.048, de 8 de novembro de 2000, que dá prioridade de atendimento às pessoas que especifica, e 10.098, de 19 de dezembro de 2000, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas portadoras de deficiência ou com mobilidade reduzida, e dá outras providências.** Diário Oficial da União, Brasília, DF, 31 dez. 2004. Seção 1, p. 5.

(BRASIL, 2009) BRASIL. Decreto nº 6.949, de 25 de agosto de 2009. **Promulga a Convenção Internacional sobre os Direitos das Pessoas com Deficiência e seu Protocolo Facultativo, assinados em Nova York, em 30 de março de 2007.** Diário Oficial da União, Brasília, DF, 26 ago. 2009. Seção 1, p. 3.

(CISCO, 2016) Cisco 2016. **Global Mobile Data Traffic** Forecast Update, 2015–2020 White Paper. (2016). http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

(DIAZ-BOSSINI; MORENO, 2013) DIAZ-BOSSINI, J.M; MORENO, L. **Accessibility to mobile interfaces for older people.** 5th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion – DSAI 2015).

(EBEID; VALOV; JACOBSEN, 2016) EBEID, E.; VALOV, M.; JACOBSEN, R. H. **Model-Driven Design Approach for Building Smart Grid Applications.** In Euromicro Conference on Digital System Design, Limassol, 2016.

(ESTADOS UNIDOS, 2002) ESTADOS UNIDOS. **Section 508 of the Reabilitation Act.** August 7, 1998. The law requires access to electronic and information technology provided the Federal government. Washington, D, 2002. Disponível em: <http://http://www.section508.gov/>. Acesso em 06 mar. 2002

(FERNANDES; MACHADO; CARVALHO, 2007) FERNANDES, J. E. M.; MACHADO, R. J.; CARVALHO, J. A. **Model-Driven Software Development for Pervasive Information Systems Implementation**. In Proceedings of the 6th International Conference on the Quality of Information and Communications Technology, IEEE, 2007, p. 218-222.

(FRANCE; RUMPE, 2007) FRANCE, R. B.; RUMPE, B. **Model-driven Development of Complex Software**: A Research Roadmap. In: IEEE Future of Software Engineering, 2007. FOSE '07, 37–54. doi:10.1109/FOSE.2007.14

(GAMECHO et al., 2015) GAMECHO, B.; MIÑÓN, R.; AIZPURUA, A.; CEARRETA, I.; ARRUE, M.; GARAY-VITORIA, N.; ABASCAL, J. **Automatic Generation of Tailored Accessible User Interfaces for Ubiquitous Services.** III Transactions on Human-Machine Systems, v. 45, p. 612-623, 2015.

(GARTNER, 2017) Gartner. 2017. **Gartner Says Worldwide** Sales of Smartphones Recorded First Ever Decline During the Fourth Quarterof 2017. (2017). https://www.gartner. com/newsroom/id/3859963

(GIL, 2007) GIL, A. C. Como elaborar projetos de pesquisa. 4. ed. São Paulo: Atlas, 2007.

(GOVERNO ELETRÔNICO, 2012) GOVERNO ELETRÔNICO. **Acessibilidade**. Disponível em: <https://www.governodigital.gov.br/eixos-de-atuacao/governo/acessibilidade>. Acesso em 10 mai 2018.

(GOVERNO ELETRÔNICO, 2014) GOVERNO ELETRÔNICO. Modelo de Acessibilidade em Governo Eletrônico. Brasil, v. 3.1, 2004. Disponível em: <http://emag.governoeletronico.gov.br/>. Acesso em: 08 mar. 2018.

(HE; MUSSBACHER, 2016) HE, C.; MUSSBACHER, G**. Model-driven engineering and elicitation techniques: A systematic literature review**, In 24th Int. Requirements Eng. Conf. Workshops, 2016.

(HEITKÖTTER; MAJCHRZAK; KUCHEN, 2013) HEITKÖTTER, H.; MAJCHRZAK, T.A.; KUCHEN, H. **Cross-Platform Model-Driven Development of Mobile Applications with MD**, SAC' 13, 2013, p. 18-23.

(JUNIOR; ALMEIDA, 2009) JUNIOR, S. M. J.; ALMEIDA, G. W. **Avaliação de Acessibilidade Web: Um estudo de caso em Sítios do Governo**. Monografia (Bacharelado em Ciência da Computação) – Universidade de Brasília – UnB, Brasília, 2009.

(KENT, 2002) KENT, S. **Model-driven engineering**. Proc. 34rd Int. Conf. on Integrated Formal Methods, 2002, p. 286-298;

(LIMA; LEITE; ALMEIDA, 2017) LIMA, B. A. V.; LEITE, P. S.; ALMEIDA, L. D. A. **Writing Towards Promoting the Empowerment of Persons With Disabilities in Digital Inclusion Texts.** In Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems (Joinville, Brazil, October 23 – 27, 2017).

(LUCRÉDIO, 2009) LUCRÉDIO, D. **Uma abordagem Orientada a Modelos para Reutilização de Softaware.** Tese. Universidade de São Paulo, 2009.

(MEDEIROS, 2013) MEDEIROS, H. **Projetando e criando aplicativos para dispositivos móveis**, 2013.

(MELO, 2008) MELO, A. M. **Acessibilidade e Design Universal.** In: PUPO, D. T.; MELO, A. M. **Acessibilidade: discursos e práticas no cotidiano das bibliotecas.** São Paulo: UNICAMP, 2008.

(MIZUNO; MATSUMOTO; MORI, 2010) MIZUNO T.; MATSUMOTO, K.; MORI, N. **A ModelDriven Development Method for Management Information Systems, Electronics and Communications.** In Japan, 2010, p. 96, 2, 245-252.

(MONTOYA, 2000) MONTOYA, R. S. **Integración holística de la tecnología adaptativa**. Cádiz: Universidad de Cádiz, 2000.

(OLIVEIRA; FILGUEIRAS, 2018) OLIVEIRA, ARTHUR F. B. A.; FILGUEIRAS, LUCIA V. L. **Developer Assistance Tools for Creating Native Mobile Applications Accessible to Visually Impaired People: A Systematic Review**. In XVII Simpósio Brasileiro Sobre Fatores Humanos em Sistemas Computacionais, (Belém, Brasil, 2018).

