
Universidade Federal de Pernambuco
Centro de Informática

# Detecção Automática de Falhas de Localização em Aplicativos Android

Matheus Epitácio Barros de Lucena

Recife
Março, 2024

Universidade Federal de Pernambuco
Centro de Informática

Matheus Epitácio Barros de Lucena

# Detecção Automática de Falhas de Localização em Aplicativos Android

*Trabalho de graduação apresentado para o programa de Bacharelado em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco em cumprimento parcial dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.*

Orientador: *Prof. Dr. Breno Alexandro Ferreira de Miranda*

Recife
Março, 2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Lucena, Matheus Epitácio Barros de.
Detecção Automática de Falhas de Localização em Aplicativos Android / Matheus Epitácio Barros de Lucena. - Recife, 2024.
34 p. : il., tab.

Orientador(a): Breno Alexandro Ferreira de Miranda
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2024.

1. aplicativos mobile. 2. localização de software. 3. reconhecimento ótico de caractares. I. Miranda, Breno Alexandro Ferreira de. (Orientação). II. Título.

000 CDD (22.ed.)

**MATHEUS EPITÁCIO BARROS DE LUCENA**

# DETECÇÃO AUTOMÁTICA DE FALHAS DE LOCALIZAÇÃO EM APLICATIVO ANDROID

Trabalho de graduação apresentado para o programa de Bacharelado em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco em cumprimento parcial dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Data de Aprovação: _25_/_03_/_2024_

Nota: _____

BANCA EXAMINADORA:

_____
Prof. Dr. Breno Alexandro Ferreira de Miranda (Orientador)
Centro de Informática - UFPE

_____
Prof. Dr. Kiev Santos da Gama (1º Titular)
Departamento de Botânica - UFPE

RECIFE
2024

# Abstract

Translating mobile apps into different languages is a key strategic step for companies looking to reach international users. However, this process can have mistakes that might go unseen because of language barriers and the complicated nature of interfaces. The present work introduces an automated tool designed to detect missing translations within mobile app interfaces by analyzing screenshots via optical character recognition and comparing them with XML files that contain the interface's hierarchy. The system makes a report of the errors found, giving testers a practical way to spot possible mistakes. The tool's effectiveness was confirmed by testing it with apps that are available in the market.

**Keywords:** mobile applications, software localization, optical character recognition.

# Resumo

A tradução de aplicativos móveis para diferentes idiomas é um passo estratégico essencial para empresas que almejam alcançar usuários internacionais. No entanto, esse processo é suscetível a erros que podem passar despercebidos devido à barreira do idioma e à complexidade das interfaces. O presente trabalho propõe uma ferramenta automatizada para detectar falhas semânticas de tradução em interfaces de aplicativos móveis, analisando capturas de tela através de reconhecimento ótico de caracteres e comparando-as com arquivos XML que contém a hierarquia da interface. O sistema gera um relatório das inconsistências encontradas, oferecendo um meio prático para testadores encontrarem possíveis erros. A validade da ferramenta foi confirmada por meio de testes com aplicativos disponíveis no mercado.

**Palavras-chave:** aplicativos mobile, localização de software, reconhecimento ótico de caracteres

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Siglas

**APK** Android Application Pack.

**I18N** Internationalization.

**L10N** Localization.

**OCR** Optical Character Recognition.

**UI** User Interface.

**XML** Extensible Markup Language.

CAPÍTULO 1

# Introdução

No panorama atual do mercado global, empresas de tecnologia de todo o mundo esforçam-se para expandir o alcance de seus produtos para além das fronteiras nacionais. Uma estratégia fundamental para alcançar este objetivo é a localização [1] de aplicativos móveis, um processo que envolve a adaptação de software para atender às necessidades e preferências de usuários em diferentes regiões geográficas. Contudo, a localização traz consigo desafios substanciais [2]. Dificuldades linguísticas e complexidades nas interfaces de usuário podem resultar em traduções imprecisas, muitas vezes não perceptíveis por desenvolvedores que não dominam a língua do público-alvo. Esses equívocos, se não identificados e corrigidos, têm o potencial de comprometer a experiência do usuário e, por extensão, prejudicar a reputação do aplicativo.

Diante da necessidade de um método mais eficiente e confiável para identificar e corrigir falhas de localização, este trabalho propõe o desenvolvimento de uma ferramenta dedicada especificamente a essa finalidade. A relevância deste trabalho é fundamentada pelos desafios identificados por Awwad e Slany [3], que destacam a crescente atenção à automação dos testes de localização de aplicativos. Além disso, a prática comum de testes manuais e esporádicos não consegue cobrir eficientemente a vasta gama de idiomas e nuances culturais presentes na localização de aplicativos globais. Este trabalho busca preencher essa lacuna ao desenvolver uma ferramenta que se concentra especificamente na detecção de ausências de tradução em aplicativos móveis.

Para identificar falhas de tradução, a ferramenta utiliza capturas de tela dos aplicativos, arquivos XML que detalham a hierarquia da interface do usuário (UI) e um dicionário contendo palavras no idioma-alvo. Analisando os arquivos XML em busca de elementos textuais e comparando-os com o dicionário, a ferramenta é capaz de identificar termos não traduzidos. Posteriormente, a ferramenta utiliza de reconhecimento óptico de caracteres (OCR)[4] para localizar essas palavras nas capturas de tela, destacando as falhas.

Após o desenvolvimento, a ferramenta foi submetida a uma avaliação com aplicativos reais, demonstrando resultados promissores. Este trabalho está estruturado da seguinte maneira: a Seção 2 apresenta o contexto e trabalhos relacionados. A Seção 3 detalha a ferramenta e descreve o desenvolvimento da mesma. A avaliação empírica da ferramenta é descrita na Seção 4 e os resultados da avaliação em aplicativos reais são apresentados na Seção 5. Discussões sobre os resultados e suas implicações são exploradas na Seção 6. Por fim, a Seção 7 conclui o trabalho, ressaltando as principais contribuições e sugerindo direções para pesquisas futuras.

