UNIVERSIDADE DE UBERABA
KAUANE SILVA PEREIRA

DESENVOLVIMENTO FACILITADO DE TELAS ACESSÍVEIS: IMPLEMENTAÇÃO DE UM MÓDULO PARA APRIMORAR A ACESSIBILIDADE EM COMPONENTES DE INTERFACE ANDROID

UBERLÂNDIA – MG
2023

KAUANE SILVA PEREIRA

DESENVOLVIMENTO FACILITADO DE TELAS ACESSÍVEIS: IMPLEMENTAÇÃO DE UM MÓDULO PARA APRIMORAR A ACESSIBILIDADE EM COMPONENTES DE INTERFACE ANDROID

Trabalho apresentado à Universidade de Uberaba, como parte das exigências à conclusão do componente Trabalho de Conclusão de Curso, da 10ª etapa, do curso de graduação em Engenharia da Computação, da UNIUBE, Campus Uberlândia.

Orientador: Prof. Me. Clênio Eduardo da Silva

UBERLÂNDIA – MG
2023

DESENVOLVIMENTO FACILITADO DE TELAS ACESSÍVEIS: IMPLEMENTAÇÃO DE UM MÓDULO PARA APRIMORAR A ACESSIBILIDADE EM COMPONENTES DE INTERFACE ANDROID

Kauane Silva Pereira
kauane.nep@gmail.com
Clênio Eduardo da Silva
clenio.silva@uniube.br

RESUMO

Atualmente a maior parte das interações tecnológicas feitas por meio de dispositivos móveis são realizadas através de aplicativos, todavia a garantia que estes recursos sejam utilizáveis pelo público de usuários que apresentam algum tipo de deficiência ou necessidades de utilização das tecnologias assistivas é responsabilidade da equipe de desenvolvimento da aplicação. Esse trabalho tem como objetivo criar um módulo de componentes acessíveis para aplicações nativo *android*, com intuito de manipular a semântica de elementos de interface utilizando linguagens nativas, para facilitar a implementação de acessibilidade durante o desenvolvimento de aplicativos.

Palavras-chave: Dispositivos móveis, aplicativos, tecnologias assistivas.

ABSTRACT

Currently, most technological interactions via mobile devices occur through applications. However, ensuring that these resources are usable by users who have some form of disability or require the use of assistive technologies is the responsibility of the application development team. This work aims to create a module of accessible components for native Android applications, intending to manipulate the semantics of interface elements using native languages to facilitate the implementation of accessibility during app.

Keywords: Mobile devices, applications, assistive Technologies.

1 INTRODUÇÃO

De acordo com a Universidade Federal do Ceará (UFC), o termo acessibilidade refere-se a prática de tornar produtos, serviços e informações, disponíveis e utilizáveis a todas as pessoas, independente de suas limitações, dificuldades ou características. O objetivo principal da acessibilidade é eliminar barreiras que possam impedir ou dificultar a participação igualitária de todas as pessoas na sociedade (EDYBURN, 2004).

Assim, é um direito social o acesso facilitado de todos os públicos a atividades contemporâneas. Atualmente a disseminação de novas tecnologias móveis vem ganhando espaço no cotidiano da população. De acordo com o sistema de telecomunicações do governo brasileiro em 2020, cerca de 234 milhões de pessoas realizaram acessos móveis através de *smartphones*, *tablets* e outros dispositivos, sendo um aumento de 3,26% em relação a 2019.

De acordo com a Organização Mundial da Saúde (OMS) em 2019, cerca de 2,2 bilhões de pessoas no mundo apresentam algum grau de deficiência visual, contudo grande parte destes indivíduos tem acessos às novas tecnologias. O questionamento a ser realizado é como essa parte da sociedade interage com os recursos digitais disponíveis atualmente (LOPES, 2023).

Os acessos móveis são intermediados através de interfaces, conhecidas comumente como aplicativos, estes que criam telas interativas para que usuários possam realizar atividades como buscas em redes de *internet*, ligações telefônicas e acessar demais recursos de seus aparelhos. Todavia, nem todo aplicativo utiliza de recursos nativos, conhecidos como tecnologias assistivas, para facilitar o acesso de usuários com alguma limitação.

Tecnologias assistivas se apresentam como recursos do sistema operacional, assim como contextualizado por (Edyburn, 2004, p.16):

> A tecnologia assistiva significa qualquer item, peça de equipamento ou sistema de produto, adquirido comercialmente pronto para uso, modificado ou customizado, que é usado para aumentar, manter ou melhorar o desempenho funcional de capacidades.

Desta maneira as características presentes em ferramentas acessíveis proporcionam leitura dinâmica de telas, legendas em tempo real, contraste de cores

e tamanhos de fontes, entre demais recursos que podem ser explorados durante o desenvolvimento de aplicativos. Cada sistema operacional presente em tecnologias móveis apresenta suas individualidades para com as tecnologias assistivas, através de bibliotecas nativas, escritas em diferentes linguagens de programação é possível manipular os elementos de visualização do usuário final.

De acordo com a Globalstates[1] entre os sistemas operacionais amplamente adotados, o *Android* ocupa 70,9% de utilização entre os usuários, sendo o sistema mais utilizado mundialmente. Distinto por sua ampla utilização e escalabilidade é evidenciado pela gama de fabricantes de dispositivos que adotaram o *Android* como seu sistema operacional padrão, contribuindo para uma base de usuários expansiva. Além disso, sua natureza de código aberto permite personalizações profundas, adequando-se a uma ampla gama de necessidades, desde *smartphones* acessíveis até dispositivos que utilizem Internet das Coisas (IOT).