(OMG, 2000) OMG. **OMG Unified Modeling Language Specification**. 2000.

(PANACH et al., 2014) PANACH, J. I.; JURISTO, N.; VALVERDE, F.; PASTOR, Ó. **A framework to identify primitives that represent usability within model-driven development methods.** Information and Software Tecnology. v. 58, p. 338-354, July 2014.

(PANACH; AQUINO; PASTOR, 2015) PANACH, J. I.; AQUINO, N.; PASTOR, Ó. **A proposal for modelling usability in a holistic MDD method.** Science of Computer Programming, v. 86, p. 74-88, 2015.

(PORTUGAL, 2009) PORTUGAL. Ministério da Ciência e da Tecnologia. Resolução do Conselho de Ministros Nº 97/99. Lisboa, Portugal: MCT, 1999. Disponível em <www.acessibilidade.gov.pt/acesso/res97_99.doc/>. Acesso em. 06 mar. 2018.

(POWER et al., 2012) Power, C.; Freire, A.; Petrie, H.; Swallow, D. **Guidelines are only half of the story: accessibility problems encountered by blind users on the web.** In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems. ACM, 2012, p. 433-442.

(SARRAF, 2008) SARRAF, V. P. **Reabilitação do Museu: políticas de inclusão cultural por meio da acessibilidade.** Dissertação (Mestrado). Escola de Comunicações e Artes, Universidade de São Paulo (USP), São Paulo, 2008, 180p.

(SCHMIDT, 2006) SCHMIDT, D. C. **Model-Driven Engineering**. IEEE Computer. V. 39, n. 2, 2006, p 25-31.

(SELIC, 2008) SELIC, B. **The pragmatics of model-driven development**. IBM Rational Software. v. 20, n. 5, 2008, p. 19-25.

(SERRA et al. 2015) SERRA, L. C.; CARVALHO, L. P.; FERREIRA, L. P.; VAZ, J. B. S.; FREIRE, A. P. **Accessibility Evaluation of E-Govemment Mobile Applications in Brazil.** 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion – DSAI 2015).

(SILVA; ABREU, 2014) SILVA. LUÍS P.; ABREU, FERNANDO, B. **A MDE Generative Approach form Mobile Business Apps.** In 9ht Internation Conference on The Quality of Information and Communications Technology, Guimaraes, 2014.

(SOMMERVILLE, 2011) SOMMERVILLE, I. **Engenharia de Software.** Pearson: São Paulo, ed. 9, 2011.

(STAHL; VOELTER; CZARNECKI, 2006) STAHL, T.; VOELTER, M.; CZARNECKI, K. **Model-Driven Software Development: Technology, Engineering, Management**. Wiley, West Sussex. 2006.

(TRAVASSOS; GUROV; AMARAL, 2002) TRAVASSOS, G. H.; GUROV, D.; AMARAL, E.. G. A. **Introdução à Engenharia de Software Experimental**. Programa de Engenharia de Sistemas e Computação COPPE/UFRJ, 2002

(TUFAIL et al., 2018) TUFAIL. H.; AZAM, F.; ANWAR, M. W.; QASIM, I. **Model-Driven Development of Mobile Applications: A Systematic Literature Review.** In 9th Annual Information Technology, Eletronics and Mobile Communication Conference (IEMCON), Vancouverr, BC, 2018.

(USMAN; IQBAL; KHAN, 2017) USMAN, M.; IQBAL, M. Z.; KHAN, M. U. **A product-line model-driven engineering approach for generating feature-based mobile applications**. The Journal of System and Software, v.123, p.1-32, janeiro 2017.

(VAUPEL et al., 2018) VAUPEL, S.; TAENTZER, G.; GERLACH, R.; GUCKERT, M. **Model-driven development of mobile applications for Android and iOS supporting role-based app variability.** Software & Systems Modeling, v. 17, p. 35-63, 2018.

(VIDYAPEETHAM, 2009) VIDYAPEETHAM, A. V. **An eclipse-based tool for modeling service-based systems**. Trabalho de Conclusão de Curso (Graduação em Tecnologia da Informação) – Amrita School of Engineering, Coimbatore, Tamil Nadu, India, 49p., 2009.

(W3C, 2015) W3C. **Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile.** 2015. Disponível em: < http://www.w3.org/TR/mobile-accessibility-mapping/>. Acesso em: 09 mar. 2018.

(W3C, 2017) W3C. **Mobile Accessibility Task Force (Mobile A11Y TF) of the AG WG.** 2017. Disponível em: <https://www.w3.org/WAI/GL/mobile-a11y-tf/>. Acesso em: 10 mai. 2018.

(WASSERMAN, 2010) Wasserman, A. I. 2010. **Software engineering issues for mobile application development**. In Proceedings of the FSE/SDP workshop on Future of software engineering research.ACM,397–400.

(WCAG 2.0, 2008) WCAG 2.0. **Web Content Accessibility Guidelines (WCAG) 2.0.** 2008. Disponível em: < http://www.w3.org/TR/2008/REC-WCAG20-20081211/>. Acesso em: 09 mar. 2018.

APÊNDICE A – PROTOCOLO DE PESQUISA – CARTA DE CONFIDENCIALIDADE

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Componentes de Interfaces Gráficas Acessíveis Para Aplicativos Móveis: Uma abordagem semiautomática

Você está sendo convidado(a) para participar, como voluntário, do experimento que é parte do estudo para geração de componentes de interfaces gráficas acessíveis para dispositivos, de responsabilidade do pesquisador Luan de Souza Melo (luan.sm@ppgia.pucpr.br), sob orientação da Profa. Dra. Andreia Malucelli e coorientação do Prof. Dr. André Menolli.

Por meio deste Termo de Confidencialidade, os Pesquisadores se comprometem a:

- Portar-se com discrição em todos os momentos da pesquisa acadêmica, não comentando ou divulgando qualquer tipo de informação que tenha sido repassada de forma oral ou escrita;
- Não divulgar nome do participante, em qualquer meio, a menos que expressamente autorizado por este;
- Não divulgar, em qualquer meio, os dados e informações individualizadas coletados durante o processo de pesquisa com o Participante;
- Divulgar, em formato de dissertação, artigos e apresentações, apenas os dados agregados, dos quais não se possa retirar ou inferir a identificação do Participante.