CAPÍTULO 2

# Contexto e Trabalhos Relacionados

## 2.1 Internacionalização e Localização

A internacionalização (i18n) e a localização (l10n) são conceitos chave no desenvolvimento de software destinados a mercados globais [1]. A internacionalização, ou i18n, descreve o processo de preparar um aplicativo para suportar múltiplas línguas e culturas, implementando uma estrutura que permite futuras adaptações. Isso envolve a abstração de todos os elementos específicos de idioma e cultura do código, facilitando a adaptação do software para diferentes idiomas e regiões de maneira eficiente.

A localização, abreviada como l10n, representa a etapa seguinte de adaptar um software internacionalizado para um mercado particular. Este processo vai além da mera tradução de textos para o idioma alvo, envolvendo também a customização conforme as peculiaridades culturais, legais e técnicas da região alo. O objetivo da l10n é assegurar que o software não apenas comunique na língua do usuário, mas também esteja alinhado às suas tradições culturais e atenda às legislações locais.

O grande desafio ao preparar um software para mercados diferentes, através da internacionalização e localização, é fazer com que ele se encaixe nas particularidades de cada lugar, incluindo a língua, cultura e leis. Isso pode significar alterações importantes na aparência do software, na forma como ele funciona e na apresentação dos dados. Um obstáculo adicional é que os desenvolvedores muitas vezes não conhecem bem as línguas e culturas dos lugares para os quais estão adaptando seus produtos, destacando a necessidade de usar ferramentas especiais e trabalhar com especialistas dessas regiões.

Esses esforços são cruciais não só para traduzir os textos corretamente, mas também para ajustar detalhes como formatos de data e hora, moeda, práticas culturais e normas locais. Dessa forma, garante-se que o software seja verdadeiramente útil e fácil de usar para as pessoas de diferentes partes do mundo.

## 2.2 Principais Falhas de Localização

Durante o processo de localização, diversos erros podem comprometer a usabilidade e a experiência do usuário com o software. Algumas falhas mais comuns incluem:

- **Ausência de Tradução:** Segmentos de texto que não são traduzidos para o idioma alvo, permanecendo na língua original do aplicativo.


Figura 2.1: Exemplo de ausência de tradução

- **Elipse:** Ocorre quando o texto traduzido é muito longo para o espaço alocado, resultando em um corte com pontos de suspensão ("..."), o que pode ocultar informações importantes.

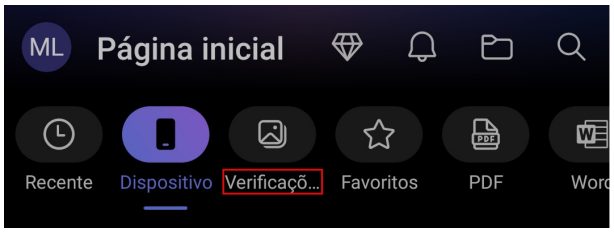



Figura 2.2: Exemplo de elipse

- **Truncamento:** Similar à elipse, mas neste caso, o texto é cortado abruptamente sem os pontos de suspensão. Isso pode tornar o texto incompreensível ou deixar a mensagem incompleta.

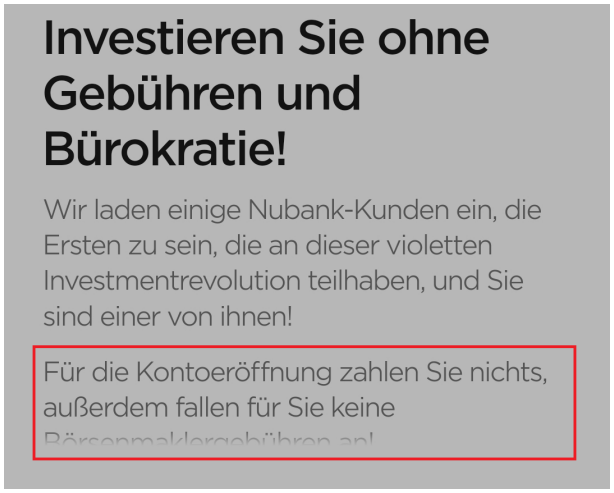



Figura 2.3: Exemplo de truncamento

- **Sobreposição:** Acontece quando textos ou elementos gráficos se sobrepõem devido ao layout não adaptado para o idioma, afetando negativamente a legibilidade e a estética do aplicativo.


Figura 2.4: Exemplo de sobreposição

## 2.3 Hierarquia da Interface do Usuário

Dentro do contexto de análise e desenvolvimento de interfaces de usuário (UI) para aplicações Android, a compreensão da hierarquia da UI é crucial. Uma ferramenta eficaz para essa análise é o uso do comando adb shell uiautomator dump, parte do Android Debug Bridge (ADB) e do framework UIAutomator. Este comando permite a extração da estrutura atual da UI de uma aplicação em um arquivo XML em runtime, ou seja, enquanto a aplicação está sendo executada no dispositivo. O arquivo gerado detalha cada elemento presente na interface, incluindo propriedades como identificadores, posições e estados, oferecendo uma visão completa de como os componentes da UI estão organizados e interagem entre si. A hierarquia da UI capturada reflete a árvore de elementos da interface, facilitando a identificação de padrões de design, problemas de acessibilidade, ou potenciais obstáculos na experiência do usuário.

A hierarquia da UI capturada reflete a árvore de elementos da interface, facilitando a identificação de padrões de design, problemas de acessibilidade, ou potenciais obstáculos na experiência do usuário.