Assim, esse projeto tem como objetivo geral desenvolver uma implementação nativa com classes abertas que modifiquem a semântica e comportamento de determinados elementos de interface presentes em aplicativos *Android*, proporcionando maior facilidade de implementação de tecnologias assistivas em tempo de desenvolvimento. Garantindo que o *software* final consiga entregar uma experiência satisfatória e interativa de interfaces acessíveis a todos os públicos.

Para tanto, os objetivos específicos são: avaliar os elementos de acessibilidade presentes na documentação do sistema operacional *android* e assim criar um módulo implementável que traga características assistivas a elementos de interface.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta sessão será explorado de forma completa os conceitos supracitados, trazendo embasamento teórico e técnico dos recursos explorados para desenvolvimento do projeto. Tais como: estruturas de projetos, usabilidade de tecnologias assistivas e desenvolvimento de código com linguagens nativas *Android*.

[1] Disponível em <https://www.globalstates.com>. Acesso em: 01 nov. 2023.

2.1 Tecnologia Assistiva

Caracterizada pelo comitê de ajudas técnicas (CTA), realizado pela secretaria dos direitos humanos da Presidência da República “Tecnologia Assistiva é uma área do conhecimento de característica interdisciplinar que engloba produtos, recursos, metodologias, estratégias, práticas e serviços. Tal área objetiva promover a funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social” (NAZARI, 2017).

Dentro do contexto de tecnologias móveis, os recursos assistivos estão presentes no sistema operacional do *hardware*, cada fabricante disponibiliza configurações próprias que interagem com interfaces de aplicativos. Segundo (Hakobyan *et al*, 2013, p.514):

> Dispositivos que são discretos ou aplicativos que estão incorporados em um dispositivo convencional, como um celular, podem ajudar as pessoas a se sentirem menos estigmatizadas ou rotuladas. Além disso, os sistemas assistivos geralmente são adaptáveis em várias plataformas móveis e podem oferecer suporte a várias deficiências.

2.1.1 Acessibilidade em sistemas operacionais android

O *Android* oferece para os desenvolvedores uma documentação vasta sobre recursos nativos disponíveis através de bibliotecas *Java*, linguagem base para o desenvolvimento de implementações gerais de aplicativos móveis. Dentre estes recursos é abrangível a manipulação de interfaces visuais, sonoras e localização.

Desenvolvido inicialmente pela Android Inc, uma empresa fundada por Andy Rubin, Rich Miner, Nick Sears e Chris White. O sistema operacional foi criado com o objetivo de ser uma plataforma aberta e flexível para dispositivos móveis. Em 2005, a Google adquiriu a Android Inc, e continuou a desenvolver e aprimorar o sistema. Atualmente, o *Android* é mantido e gerenciado pela Google, que lança atualizações periódicas do sistema operacional, juntamente com suas versões personalizadas por fabricantes de dispositivos específicos (OLIVEIRA *et al*, 2014).

2.1.2 Leitor de tela - *Talkback*

O *Talkback* é uma funcionalidade acessível que auxilia pessoas com deficiência visual, analfabetismo ou até dificuldades momentâneas de reconhecer os elementos visuais presentes nas aplicações *mobile*[2]. Este é um recurso da Google incluído em dispositivos *android*, tendo sua configuração especificada por cada fabricante, podendo ter alterações de gestos e comportamentos, dependendo da versão do *android* instalada dentro de cada *hardware*.

De acordo com a documentação presente na suportgoogle[3], o leitor de tela após ser ativado oferece uma experiência de leitura dinâmica, sua navegação se dá através de toques na tela. Em versões mais recentes é possível configurar múltiplos toques com até 3 dedos, permitindo que o usuário deslize de cima para baixo ou da esquerda para direita, fazendo assim a movimentação do leitor de forma ordenada pelos elementos de visualização presentes na interface.

Atualmente também é disponibilizado a configuração de atalhos por meio de gestos, customizáveis de acordo com a necessidade do usuário, trazendo a possibilidade de rápido acesso de aplicações e recursos mais utilizados.

Todavia, em versões anteriores do sistema *android*, as configurações do *talkback* permitiam uma movimentação limitada, fazendo a leitura de apenas um dedo para deslizar entre os elementos, com movimentos de cima para baixo, controlando assim a leitura.

2.1.3 Conversor de texto em voz

Outra funcionalidade a ser explorada no contexto de tecnologias assistivas no sistema *android* é a conversão de voz em texto, podendo ser usada em mídias como vídeos, *podcasts*, chamadas telefônicas e vídeo chamadas. Seu funcionamento consiste em converter dinamicamente o som em legendas que podem ser acompanhadas por usuários que possuem alguma deficiência auditiva ou até indisponibilidade momentânea de escutar o som transmitido por seu aparelho móvel.

Sendo um recurso relativamente novo, de acordo com a documentação *android*, seu funcionamento é garantido de forma mais eficaz em *hardwares* com

[2] Tecnologias móveis
[3] Disponível em <https://www.suportgoogle.com>. Acesso em: 01 nov. 2023.

versões superiores ao *android* 10. Sendo possível a ativação de legendas instantâneas em dispositivos *pixel* 2 configurados com idioma inglês, e dispositivos *pixel* 6 *e pixel* 6 *pro* em inglês, francês, alemão, italiano, japonês e espanhol, podendo também realizar a tradução automática por meio das legendas de mídias destes aparelhos.