Eu, _____, após ter lido e entendido as informações e esclarecido todas as minhas dúvidas referentes a este estudo com pesquisador Luan de Souza Melo, **CONCORDO VOLUNTARIAMENTE**, em participar do mesmo.

_____ Data: ____/____/_____
Assinatura (do pesquisado ou responsável) ou impressão datiloscópica

Eu, Luan de Souza Melo, declaro que forneci todas as informações referentes ao estudo ao participante.

_____ Data: ____/____/_____
Assinatura

Equipe:

1. Luan Melo (Pesquisador)
   Pontifícia Universidade Católica do Paraná - Programa de Pós-Graduação em Informática - Grupo de Pesquisa em Engenharia de Software

2. Andreia Malucelli (Orientadora)
   Pontifícia Universidade Católica do Paraná - Programa de Pós-Graduação em Informática - Grupo de Pesquisa em Engenharia de Software

3. André Luís Andrade Menolli (Coorientador)
   Universidade Estadual do Norte do Paraná - Campus Luíz Meneghel - Centro de Ciência Tecnológica

4. Sheila Reinehr (Coorientadora)
   Pontifícia Universidade Católica do Paraná - Programa de Pós-Graduação em Informática - Grupo de Pesquisa em Engenharia de Software

APÊNDICE B – QUESTIONÁRIO DE SELEÇÃO DE PARTICIPANTES

Seleção de participantes

*Obrigatório

1. **Nome** *

2. **Qual seu gênero?**
   Marcar apenas uma oval.
   - Feminino
   - Masculino
   - Prefiro não dizer

3. **Qual sua idade?** *

4. **Você é aluno(a) de qual instituição** *
   Marcar apenas uma oval.
   - PUCPR
   - UENP
   - Não estou estudando

5. **Nível de Escolaridade** *
   Marcar apenas uma oval.
   - Cursando Graduação
   - Graduado
   - Pós-Graduado
   - Cursando Pós-Graduação

6. **Caso esteja na graduação, em qual período letivo se encontra?**

7. **Você já cursou a disciplina de IHC?** *
   Marcar apenas uma oval.
   - Sim
   - Estou cursando
   - Não

Sobre Interface Gráfica

8. **Você já desenvolveu alguma interface gráfica?** *
   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não
   - ◯ Talvez

9. **Você sabe o que são componentes de interface gráfica?** *
   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Não
   - ◯ Talvez

10. **Como classificaria o seu nível de habilidade/conhecimento no desenvolvimento de interface gráfica?** *
    1 Não tenho conhecimento/Habilidade 5 Sou especialista
    Marcar apenas uma oval.

1	2	3	4	5
◯	◯	◯	◯	◯

11. **Como classificaria o seu conhecimento de interfaces gráficas acessíveis para aplicativos móveis?** *
    1 Não tenho conhecimento 5 Sou especialista
    Marcar apenas uma oval.

1	2	3	4	5
◯	◯	◯	◯	◯

12. **Como classificaria o seu conhecimento no desenvolvimento de aplicações móveis?** *
    1 Não tenho conhecimento 5 Sou especialista
    Marcar apenas uma oval.

1	2	3	4	5
◯	◯	◯	◯	◯

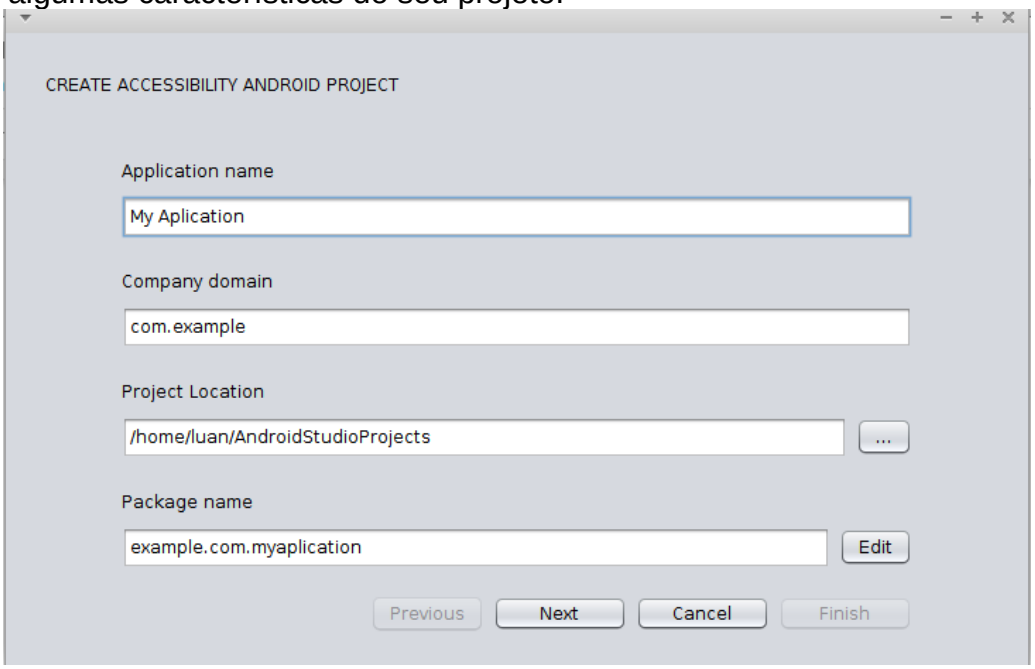
APÊNDICE C – MANUAL DA FERRAMENTA

Ferramenta de apoio a geração de componentes de interface gráfica para dispositivos móveis.