Figura 2.5: Exemplo de tela e XML com a hierarquia da UI correspondente

## 2.4 Reconhecimento Óptico de Caracteres

O Reconhecimento Óptico de Caracteres, conhecido pela sigla em inglês OCR (Optical Character Recognition), refere-se ao processo tecnológico capaz de identificar e converter caracteres presentes em uma imagem em texto [4]. Essa técnica pode ser empregada através de diversas

estratégias, sendo a aprendizagem profunda uma das abordagens mais prevalentes atualmente devido à sua eficácia na interpretação de complexidades visuais e textuais.

A tarefa de reconhecimento óptico de caracteres enfrenta múltiplos desafios, em grande parte devido à ampla variação na forma como os caracteres podem ser apresentados. Variações na fonte, tamanho, estilo, e até a distorção provocada por condições de iluminação ou qualidade da imagem, podem impactar significativamente a precisão do OCR. Adicionalmente, a diversidade linguística global introduz um espectro ainda mais amplo de caracteres a serem reconhecidos, ampliando os desafios enfrentados pelos sistemas de OCR em sua capacidade de oferecer soluções universalmente aplicáveis e precisas.[5]

## 2.5 Trabalhos Relacionados

Alameer, et al [6] desenvolveram uma técnica inovadora para a detecção de erros de apresentação em páginas web causadas pelo processo de localização e internacionalização. O presente trabalho apresenta uma ferramenta para encontrar erros em aplicativos Android.

Escobar-Velásquez, et al [7] apresentaram o ITDroid, uma ferramenta avançada criada para detecção automática de falhas de internacionalização em aplicativos Android. A ferramenta realiza uma análise do arquivo APK do aplicativo, identificando textos que estão sem tradução e gerando, de forma automática, um novo arquivo APK traduzido. A ferramenta também analisa a UI para encontrar erros de apresentação causados pelo processo de localização. O presente trabalho se diferencia por apenas analisar as capturas de tela e arquivos XML com a hierarquia da UI, sem precisar verificar o APK do aplicativo.

Couto and Miranda [8], [9] introduziram o l10n-trainer, uma ferramenta projetada para auxiliar no treinamento de testadores novatos em i18n/l10n, permitindo que os treinadores insiram intencionalmente falhas de i18n/l10n em um conjunto selecionado de aplicativos Android.

Felipe and Miranda [10] propuseram uma ferramenta para apoiar testes de i18n/l10n em aplicativos Android. O objetivo da ferramenta proposta é auxiliar testadores a reduzir o tempo necessário para realizar testes manuais e aumentar a cobertura dos testes.

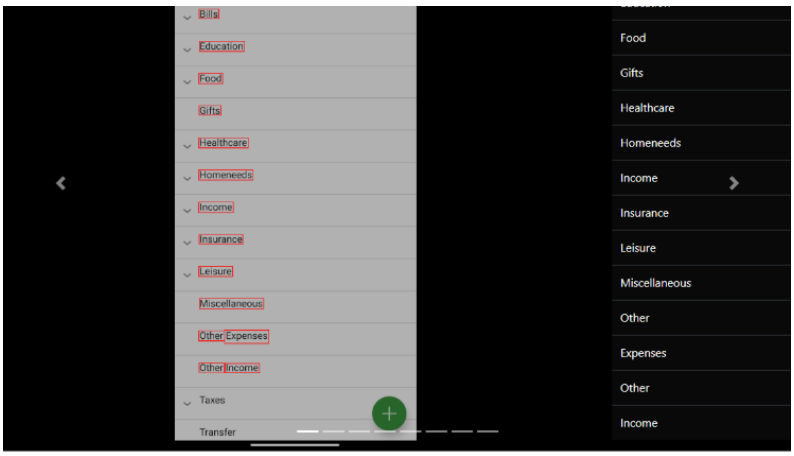
CAPÍTULO 3

# Ferramenta

## 3.1 Descrição da Ferramenta

Esta seção detalha a ferramenta desenvolvida para identificar ausências de tradução em aplicativos Android, utilizando capturas de tela e arquivos XML que representam a hierarquia da interface do usuário. A ferramenta compara o texto extraído desses arquivos com um dicionário do idioma alvo e com uma lista de palavras que devem ser ignoradas, marcando as palavras ausentes como não traduzidas. Além dos dicionários padrão em português e espanhol, o usuário pode fornecer dicionários personalizados em outros idiomas. O arquivo

Para facilitar a identificação visual das falhas, a ferramenta emprega OCR nas capturas de tela, localizando as palavras marcadas como não traduzidas. Os resultados são apresentados em um relatório HTML, que inclui as imagens com as falhas destacadas e uma tabela de ocorrências de palavras não traduzidas. A ferramenta também permite que o usuário ignore palavras específicas, como termos técnicos ou nomes próprios, que não requerem tradução. Um exemplo do relatório gerado pela ferramenta, que destaca falhas de tradução, pode ser visto na Figura 3.1.




Count of missing translations

| Word | Count | Actions |
|---|---|---|
| automobile | 1 | |
| bills | 1 | |
| education | 1 | |

Figura 3.1: Exemplo de relatório gerado pela ferramenta

## 3.2 Parâmetros da Ferramenta

Os parâmetros da ferramenta são os seguintes:

- **xml_dir**: Este parâmetro especifica o diretório onde estão localizados os arquivos XML contendo as hierarquias da interface da UI ao aplicativo.
- **screenshot_dir**: Define o diretório que contém todas as capturas de tela do aplicativo.
- **lang**: Identifica o idioma alvo do aplicativo.
- **dic**: Especifica o caminho para o arquivo de texto que contém as palavras do idioma alvo.
- **ignore**: Indica o caminho para o arquivo de texto com as palavras que devem ser ignoradas durante a análise.
- **output_dir**: Define o diretório de saída da ferramenta, onde os resultados da análise serão armazenados.