Passível de configurações é possível ajustar o local da caixa de texto com as legendas, juntamente com o tamanho da fonte configurada no dispositivo, apresentando conexão com o *talkback,* sendo possível exibir o texto verbalizado por ele.

2.2 *Kotlin*

Kotlin pode ser descrito como um projeto de código aberto sem custos, financiado pela Apache 2.0. Seu código é disponibilizado pelo *github* sendo mantido principalmente pelas equipes da JetBrains e Google. Seu desenvolvimento é multiplataforma, sendo disponibilizado para sistemas operacionais como iOS, Android, macOS, Windows, Linux, watchOS, entre outros conforme kotlinlang[4].

Baseada em princípios de orientação a objeto a linguagem de programação *kotlin* é compilada pela máquina virtual *Java*, apresentando vantagens em sua escrita como maior facilidade de manipulação de recursos, sintaxe mais simples e resumida. De acordo com developer[5] popularizou-se principalmente para desenvolvimento de aplicações nativas *android*, sendo amplamente adotado por empresas como Netflix, Duolingo, Kindle, entre outras.

Portanto, neste projeto o desenvolvimento será voltado para linguagem de programação *kotlin*, sendo uma tecnologia atual e reconhecida por grandes empresas, tendo uma sintaxe simples e direta, com documentações atuais e de fácil acesso.

2.3 Diretrizes para qualidade de código

Para o desenvolvimento de *softwares*, conceitos e diretrizes voltados para qualidade são amplamente utilizados, representando boas práticas aplicáveis

[4] Disponível em <http://www kotlinlang.org>. Acesso em: 01 nov. 2023.
[5] Disponível em <http://www.developer.android.com>. Acesso em: 01 nov. 2023.

durante o desenvolvimento de um produto digital. Cada código tem suas particularidades, cabendo ao desenvolvedor garantir a qualidade, assim como citado na obra o mítico homem-mês por (Brooks, 1975, p.48), segundo ele:

> O programador, como o poeta, trabalha sozinho. Mesmo hoje, cada código é um ato criativo de pensamento solitário. Ninguém pode antecipar com clareza qual será a sua forma definitiva.

Conforme o pensamento de Brooks, pode-se concluir que não existe uma definição única de como de desenvolver uma aplicação, todavia cabe ao programador decidir quais conceitos devem ser adotados de acordo com o escopo de seu projeto.

2.3.1 Código Limpo – *Clean Code*

O conceito de *clean code* se entende como um conjunto de boas práticas aplicáveis no dia a dia de um desenvolvedor, seu intuito é manter um código limpo, de fácil manutenção. Criado por Robert Martin, a metodologia presente neste padrão de projeto foi descrita em sua obra Código limpo: Habilidades práticas de Agile Software, lançado no ano de 2008.

Neste projeto serão aplicados princípios utilizados na obra supracitada, como, nomes significativos, mantendo a nomenclatura de classes, objetos e variáveis de fácil entendimento e compreensão aos demais desenvolvedores. Formatação coerente das estruturas de código, visando manter o alinhamento das linhas para fácil visualização e testes de unidades, criando testes unitários de funções desenvolvidas para garantir seu funcionamento em diferentes cenários.

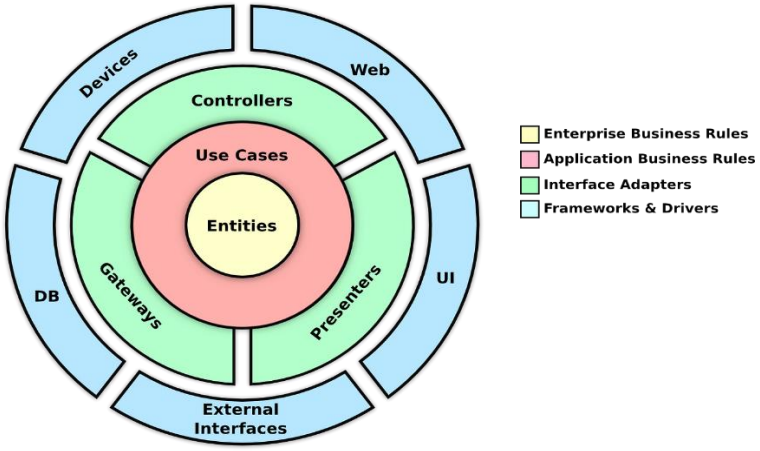
2.3.2 Arquitetura Limpa – *Clean Architecture*

Arquitetura limpa se entende como um conjunto de conceitos que tendem a desenvolver camadas de código independentes, viabilizando a recursividade de classes e objetos, permitindo criar segmentações voltadas para cada etapa,

desenvolvendo códigos testáveis e de fácil alteração do *software*, assim como mencionado na obra Arquitetura Limpa de Robert Martin.

Pode-se observar as seguintes camadas de desenvolvimento na imagem abaixo:

Figura 1 – Camadas arquitetura limpa



Fonte: Blog Medium, Disponível em: https://joaogbsczip.medium.com/clean-architecture-reactjs-pt-2-fa1a166dcfea. Acesso em: 02 nov. 2023.