Testado e desenvolvido em:
Sistema operacional: Linux (não testado em outros S.Os)
Memória RAM: 4GB
Disco Rígido: 500GB
Processador: I3

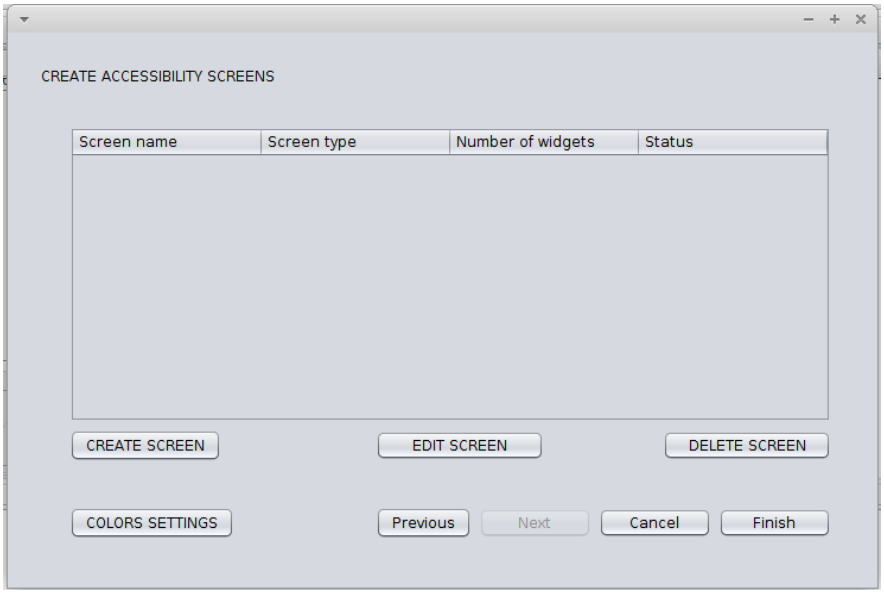
Manual de uso

1) Na primeira tela “Create Accessibility Android Project” você pode denominar algumas características do seu projeto.



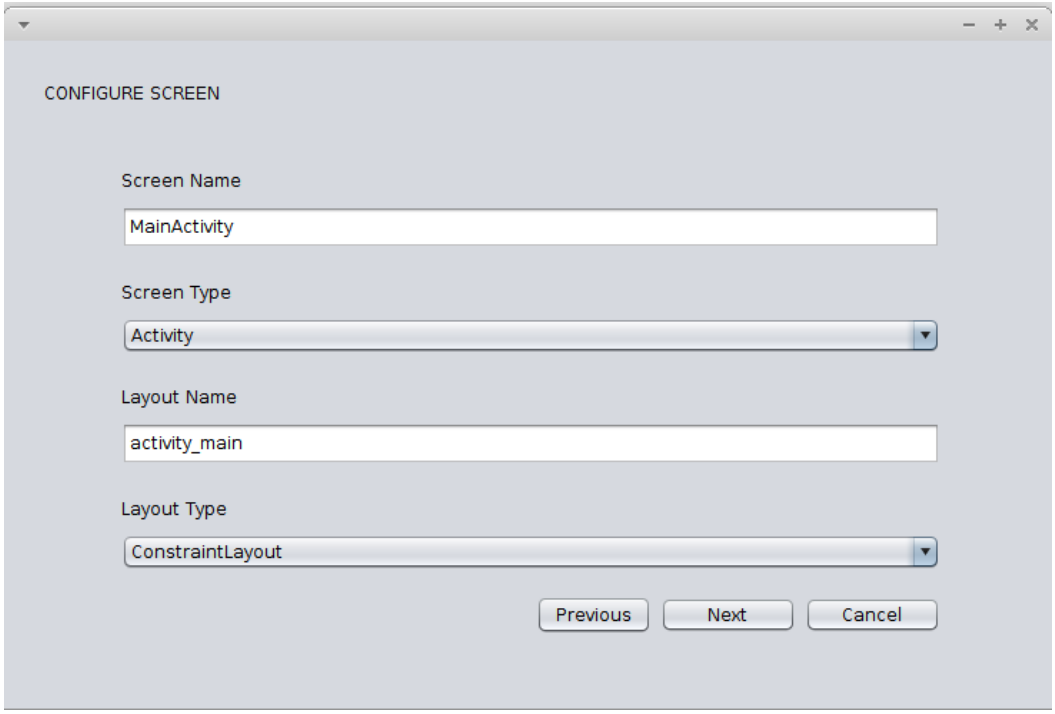
   A. Application name: campo para informar o nome da aplicação
   B. Company domain: nome do domínio. O android usa este campo para descrever o domínio da aplicação quando for indexar no google play.
   C. Project Location: Onde o projeto será armazenado.
   D. Package name: nome do pacote principal do projeto. Onde as classes serão armazenadas
   E. Next: Próxima tela
   F. Cancel: Finaliza a aplicação.

2) Posteriormente é a tela “Create Accessibility Screens”. Esta tela é responsável pela criação das Activities (telas do android).



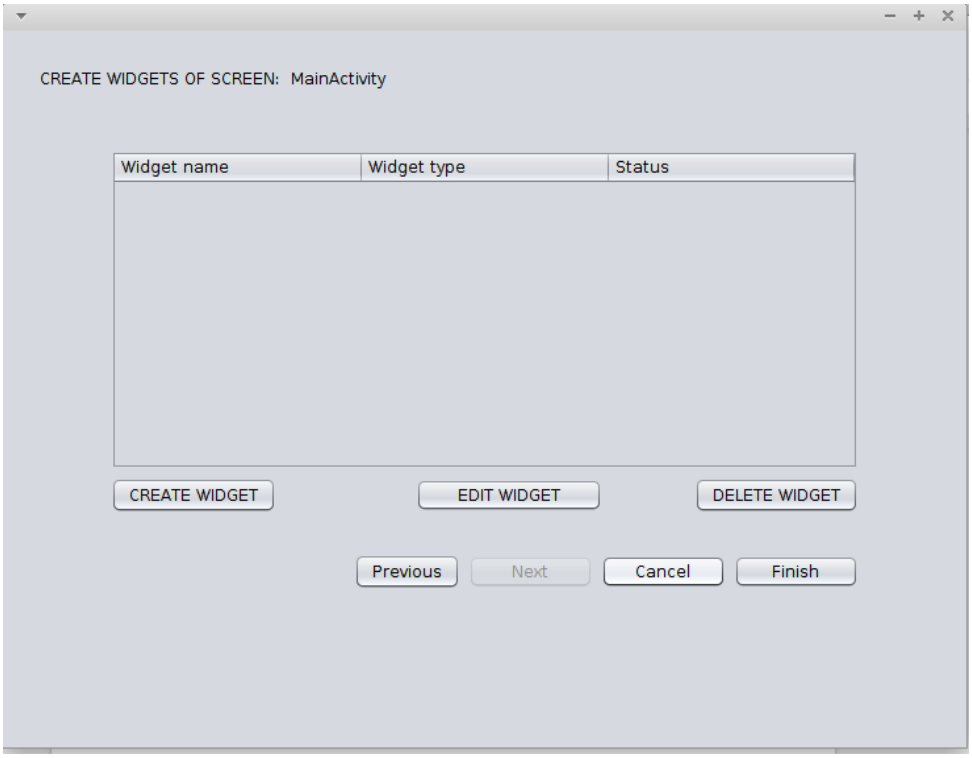
A. Dentro da tabela: após a geração de screens algumas informações serão inseridas na tabela
    a. Screen name: nome da screen
    b. Screen type: tipo de screen
    c. number of widgets: número de componentes que a screen possui
    d. status: verifica se screen está acessível
B. Create Screen: Cria uma nova Screen
C. Edit Screen; após selecionar uma screen, você pode editar ela. Basta clicar na linha que deseja alterar e clicar neste botão
D. Delete Screen: após selecionar uma screen, você pode deletar ela. Basta clicar na linha que deseja deletar e clicar neste botão
E. Definir Cores: Você pode definir algumas cores padrão para sua interface, para que não precise ficar alterando em todos os lugares. Para isto basta clicar neste botão
F. Previous: Volta para tela “Create Accessibility Android Project”.
G. Cancel: Finaliza a ferramenta sem criar o projeto
H. Finish: gera o projeto acessível