## 3.3 Desenvolvimento da Ferramenta

Inicialmente, o objetivo da ferramenta era identificar falhas de tradução apenas com base nas capturas de tela dos aplicativos, aplicando extensivamente o OCR para extrair texto. No entanto, desafios com a precisão do OCR, que frequentemente confundia caracteres similares (como 'o' e '0'), levaram à busca por alternativas. A solução encontrada foi o uso do comando `adb shell uiautomator dump`, que extrai com precisão o texto das interfaces de usuário em arquivos XML, aumentando a confiabilidade da ferramenta, mas adicionando uma etapa adicional para os usuários.

A ideia de usar APIs de tradução para identificar textos não traduzidos foi considerada. Durante o desenvolvimento, a ferramenta foi capaz de se integrar com a API do Google Translate para detectar idiomas. Contudo, para evitar custos adicionais aos usuários, decidiu-se por uma abordagem que necessita de um dicionário do idioma alvo fornecido pelo usuário. Dicionários para o português e o espanhol foram incorporados à ferramenta, provenientes de fontes confiáveis [11] [12].

Embora o OCR não fosse mais utilizado para a extração de texto direta das imagens, ele se tornou essencial para identificar visualmente as falhas de tradução nas capturas de tela. Neste contexto, a EasyOCR [13] destacou-se entre as bibliotecas testadas pela sua eficácia e suporte a múltiplos idiomas. No entanto, um desafio significativo foi o tempo de processamento nas máquinas sem aceleração gráfica, chegando a quase 40 segundos por imagem. Para mitigar isso, implementou-se um sistema de cache, garantindo que cada imagem fosse processada pelo OCR apenas uma vez, reduzindo significativamente o tempo total de análise em execuções subsequentes da ferramenta.

A necessidade de excluir certas palavras da análise, como nomes de empresas ou termos técnicos que não requerem tradução, levou à implementação de uma funcionalidade que permite aos usuários definir uma lista de palavras a serem ignoradas. Esse ajuste refinou ainda mais a precisão da ferramenta, permitindo personalização conforme o contexto de uso específico de cada aplicativo.

## 3.4 Validação em Ambiente Controlado

Uma validação preliminar da ferramenta foi realizada em um ambiente controlado. Para isso, um aplicativo foi intencionalmente editado para incluir falhas de tradução específicas. Este

método permitiu uma avaliação detalhada da capacidade da ferramenta em detectar e relatar falhas de tradução de forma confiável. Um exemplo desse teste é apresentado na Figura 3.2, que mostra a detecção de uma palavra não traduzida - "amount" - na interface do usuário do aplicativo. Este teste destaca a precisão da ferramenta em capturar falhas de tradução em um cenário simulado, preparando o caminho para testes em aplicativos reais.

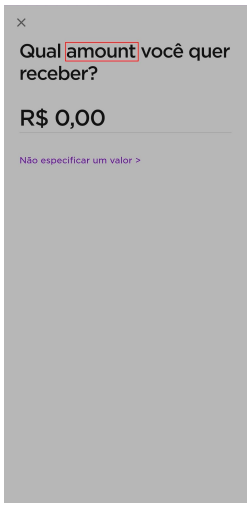



Figura 3.2: Exemplo de falha capturada pela ferramenta durante o desenvolvimento

CAPÍTULO 4

# Avaliação

## 4.1 Introdução à Avaliação Empírica

Após a validação inicial em um ambiente controlado, a próxima etapa consistiu em testar a ferramenta em aplicativos reais. Cinco aplicativos foram escolhidos para esta fase de testes, com o intuito de avaliar a eficácia da ferramenta em diferentes contextos e configurações. Três destes aplicativos foram obtidos da Google Play Store e dois da F-Droid, representando um espectro diversificado de uso e popularidade. Esta seção detalha a metodologia aplicada na avaliação empírica, descrevendo os parâmetros de execução e as bases para a análise dos resultados.

## 4.2 Metodologia

Esta seção descreve o procedimento utilizado para avaliar empiricamente a ferramenta proposta. A abordagem foi organizada em etapas para assegurar uma coleta de dados estruturada e uma análise objetiva da eficiência da ferramenta.

### Critério de Seleção dos aplicativos

Para a seleção dos aplicativos da Google Play Store, os critérios incluíram a diversidade de áreas para cobrir um amplo espectro de contextos de uso. Os aplicativos selecionados foram aqueles disponíveis em vários idiomas, incluindo o português, para testar a eficácia da ferramenta na detecção de falhas de tradução, apesar de a avaliação final ter sido conduzida especificamente em português. Priorizou-se também aplicativos com uma grande base de usuários, indicando alta relevância e demanda, o que poderia introduzir uma maior complexidade nas interfaces de usuário analisadas.

Para os aplicativos da F-Droid, a escolha foi direcionada para aqueles disponíveis em múltiplos idiomas, incluindo o português, para manter a consistência com os critérios da Play Store. Um aspecto adicional considerado foi a frequência de atualizações. Dado o caráter de código aberto dos aplicativos da F-Droid, foi possível analisar as últimas atualizações, selecionando-se aplicativos que tiveram mudanças recentes.

### Seleção e Download dos Aplicativos

Inicialmente, cinco aplicativos de diferentes categorias e fontes foram selecionados. Os aplicativos escolhidos foram então baixados para um dispositivo de teste.

### Exploração e Coleta de Dados

Para cada aplicativo, realizou-se uma navegação manual, com o objetivo de explorar o máximo de telas diferentes possíveis para cada aplicativo, a fim de obter uma amostra representativa

da interface do usuário. Durante esta navegação, para cada tela distinta,. Durante esta navegação, para cada tela diferente, executou-se o comando `adb shell uiautomator dump` para capturar a hierarquia da interface de usuário em um arquivo XML. Após a execução do comando, realizou-se uma captura de tela correspondente. Registrou-se o número total de telas exploradas e o tempo total da exploração de cada aplicativo, fornecendo uma base quantitativa para a análise subsequente.