- *Entities*, conhecida como camada de entidades, tem como objetivo representar os objetos de negócio presentes na estruturação do projeto, encapsulando o comportamento, relacionado a conceitos específicos do domínio.
- *Use case,* reconhecida como camada de regras de negócio, apresentando os casos de uso presentes na aplicação.
- *Interface Adapters,* a camada de interface representa a programação de telas e funções que serão manipuladas pelo usuário final, realizando o desenvolvimento de futuras ações e comportamentos de *view*[6].
- *Frameworks* e *Drivers*, são responsáveis pelas comunicações mais externas, utilizando recurso como banco de dados e chamadas de *api*, podendo depender de camadas anteriores como interfaces e regras de negócio.

[6] Termo utilizado para elementos de visualização, como layouts e telas.

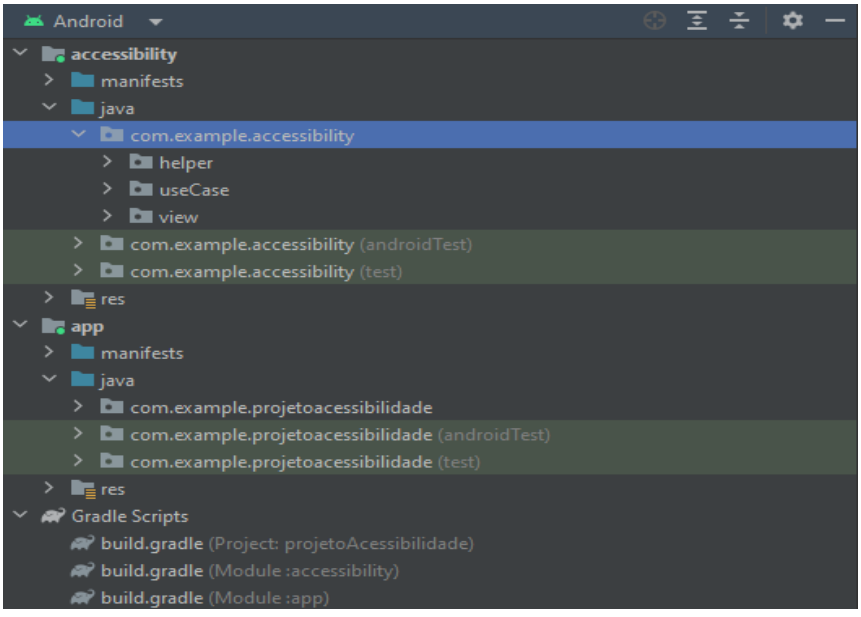
Diante disso, este projeto é baseado nos princípios de desenvolvimento supracitados, apresentando em seu desenvolvimento uma camada de *use case*, contendo regras de funcionamento para as funções implementáveis, exemplos utilizando interfaces interativas e independentes, juntamente com *frameworks* como o *Retrofit*[7], que proporcionam maior facilidade para chamadas de *api*, compondo o conteúdo dos arquivos de *view*.

2.3.3 Modularização

Componentização, termo análogo a modularização, referem-se ao ato de subdividir um projeto em módulos menores e independentes. Em um contexto de desenvolvimento *android*, ao subdividir um projeto, cada componente apresenta suas próprias configurações de *gradle*[8], implementações e serviços (ELY, 2019).

Uma grande vantagem de utilizar essa abordagem é manter um versionamento independente, abordagem utilizada neste projeto para facilitar a implementação de recursos assistivos dentro de exemplos de aplicações *mobile*.

Figura 2 – Modularização de pacotes e arquivos no Android Studio.



Fonte: Autoria própria, 2023.

[7] Framework utilizado para realizar requisições de serviços Rest.
[8] Sistema de automação e compilação de código.

Nesta imagem é possível encontrar dois módulos distintos que compartilham o mesmo projeto, sendo *accessibility* o primeiro componente, que é responsável pelas implementações de acessibilidade, o segundo armazena os arquivos de visualização do *app*, sendo executado ao iniciar o projeto.

2.4 *Github*

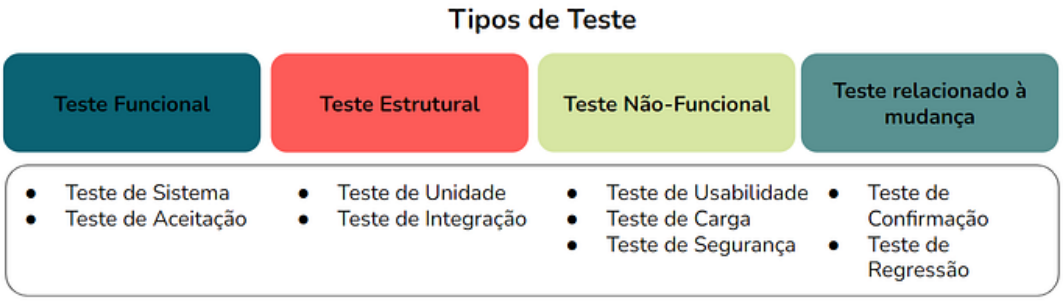
O *GitHub* é uma plataforma online de renome no universo do desenvolvimento de *software*, largamente adotada para gerenciamento de projetos. Sua funcionalidade central consiste em fornecer um ambiente unificado onde desenvolvedores podem depositar, controlar e colaborar no código-fonte de suas criações. Através do emprego do sistema de controle de versão *Git,* o *GitHub* capacita equipes a monitorarem e gerirem as alterações efetuadas no código ao longo do tempo, o que facilita uma colaboração sinérgica e a manutenção de projetos de *software,* assim como citado em github.com[9].