3) Posteriormente ao clicar em “Create Screen” você pode criar uma nova tela. Assim você será direcionado para tela de configuração da screen “Configure Screen”



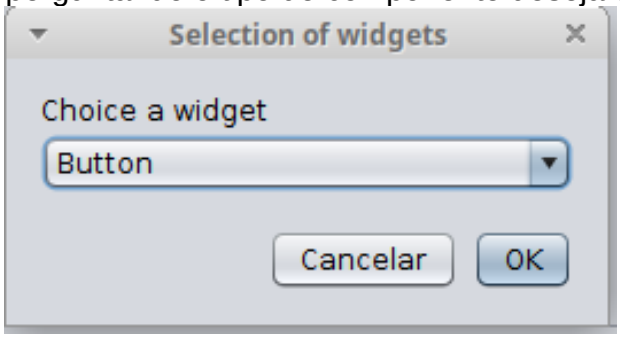
A. Screen Name: Esse é nome da Screen. Com este campo será criado a classe “activity”.
B. Screen Type: nesta versão você pode escolher somente um tipo de screen que é “Activity”
C. Layout Name: o conteúdo deste campo será gerado automaticamente de acordo com o nome da Screen Name. No entanto, é possível modificar
D. Layout Type: nesta versão você pode escolher somente um tipo de layout de screen que é “Constraint Layout”
E. Previous: Volta para tela “Create Accessibility Screen”
F. Next: Vai para Próxima tela “Create Widgets of Screen”
G. Cancel: Volta para tela “Create Accessibility Screen”

4) Nesta tela “Create Widgets of Screen” você poderá adicionar componentes a sua screen



A. Dentro da tabela: após a geração dos componentes algumas informações serão inseridas na tabela
    a. Widget name: nome do componente
    b. Widget type: tipo de widget (button, inputtext…)
    c. status: verifica se o componente está acessível
B. Create Widget: Cria um novo componente
C. Edit Widget; após selecionar um componente, você pode editar ele. Basta clicar na linha que deseja alterar e clicar neste botão
D. Delete Widget: após selecionar um widget, você pode deletar ele. Basta clicar na linha que deseja deletar e clicar neste botão
E. Previous: Volta para tela “Configure Screen”.
F. Cancel: Volta para tela “Configure Screen”.
G. Finish: Caso tenha algum componente criado, estes serão adicionados a screen e voltará para tela “Create Screen”.

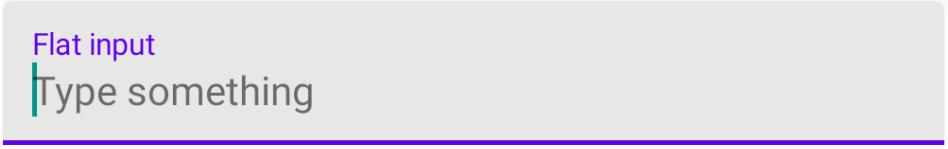
5) Após clicar em “Create Widget” na tela anterior aparecerá um componente flutuante na tela, perguntando o tipo de componente deseja criar.



Você poderá escolher os seguintes componentes:

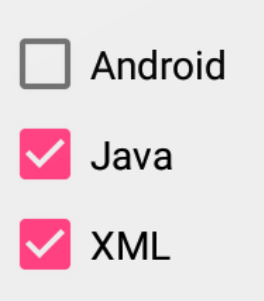
- Button (Botão de ação)

- Input Text (entrada de texto)

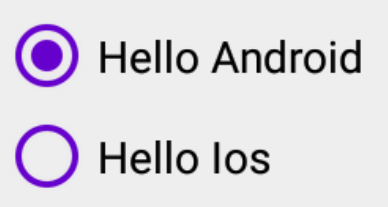


- Output Text (Saída de texto)

- CheckBox (campo de checagem)