#### Execução da Ferramenta

Com os dados coletados, executou-se a ferramenta em duas rodadas para cada aplicativo: uma sem utilizar o arquivo de palavras a ignorar e outra com o arquivo configurado. Isso possibilitou a comparação do desempenho da ferramenta sob as duas condições diferentes.

#### Avaliação e Registro de Resultados

Após a execução, contabilizou-se a quantidade de palavras identificadas como não traduzidas e o número de telas que apresentaram falhas. Estes dados foram registrados para análise comparativa entre as execuções com e sem o arquivo de palavras a ignorar.

Este método forneceu um panorama detalhado da capacidade da ferramenta em identificar falhas de tradução e ajudou a entender a influência do arquivo *ignore* na precisão dos resultados.

## 4.3 Seleção dos Aplicativos

A seleção dos aplicativos para a avaliação empírica foi cuidadosamente planejada para incluir uma diversidade de usos e públicos-alvo, escolhendo aplicativos de duas principais fontes de distribuição: Google Play Store e F-Droid. Essas plataformas foram selecionadas por representarem ecossistemas distintos no universo de aplicativos Android.

### 4.3.1 Google Play Store

A Google Play Store, a principal loja de aplicativos para dispositivos Android, oferece uma vasta gama de aplicativos, desde soluções corporativas a ferramentas educacionais. Os aplicativos selecionados desta loja foram escolhidos por sua ampla disponibilidade em diversas línguas e por receberem constantes atualizações, o que os torna candidatos ideais para testar a robustez da ferramenta de detecção de falhas de tradução. Os aplicativos escolhidos foram:

- **Microsoft 365**: Um aplicativo abrangente para criação e edição de documentos, planilhas e apresentações, amplamente utilizado no ambiente corporativo e educacional.
- **Notion**: Uma ferramenta de organização e produtividade que permite aos usuários criar notas customizadas, bases de dados e muito mais, favorecido por sua flexibilidade.
- **Kahoot**: Uma plataforma de aprendizagem baseada em jogos, utilizada por educadores para criar quizzes interativos que engajam os alunos de maneira divertida.

### 4.3.2 F-Droid

Diferentemente da Play Store, a F-Droid foca em software livre e de código aberto, atraindo usuários interessados em privacidade e transparência. Os aplicativos da F-Droid selecionados

compartilham a característica de serem localizados, disponíveis em várias línguas, e recebem atualizações que os mantêm relevantes e funcionais. Incluem:

- **Blood Pressure Monitor**: Um aplicativo simples e eficaz para registrar medições de pressão arterial, permitindo aos usuários acompanhar sua saúde de maneira prática.
- **MoneyManagerEx**: Um aplicativo de gerenciamento de finanças pessoais que oferece funcionalidades para rastrear despesas, rendimentos e planejar orçamentos, ajudando os usuários a gerir suas finanças com maior controle.

A escolha desses aplicativos, todos caracterizados por sua localização e recebimento constante de atualizações, visa fornecer um teste abrangente da ferramenta em um espectro amplo de cenários de uso e necessidades dos usuários.

## 4.4 Limitações da Avaliação

Uma limitação significativa deste estudo é o número relativamente limitado de aplicativos testados. Com a análise concentrada em apenas cinco aplicativos, sendo três da Google Play Store e dois da F-Droid, a amostra pode não ser suficientemente representativa de toda a gama de aplicativos disponíveis em ambas as plataformas. Esta seleção, embora diversificada em termos de categorias e fontes de distribuição, pode não capturar completamente a variedade de contextos de uso, design de interface e desafios de localização presentes no vasto ecossistema de aplicativos móveis.

Além disso, a avaliação foi realizada sob condições controladas, com a exploração dos aplicativos e a coleta de dados sendo executadas manualmente. Esta abordagem pode introduzir um viés na seleção de telas e funcionalidades exploradas, limitando potencialmente a abrangência da avaliação.

CAPÍTULO 5

# Resultados

A tabela 5.1 resume os resultados da aplicação da ferramenta em cinco diferentes aplicativos, focando exclusivamente nas métricas temporais relacionadas à exploração dos aplicativos e à execução da ferramenta. Ela oferece uma visão quantitativa sobre o tempo de exploração de cada aplicativo e o tempo de execução da ferramenta.

Tabela 5.1: Resumo dos Tempos de Exploração e Execução por Aplicativo

| **Aplicativo** | **Quantidade de Telas** | **Tempo de Exploração** | **Tempo de Execução** |
|---|---|---|---|
| Notion | 19 telas | 23 min | 1 min 5 seg |
| Kahoot | 27 telas | 37 min | 1 min 27 seg |
| Microsoft 365 | 17 telas | 23 min | 2 min 22 seg |
| Blood Pressure Monitor | 11 telas | 16 min | 37 seg |
| MoneyManagerEX | 17 telas | 27 min | 1 min 2 seg |

A Tabela 5.2 resume os resultados obtidos na identificação de palavras não traduzidas e a quantidade de telas que apresentaram falhas nos aplicativos avaliados, com e sem a aplicação do arquivo de ignorar palavras específicas.

Tabela 5.2: Resumo de Palavras Não Traduzidas e Telas com Falha por Aplicativo

| **Aplicativo** | **Sem *Ignore*** | | **Com Ignore** | |
|---|---|---|---|---|
| | **Quant. Palavras Não Traduzidas** | **Quant. Telas com Falhas** | **Quant. Palavras Não Traduzidas** | **Quant. Telas com Falha** |
| Microsoft 365 | 163 | 17 | 0 | 0 |
| Kahoot | 180 | 26 | 19 | 4 |
| Notion | 179 | 19 | 132 | 6 |
| Blood Pressure Monitor | 108 | 8 | 17 | 6 |
| MoneyManagerEX | 115 | 14 | 75 | 8 |

## 5.1 Resultados Kahoot

Nas Figuras 5.1 e 5.2, são mostrados algumas falhas no aplicativo Kahoot identificadas pela ferramenta, antes e após a aplicação do arquivo *Ignore*, respectivamente. As palavras não traduzidas identificadas pela ferramenta foram: acceleration, follow, system, geometry, multiplication, academy, verified, educators, creators, based, on, previous, searches, e also.