2.5 Cenários de teste

Existem diferentes metodologias que podem ser aplicadas para o teste de um aplicativo, cada uma contendo suas particularidades, assim são aplicadas de acordo com a necessidade do projeto, garantindo a integridade e usabilidade de um *software* antes de seu lançamento. A seguir estão alguns exemplos de testes a serem aplicados:

[9] Disponível em <https://docs.github.com/pt/get-started/quickstart/hello-world>. Acesso em: 03 nov. 2023.

Figura 3 – Tipos de teste de software



Fonte: Blog Medium, Disponível em: https://medium.com/@annerocha_qa/principais-tipos-de-teste-de-software-4aeeb7fd23f1. Acesso em: 03 nov. 2023.

Testes estruturais conhecidos também como testes de caixa branca são realizados através do código, desenvolvidos pela própria equipe de desenvolvimento, geralmente usados em aplicações que possuam o consumo de serviços externos.

Os testes funcionais são realizados pelo profissional de teste, sendo ele um teste da aplicação, que busca evidenciar sua experiência ao utilizar o programa. Um teste não funcional garante a estabilidade da aplicação, podendo observar a quantidade de dados máximos a serem trafegados, a segurança e vazamento de informações sensíveis, entre outros aspectos que englobam a segurança.

Testes relacionados a mudança podem ser utilizados de diferentes formas, geralmente são associados a correções direcionadas ao desenvolvedor, que devem ser corrigidas e testadas novamente (DELMARO, 2013).

Neste projeto foram aplicados conceitos de testes funcionais e relacionados a mudança, posteriormente serão exemplificados o progresso destas execuções.

3 METODOLOGIA

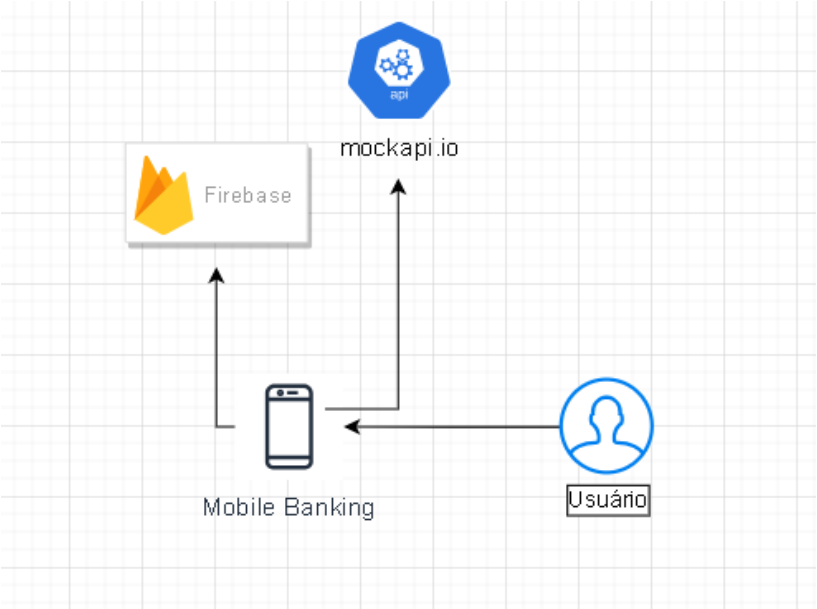
Para desenvolvimento deste projeto foi necessário a criação de uma aplicação *mobile android* base. Durante seu desenvolvimento foram incorporadas funções do módulo *accessibility*, que abriga as ferramentas desenvolvidas para simplificar a integração das tecnologias assistivas previamente estudadas.

Assim, a seguir serão apresentados os caminhos utilizados para desenvolvimento deste projeto, juntamente com as considerações acerca dos resultados desejados.

3.1 Projeto base

Para incorporar as funcionalidades de tecnologia assistiva desenvolvidas neste projeto, foi necessário criar uma aplicação *Android* que oferecesse interfaces interativas, com o objetivo de representar um aplicativo móvel convencional, amplamente utilizado pelos usuários *mobile*. Assim, foi desenvolvido um aplicativo de interface bancária, integrando em sua estrutura a interação com serviços externos, permitindo a exemplificação de cenários realistas enfrentados por usuários cotidianamente.

Figura 4 – Estrutura da aplicação



Fonte: Autoria própria, 2023.

Acima é possível visualizar a estrutura do projeto base, denominado *Mobile Banking*, sendo assim o serviço disponibilizado ao usuário, o qual representa o público que irá acessar a aplicação fornecida. Pode-se destacar também o uso de dois *frameworks*, que serão utilizados para requerer funcionalidades externas.