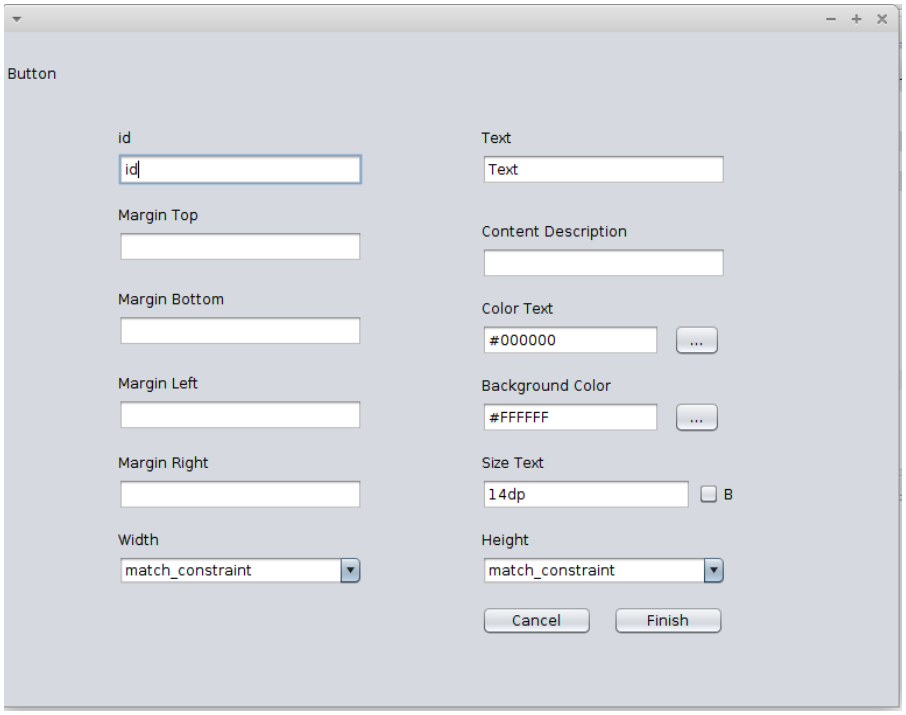
- RadioButton



- Switch

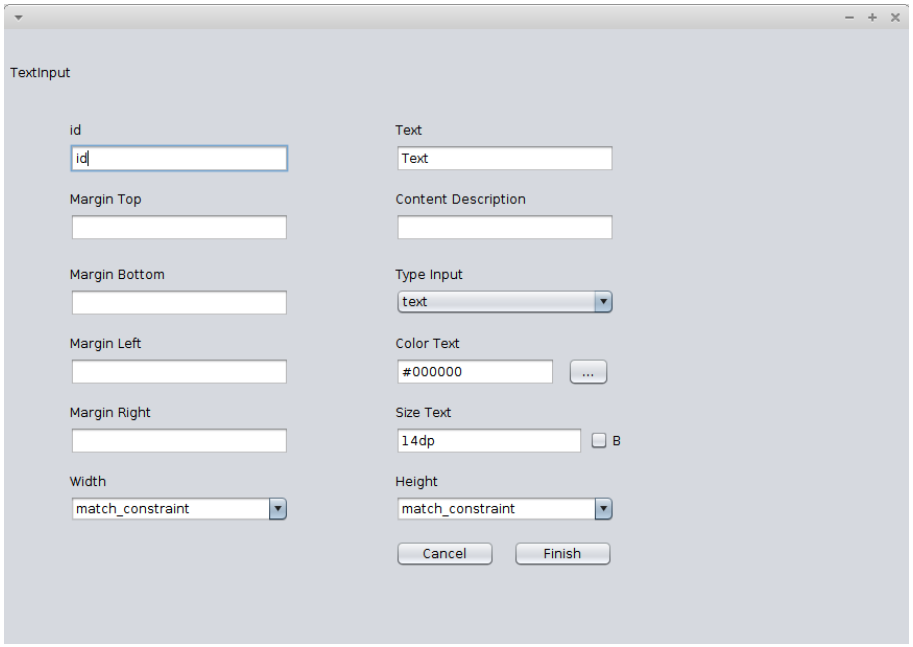
- SeekBar

6) Independente do componente escolhido os campos de todas as telas são os mesmos, tendo variação somente nas telas Button, InputText e, a maior variação, RadioButton.



A. Id: Identificador do componente (Campo obrigatório);
B. Margin Top: Margin do topo do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
C. Margin Bottom: Margin de rodapé do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
D. Margin Left: Margin da esquerda do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
E. Margin Right: Margin da direita do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
F. Width: Você definir o tamanho em comprimento do componente. Ele pode ter uma das alternativas:
    a. match_constraint: o sistema operacional arruma o tamanho em comprimento de acordo com a tela
    b. wrap_containt: usa todo o comprimento da tela para definir o tamanho do componente;
    c. [número]dp: pode definir um tamanho fixo
G. Height: Você definir o tamanho em largura do componente. Ele pode ter uma das alternativas:
    a. match_constraint: o sistema operacional arruma o tamanho em largura de acordo com a tela
    b. wrap_containt: usa toda a largura da tela para definir o tamanho do componente;
    c. [número]dp: pode definir um tamanho fixo
H. Text: Este campo define o que será escrito visualmente no componente (Campo obrigatório)
I. Content Description: Usado pela leitura de tela. (Campo obrigatório)
J. Color Text: Cor do texto. (É obrigatório, porém já vem definido)
K. Background Color: Cor de fundo do componente de botão (É obrigatório, porém já vem definido; este campo só tem no componente button)
L. Size Text: Tamanho do texto (É obrigatório, porém já vem definido; e deve ser escrito da seguinte forma [número]dp)