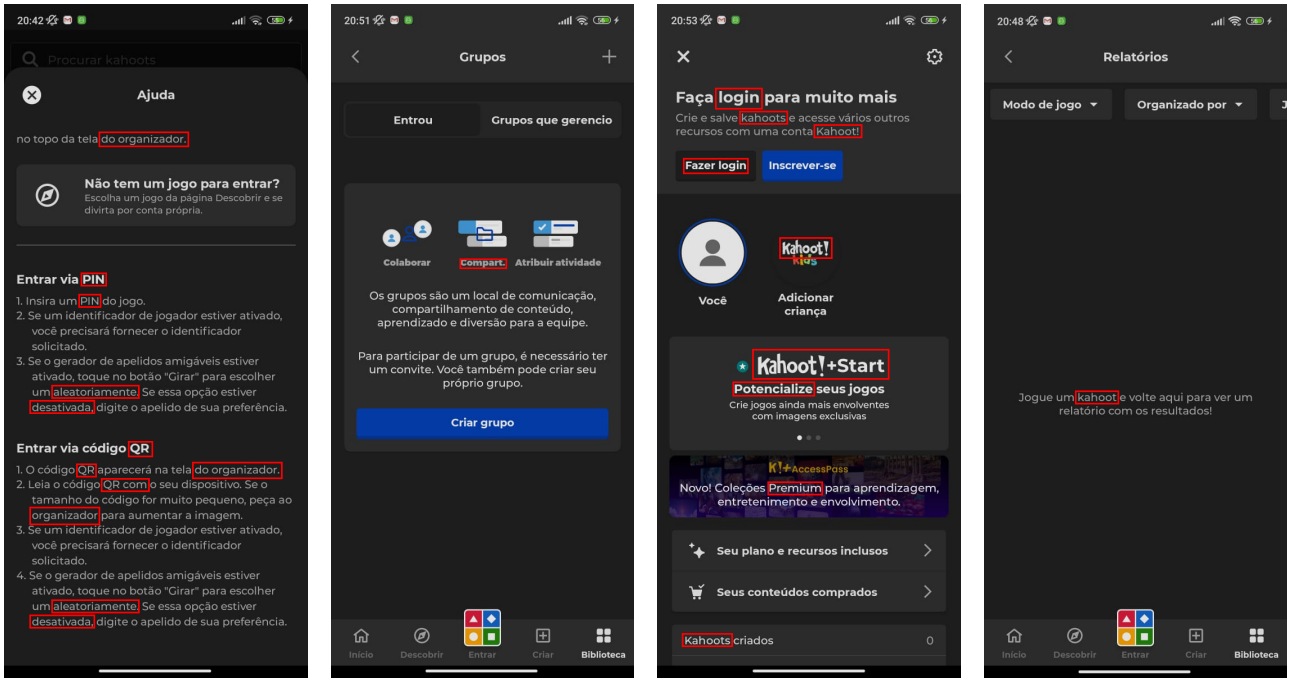



Figura 5.1: Falhas encontradas no aplicativo Kahoot sem o arquivo *Ignore*

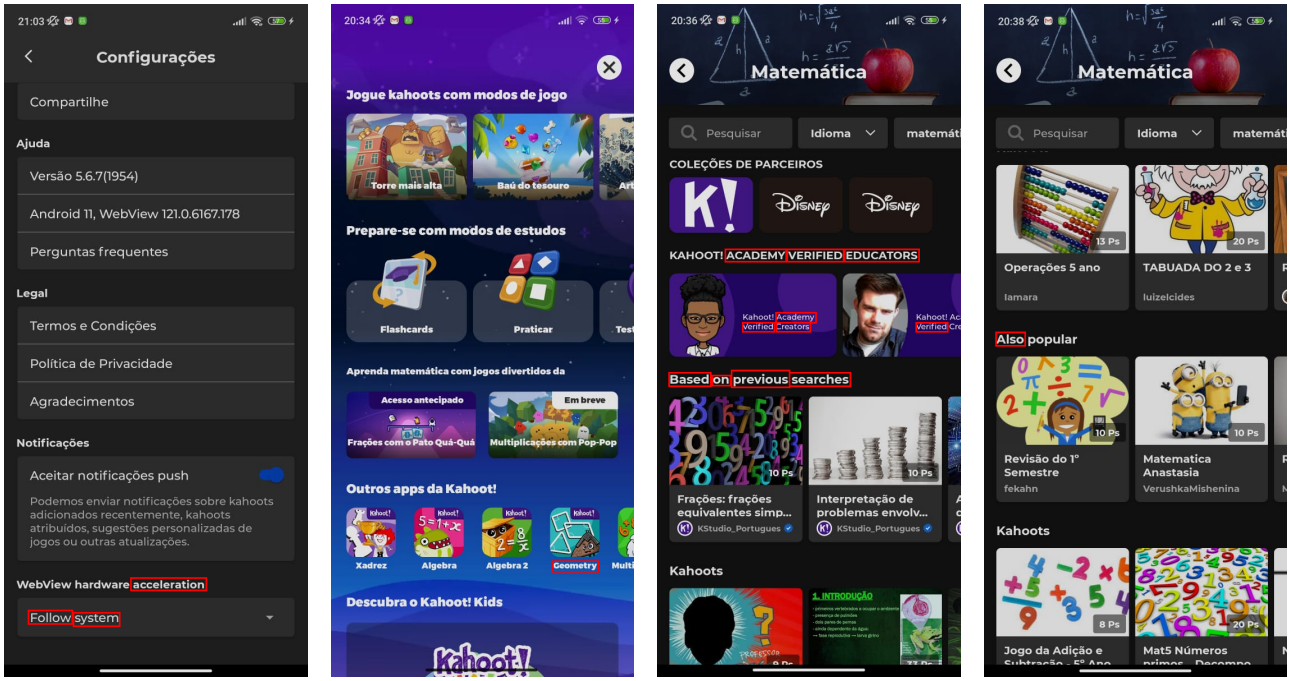



Figura 5.2: Falhas encontradas no aplicativo Kahoot com o arquivo *Ignore*

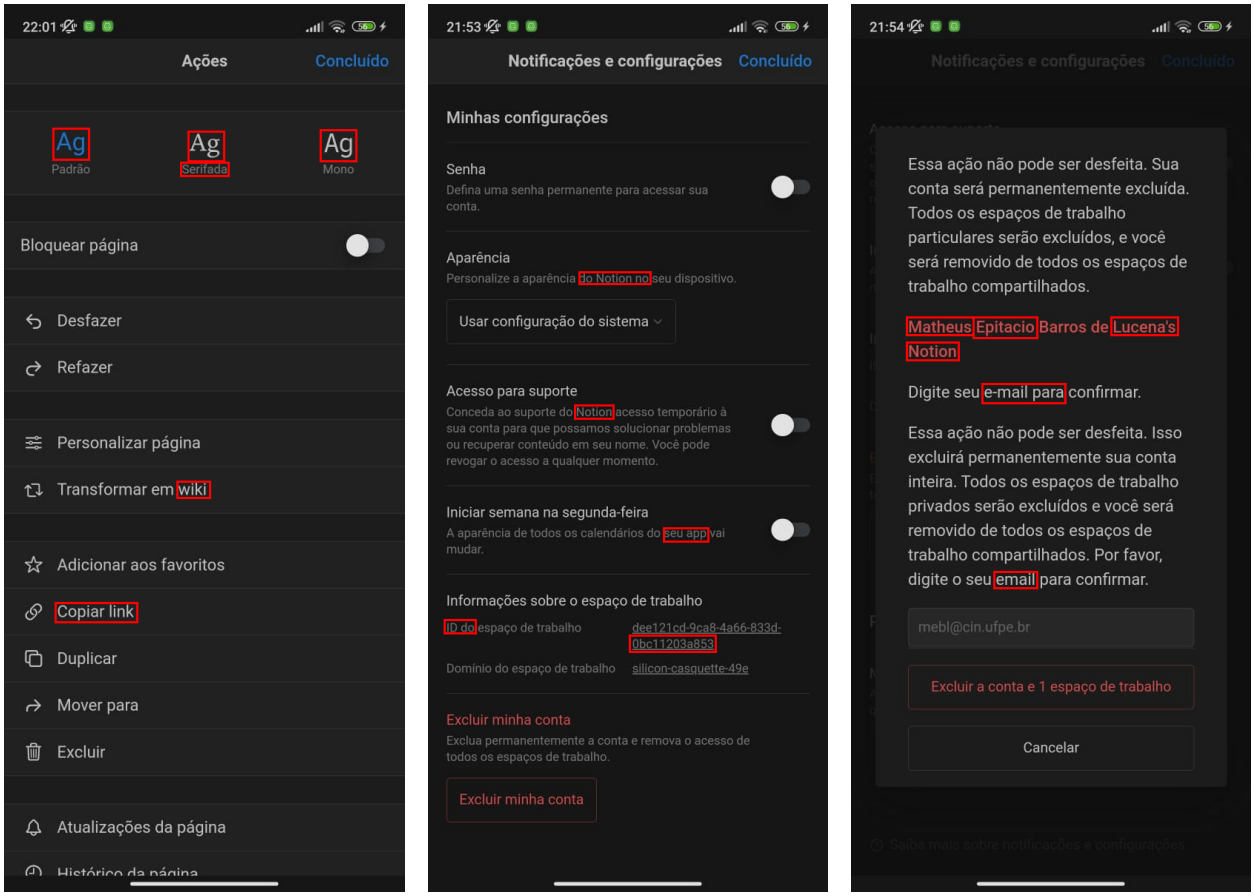



Figura 5.3: Falhas encontradas no aplicativo Notion sem o arquivo *Ignore*

## 5.2 Resultados Notion

Nas Figuras 5.3 e 5.4, são mostrados algumas falhas no aplicativo Notion identificados pela ferramenta, antes e após a aplicação do arquivo *Ignore*, respectivamente. As palavras não traduzidas identificadas pela ferramenta foram: notify, connect, docs, team, home, tasks, example, page, task, list, all, meetings, by, type, in, review, created, stakeholders, include, whats, new, mission, objectives, board, assignee, assigned, calendar, view, e popover.

## 5.3 Resultados Microsoft 365

A figura 5.5 ilustra alguns possíveis Falhas encontradas pela ferramenta no aplicativo Microsoft antes da aplicação do arquivo *Ignore*. Não foram detectadas falhas após a aplicação do arquivo.

## 5.4 Resultados Blood Pressure Monitor

Nas Figuras 5.6 e 5.7, são mostrados algumas falhas no aplicativo Blood Pressure Monitor identificados pela ferramenta, antes e após a aplicação do arquivo *Ignore*, respectivamente. As palavras não traduzidas identificadas pela ferramenta foram: medications, value, distribution, fields, preset, manage, columns, record, null, build, in, add, e medication.