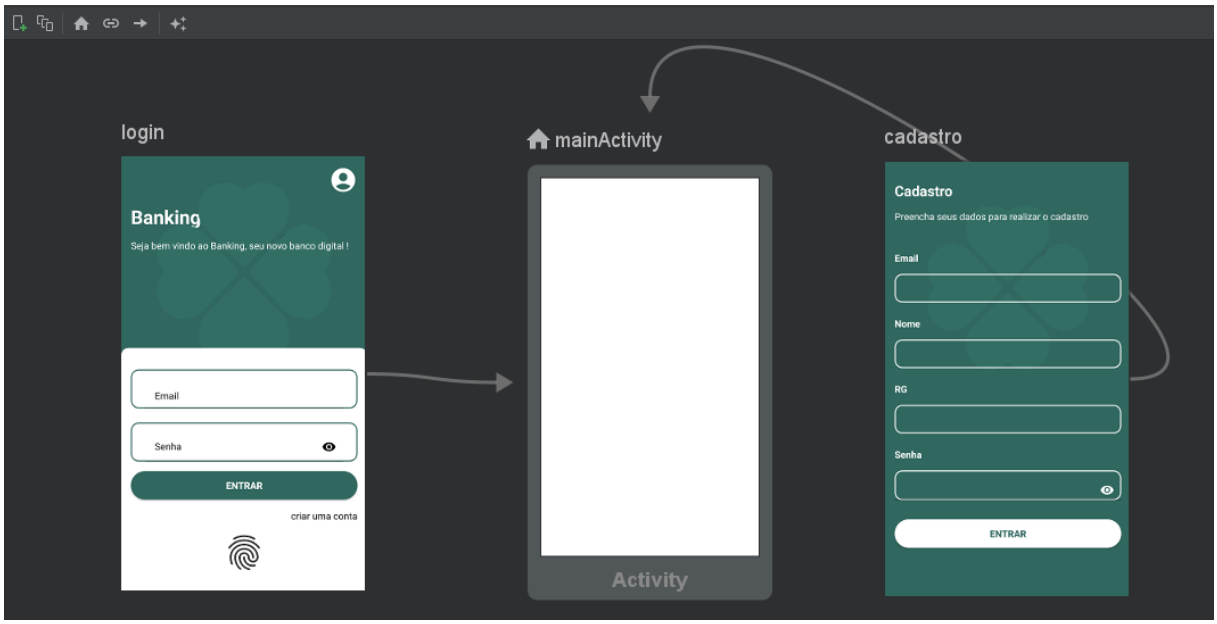
Desenvolvido como um serviço de computação em nuvem voltado para plataformas de desenvolvimento de aplicativos, o *firebase* é um *framework* disponibilizado pela equipe Google. Sendo implementado neste projeto, para consumo de recursos como autenticação de usuários através de chaves *key*, criadas na realização de novos cadastros, juntamente com armazenamento para banco de dados, responsáveis por reunir características de novos usuários.

Outro recurso explorado para criação de interfaces dinâmicas, foi o consumo da *api mock.io*, nela é possível gerar *jsons*[10] dinâmicos, trazendo dados mocados conforme a configuração do projeto, desta forma é possível receber listas de dados que serão aplicados durante o desenvolvimento como alternativa para popular objetos criados na interface visual.

3.2 Jornada da aplicação

Para a implementação do módulo de acessibilidade, foi desenvolvida uma jornada padrão que inclui elementos de visualização comumente utilizados na composição do arquivo de *layout XML*[11]. A atividade principal, inicialmente, abriga dois fragmentos, o primeiro é destinado ao *login*, permitindo aos usuários acessar a aplicação, enquanto o segundo é voltado para o cadastro, possibilitando que novos usuários se inscrevam no sistema, caso ainda não tenham uma conta.

Figura 5 – Jornada inicial



Fonte: Autoria própria, 2023.

Ao analisar as interfaces fornecidas é possível encontrar componentes visuais que não apresentam semânticas apropriadas para usuários de tecnologia assistiva, dentre eles é possível destacar a presença de títulos, estes que visualmente se

[10] Formato baseado em texto padrão para representar dados estruturados
[11] Estrutura visual para interface.

destacam por apresentar características com fontes maiores, e estilo mais robusto em comparação aos demais.

Assim, para um usuário que não necessita de nenhum recurso de acessibilidade é fácil identificar o começo da tela baseado na configuração do título, representando o início das informações fornecidas, todavia para um usuário de acessibilidade é preciso que o título seja verbalizado com a literal apropriada, para identificação de onde a aplicação é iniciada, dando melhor direcionamento entre o começo e o final da leitura de tela.

Neste mesmo fluxo é possível encontrar na tela de *login*, dois objetos clicáveis que fogem da estética padrão de um botão, que seria a alternativa técnica comumente utilizada para direcionar um usuário para um próximo passo. Todavia é normalmente encontrado em *design*[12] de interface, componentes que apresentam a função de um botão, porém precisam ter uma estética diferente para tornar a tela mais harmônica ao *layout* proposto.

Neste caso podemos encontrar esta situação em *links* ou em imagens que remetam a uma ação interativa, a interface fornecida apresenta ambas as situações, é possível encontrar no canto inferior direto abaixo da caixa de edição um *link* que carrega o usuário para realizar um novo cadastro, também é visível a imagem de uma digital que possibilita uma entrada alternativa para usuários que possuam aparelhos com essa tecnologia de autenticação.

Para ambos os casos, não é fornecido pelo sistema operacional uma semântica apropriada, pois ambos os elementos são declarados de forma genérica, então para que seja possível tornar-se acessível, é preciso atribuir as características assistivas de um objeto clicável em ambos os casos.

3.3 Aplicando elementos assistivos

Referenciando as problemáticas supracitadas anteriormente, foram desenvolvidas alternativas técnicas que solucionam os problemas encontrados, podendo destacar inicialmente a verbalização de componentes do tipo título, logo abaixo está disponibilizado uma função aberta que pode ser instanciada através do objeto *AccessibilityCommons*, presente no módulo *accessibility*.