M. CheckBox (B): define se o texto é negrito ou não;
TextInput

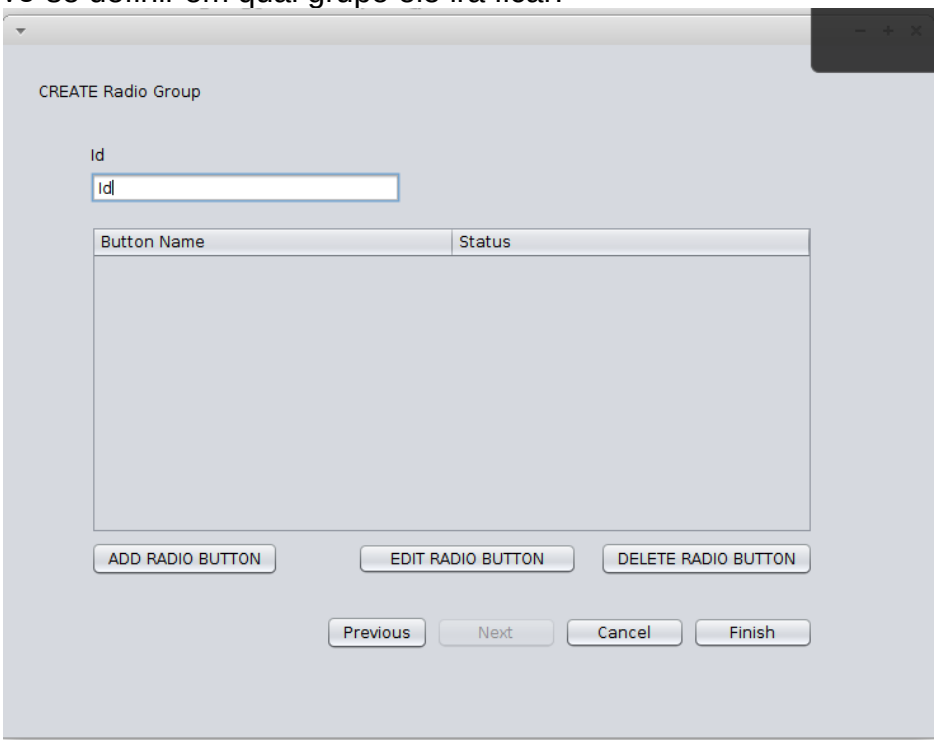


A. Id: Identificador do componente (Campo obrigatório);
B. Margin Top: Margin do topo do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
C. Margin Bottom: Margin de rodapé do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
D. Margin Left: Margin da esquerda do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
E. Margin Right: Margin da direita do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
F. Width: Você definir o tamanho em comprimento do componente. Ele pode ter uma das alternativas:
    a. match_constraint: o sistema operacional arruma o tamanho em comprimento de acordo com a tela
    b. wrap_containt: usa todo o comprimento da tela para definir o tamanho do componente;
    c. [número]dp: pode definir um tamanho fixo
G. Height: Você definir o tamanho em largura do componente. Ele pode ter uma das alternativas:
    a. match_constraint: o sistema operacional arruma o tamanho em largura de acordo com a tela
    b. wrap_containt: usa toda a largura da tela para definir o tamanho do componente;
    c. [número]dp: pode definir um tamanho fixo
H. Text: Este campo define o que será escrito visualmente no componente (Campo obrigatório)
I. Content Description: Usado pela leitura de tela. (Campo obrigatório)
J. Type Input: Tipo de entrada (ESTE CAMPO SÓ TEM NO COMPONENTE TYPE INPUT)
K. Color Text: Cor do texto. (É obrigatório, porém já vem definido)

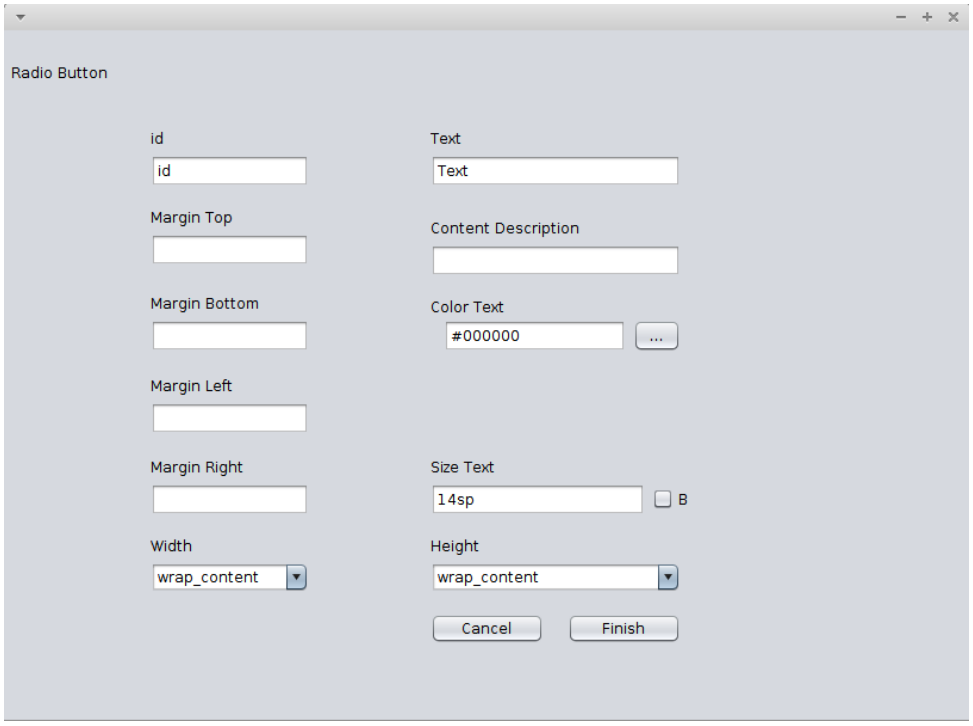
L. Size Text: Tamanho do texto (É obrigatório, porém já vem definido; e deve ser escrito da seguinte forma [número]dp)
M. CheckBox (B): define se o texto é negrito ou não;

RadioButton

No Radio Button é necessário ter um “Radio Group”. Assim antes de criar os “Radio Button” deve-se definir em qual grupo ele irá ficar.



A. Id: Identificador do componente (Campo obrigatório);
B. Dentro da tabela: após a geração dos componentes algumas informações serão inseridas na tabela
    a. Button name: nome do componente Radio Button
    b. status: verifica se o componente está acessível
C. Add Radio Button: Cria um novo componente
D. Edit Radio Button; após selecionar um componente, você pode editar ele. Basta clicar na linha que deseja alterar e clicar neste botão
E. Delete Radio Button: após selecionar um componente, você pode deletar ele. Basta clicar na linha que deseja deletar e clicar neste botão
F. Previous: Volta para tela “Create Widget”.
G. Cancel: Volta para tela “Create Widget”.
H. Finish: Caso tenha algum componente criado, estes serão adicionados a screen e voltará para tela “Create Widget”



A. Id: Identificador do componente (Campo obrigatório);
B. Margin Top: Margin do topo do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
C. Margin Bottom: Margin de rodapé do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
D. Margin Left: Margin da esquerda do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
E. Margin Right: Margin da direita do componente (Campo não obrigatório e deve ser escrito da seguinte forma [número]dp)
F. Width: Você definir o tamanho em comprimento do componente. Ele pode ter uma das alternativas:
    a. match_constraint: o sistema operacional arruma o tamanho em comprimento de acordo com a tela
    b. wrap_containt: usa todo o comprimento da tela para definir o tamanho do componente;
    c. [número]dp: pode definir um tamanho fixo
G. Height: Você definir o tamanho em largura do componente. Ele pode ter uma das alternativas:
    a. match_constraint: o sistema operacional arruma o tamanho em largura de acordo com a tela
    b. wrap_containt: usa toda a largura da tela para definir o tamanho do componente;
    c. [número]dp: pode definir um tamanho fixo
H. Text: Este campo define o que será escrito visualmente no componente (Campo obrigatório)
I. Content Description: Usado pela leitura de tela. (Campo obrigatório)
J. Type Input: Tipo de entrada (ESTE CAMPO SÓ TEM NO COMPONENTE TYPE INPUT)
K. Color Text: Cor do texto. (É obrigatório, porém já vem definido)

L. Size Text: Tamanho do texto (É obrigatório, porém já vem definido; e deve ser escrito da seguinte forma [número]dp)
M. CheckBox (B): define se o texto é negrito ou não;

APÊNDICE D – REGRAS DE TRANSFORMAÇÃO

PRIMEIRA REGRA – UM COMPONENTE ABAIXO DO OUTRO

Este método é responsável por gerar o código dos componentes. Dentro do método há um laço de repetição que gera o código do componente. Assim ele adiciona o código um abaixo do outro.

```
if (window.getLayoutType().equals("ConstraintLayout") && window.getScreenType().equals("Activity")) {
    winXML = "<android.support.constraint.ConstraintLayout xmlns:android=\"http://schemas.android.com/apk/res/android\"\n"
            + "    xmlns:app=\"http://schemas.android.com/apk/res-auto\"\n"
            + "    xmlns:tools=\"http://schemas.android.com/tools\"\n"
            + "    android:layout_width=\"match_parent\"\n"
            + "    android:layout_height=\"match_parent\"\n"
            + "    tools:context=\"." + window.getScreenName() + "\">";
    for (Widget widget : window.getWidgets()) {
        switch (widget.getTipoWidget()) {
            case "Button":
                winXML += "\n" + createButton(widget, local);
                local += 30;
                break;
            case "TextInput":
                winXML += "\n" + createTextInput(widget, local);
                local += 30;
                break;
            case "CheckBox":
                winXML += "\n" + createCheckBox(widget, local);
                local += 30;
                break;
            case "RadioGroup":
                winXML += "\n" + createRadioGroup(widget, local);
                local += 30 * ((RadioGroup) widget).getRadioButtons().size();
                break;
            case "TextOutput":
                winXML += "\n" + createTextOutput(widget, local);
                local += 30;
                break;

            case "Switch":
                winXML += "\n" + createSwitch(widget, local);
                local += 30;
                break;
            case "SeekBar":
                winXML += "\n" + createSeekBar(widget, local);
                local += 30;
                break;
            case "ImageButton":
                winXML += "\n" + createImageButton(widget, local);
                local += 30;
                break;
        }
    }
    winXML += "\n" + "</android.support.constraint.ConstraintLayout>";
```

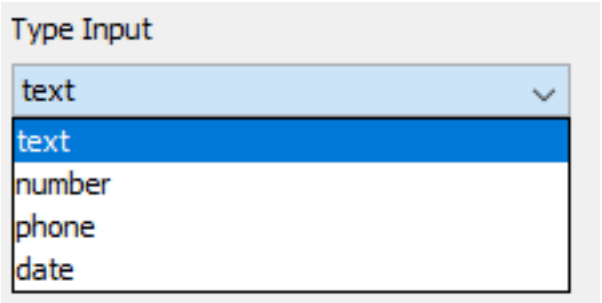
SEGUNDA E TERCEIRA REGRA – CONTRASTE

Neste método é realizado o cálculo de luminância das cores. Para isto método recebe de entrada duas cores e retorna o valor relativo entre as duas.

Após e feito a verificação se a cor que o usuário colocou é válida, ou seja, se atende as diretrizes de acessibilidade. O valor para ser aceito deve ser no mínimo maior que 4.5.

```
public static Double validarCor(Color c1, Color c2)
{
    //Obtem o valor da primeira cor
    Double rs1 = new Double(c1.getRed() / 255);
    Double gs1 = new Double(c1.getGreen() / 255);
    Double bs1 = new Double(c1.getBlue() / 255);

    //Obtem o valor da segunda cor
    Double rs2 = new Double(c2.getRed() / 255);
    Double gs2 = new Double(c2.getGreen() / 255);
    Double bs2 = new Double(c2.getBlue() / 255);

    //Obtem o valor da segunda corw
    Double r1 = rs1 <= 0.03928 ? rs1 / 12.98 : Math.pow(((rs1 + 0.055) / 1.055), 2.4);
    Double g1 = gs1 <= 0.03928 ? gs1 / 12.98 : Math.pow(((gs1 + 0.055) / 1.055), 2.4);
    Double b1 = bs1 <= 0.03928 ? bs1 / 12.98 : Math.pow(((bs1 + 0.055) / 1.055), 2.4);

    Double r2 = rs2 <= 0.03928 ? rs2 / 12.98 : Math.pow(((rs2 + 0.055) / 1.055), 2.4);
    Double g2 = gs2 <= 0.03928 ? gs2 / 12.98 : Math.pow(((gs2 + 0.055) / 1.055), 2.4);
    Double b2 = bs2 <= 0.03928 ? bs2 / 12.98 : Math.pow(((bs2 + 0.055) / 1.055), 2.4);

    Double l1 = (r1 * 0.2126) + (g1 * 0.7152) + (b1 * 0.0722);
    Double l2 = (r2 * 0.2126) + (g2 * 0.7152) + (b2 * 0.0722);

    l1 += 0.05;
    l2 += 0.05;

    return l1 <= l2 ? l2 / l1 : l1 / l2;
}
```

QUINTA REGRA – TIPO DE TECLADO VIRTUAL



SEXTA REGRA – LEITOR DE TELA

Caso o usuário não insira nada no campo essa verificação exibi uma mensagem de erro.

```
if (tfContentDescription.getText().equals("")) {
    erro.setText("Corrigir campo(os)");
    erro.setVisible(true);
    lbContentDescription.setText("Content Description  (Inserir uma Context Description)");
    lbContentDescription.setForeground(Color.RED);
    finish = false;
}
```