[12] Idealização, projeção ou concepção do layout final.

Figura 6 – Código de aplicação, função delegada para acessibilidade de títulos

```
fun addTitleSemantics(view: View) {
    ViewCompat.setAccessibilityDelegate(
        view,
        object : AccessibilityDelegateCompat() {
            override fun onInitializeAccessibilityNodeInfo(
                host: View,
                info: AccessibilityNodeInfoCompat
            ) {
                super.onInitializeAccessibilityNodeInfo(host, info)
                info.isHeading = true
            }
        }
    )
}
```

Fonte: Autoria própria, 2023.

A função *addTitleSemantics* é responsável por configurar informações de acessibilidade para uma determinada *view* em um contexto de desenvolvimento de aplicativos móveis. Ela utiliza a classe *ViewCompat* para atribuir um *AccessibilityDelegate* personalizado. O *AccessibilityDelegate* é um mecanismo que permite a personalização das informações de acessibilidade de uma *view* para melhorar a experiência de usuários que dependem de tecnologias assistivas, como leitores de tela.

Dentro desse *AccessibilityDelegate*, a função substitui o método *onInitializeAccessibilityNodeInfo*, que é chamado durante a inicialização das informações de acessibilidade para a *view*. Ela mantém o comportamento padrão chamando o método correspondente na classe pai por meio de *super.onInitializeAccessibilityNodeInfo(host, info).*

Assim, a função configura a propriedade *isHeading* do objeto *info* como *true*, essa configuração indica que a *view* em questão representa um título ou cabeçalho, fornecendo um contexto semântico importante para os usuários de tecnologia assistiva. Essa ação visa aprimorar a compreensão e a navegabilidade da interface do usuário para esse público-alvo.

Seguindo o mesmo contexto de desenvolvimento para aplicação acima, foi desenvolvido uma função que pode ser aplicada aos elementos de interface que apresentam comportamento de botões, todavia são desenvolvidos com componentes diferentes de acordo com o *layout*, assim o resultado desta codificação se apresenta:

Figura 7 - Código de aplicação, função delegada para acessibilidade em botões

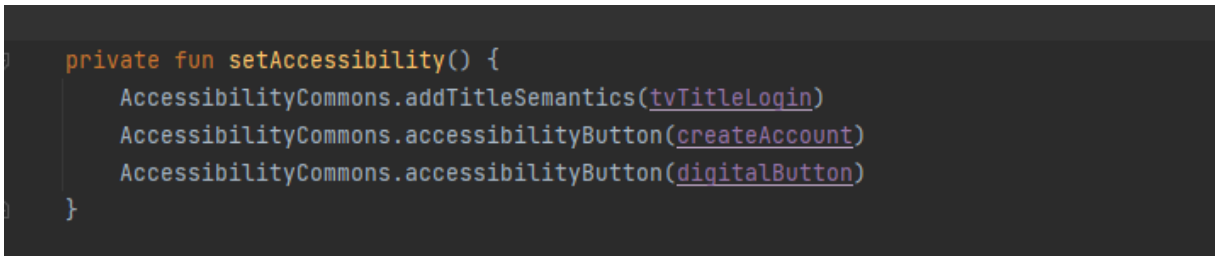
```
fun accessibilityButton(view: View) {
    ViewCompat.setAccessibilityDelegate(view, object : AccessibilityDelegateCompat() {
        override fun onInitializeAccessibilityNodeInfo(host: View, info: AccessibilityNodeInfoCompat) {
            super.onInitializeAccessibilityNodeInfo(host, info)
            info.isClickable = true
            info.className = "android.widget.Button"
        }
    })
}
```

Fonte: Autoria própria, 2023.

Neste caso o objeto *info* recebe a propriedade *isClickable* como *true*, indicando que o elemento de *view* é clicável pelo usuário, na linha subsequente ocorre a atribuição da *className* do objeto para classe pertencente aos botões, garantindo assim a definição correta do contexto.

Por fim para que seja possível utilizar as funções evidenciadas acima, o módulo *accessibility* deve ser importado no projeto desejado, neste caso foi adicionado a referência do arquivo *AAR*[13] no módulo que continha a aplicação *mobile*, posteriormente ao sincronizar o projeto, foi possível instanciar o objeto que contém as implementações desejadas, como podemos ver abaixo:

Figura 8 – Código de implementação das funções de acessibilidade



```
private fun setAccessibility() {
    AccessibilityCommons.addTitleSemantics(tvTitleLogin)
    AccessibilityCommons.accessibilityButton(createAccount)
    AccessibilityCommons.accessibilityButton(digitalButton)
}
```

Fonte: Autoria própria, 2023.

Como é possível observar ao instanciar o objeto *AccessibilityCommons* é necessário apenas encontrar a função desejada e adicionar ao seu construtor a *view* que precisa ser customizada.

[13] Archive do Android (AAR) é um formato de arquivo usado para distribuir bibliotecas Android.

4 RESULTADOS E DISCUSSÕES

Nesta sessão será discutido os resultados e conclusões acerca dos testes de usabilidade realizados após a implementação do módulo de acessibilidade, este que se encontra na plataforma *github* e pode ser acessado através do *link* github.com[14].

4.1 Configuração do ambiente de testes

Utilizando a *IDE*[15] *Android Studio* para alocação e desenvolvimento do código deste projeto, foi realizado a configuração do ambiente de testes através das ferramentas por ele disponibilizadas. Para a instalação do aplicativo foi necessário habilitar as configurações de desenvolvedor de um *device*[16], sendo ele um *hardware* com sistema operacional e interface *android* 12.

Após ocorrer o processo de instalação do *software*, foram mapeados os cenários e eventos que devem ser observados para concluir a efetividade das funções atribuídas.

E por fim foi preciso conferir a instalação dos recursos de acessibilidade inclusos do *hardware* a ser utilizado, sendo ele a versão 14.0.1 lançada pela equipe do sistema operacional *android*. Este projeto pode ser encontrado através do repositório github.com[17], contendo o aplicativo mencionado acima.

4.2 Aplicação de testes de *software*

Os testes manuais atribuídos neste projeto foram divididos e documentados em duas etapas, inicialmente foi evidenciado por meio da gravação de tela do *device*, o aplicativo sem a inserção de nenhum tipo de recurso assistivo aplicado ao código.

O intuito desta etapa inicial é proporcionar evidencias que exemplifiquem uma leitura de *talkback*, cujos elementos de interface não apresentem a semântica apropriada impedindo que o usuário tenha uma experiência satisfatória no uso do

[14] Disponível em <https://github.com/Kauane-SP/model-accessibility>. Acesso em: 03 nov. 2023.
[15] Ambiente de desenvolvimento integrado.
[16] Aparelho de hardware utilizado para testes.
[17] Disponível em <https://github.com/Kauane-SP/app-implementation-accessibility>. Acesso em: 03 nov. 2023.

software. Esta evidência pode ser acessada através da plataforma *youtube*[18] no *link* youtube.com[19]*.*

Em seguida foram mapeados os problemas encontrados nesta execução, sendo possível destacar a não verbalização dos títulos com a semântica correta, tornando a execução confusa e não padronizada, a semântica de objetos clicáveis sem a verbalização de botão e a não verbalização de ícones que apresentam interatividade com o usuário.

Posterior ao primeiro cenário, a aplicação agora é testada após as implementações do módulo *accessibility*, sua evidência através da gravação da tela foca principalmente na navegação linear da aplicação, após as correções necessárias. Assim, também é disponibilizada através do *link*: youtube.com[20], desta forma é possível comparar as mudanças ocorridas de uma execução para a outra.

5 CONCLUSÃO

Ao finalizar a implementação do projeto em um aplicativo nativo *android*, foi possível concluir que o módulo de acessibilidade desenvolvido no decorrer deste artigo proporciona um desenvolvimento facilitado de recursos assistivos.

Em poucos passos foi possível modificar a semântica de objetos fazendo com que as tecnologias assistivas do sistema operacional interpretem os elementos de *view* conforme o esperado. Contudo, é possível considerar não apenas as funcionalidades atribuídas, mas também a economia de linhas de código visando a qualidade e recursividade das funções.

Viabilizando a progressão deste projeto, os próximos passos serão mapear novas funcionalidades que podem ser atribuídas a este módulo, o tornando cada vez mais completo para os mais variados cenários, também buscar atribuir ao projeto uma estrutura que remeta a uma biblioteca *android*, facilitando o uso destes recursos criados a novos aplicativos.

[18] Plataforma de vídeos online.
[19] Disponível em <https://youtube.com/shorts/yHz4QI_-7_Q?feature=share>. Acesso em: 03 nov. 2023.
[20] Disponível em < https://youtu.be/S0nqoTZuFOo?feature=shared >. Acesso em: 03 nov. 2023.

6 REFERÊNCIAS

BROOKS, F. P. **O Mítico Homem-Mês**: Ensaios Sobre Engenharia de Software. Edição de 20º aniversário. Rio de Janeiro: Alta Books, 2018.

DELAMARO, Marcio; JINO, Mario; MALDONADO, Jose. **Introdução ao teste de software**. Elsevier Brasil, 2013.

EDYBURN, Dave L. **Rethinking assistive technology**. Special Education Technology Practice, v. 5, n. 4, p. 16-23, 2004.

OLIVEIRA, Andre Lucio et al. Um estudo sobre o sistema operacional android**. REVISTA DE TRABALHOS ACADÊMICOS-CAMPUS NITERÓI**, n.1, 2014.

LOPES, Paulo Ricardo da Silva. **Verificação do Nível de Acessibilidade de Ferramentas para Desenvolvimento de Software por um Desenvolvedor Baixa Visão com a Ferramenta de Ampliação de Tela**, 2023, Trabalho de Conclusão de Curso (Engenharia da Computação) - Universidade Federal do Ceará, Ceará, 2023.

ELY, Leo. **App Bundle e Dynamic Delivery**: modulação de apps para Android. 2019.

HAKOBYAN, Lilit et al. **Mobile assistive technologies for the visually impaired**. Survey of ophthalmology, v. 58, n. 6, p. 513-528, 2013.

NAZARI, Ana Clara Gomes; NAZARI, Juliano; GOMES, Maria Aldair. **Tecnologia Assistiva (TA): do conceito a legislação–discutindo a TA enquanto política de educação inclusiva que contribui na formação e inclusão de pessoas com deficiência**. In: V CONGRESSO DE PSICOPEDAGOGIA ESCOLAR EI ENCONTRO DE PESQUISADORES EM PSICOPEDAGOGIA ESCOLAR. 2017. p. 1-16.