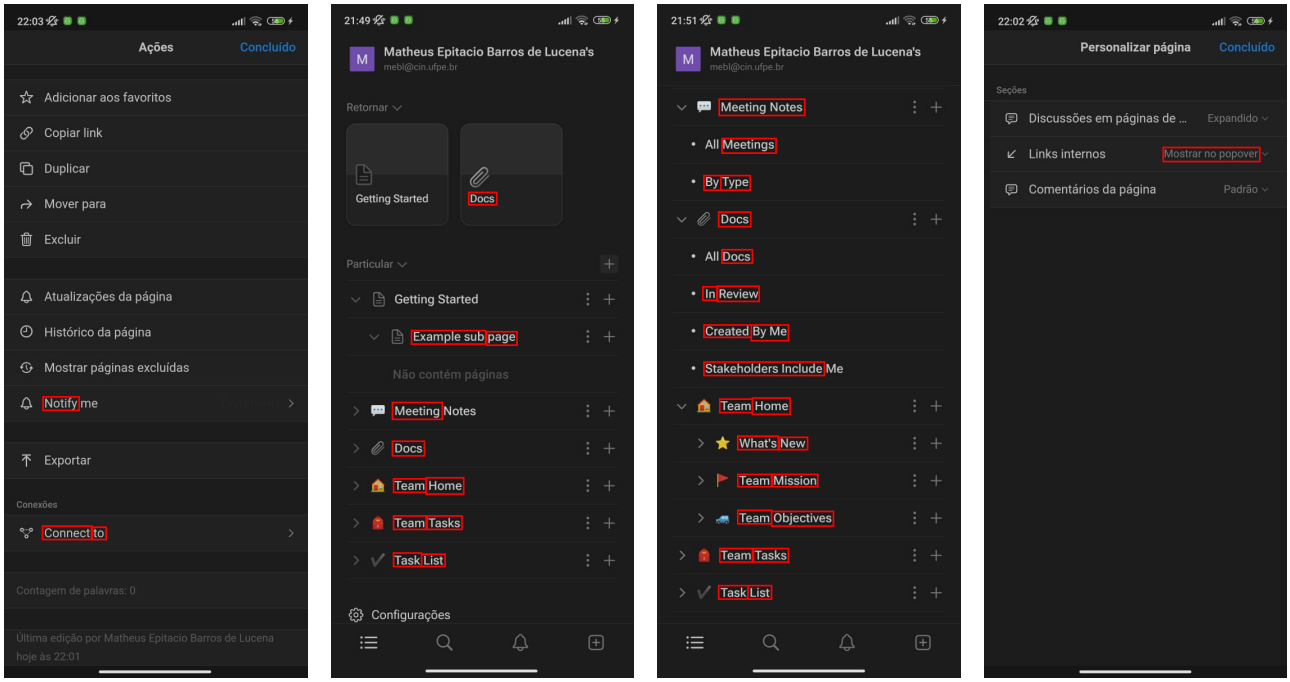



Figura 5.4: Falhas encontradas no aplicativo Notion com o arquivo *Ignore*

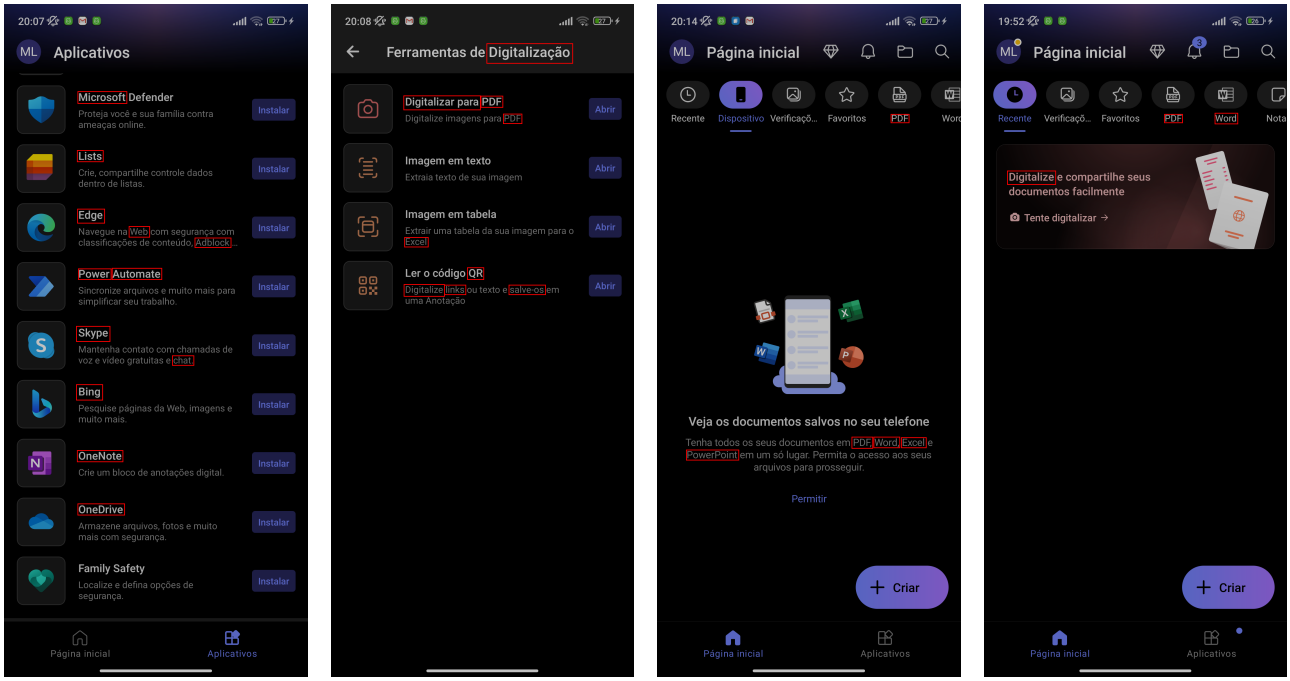



Figura 5.5: Falhas encontradas no aplicativo Microsoft 365 sem o arquivo *Ignore*

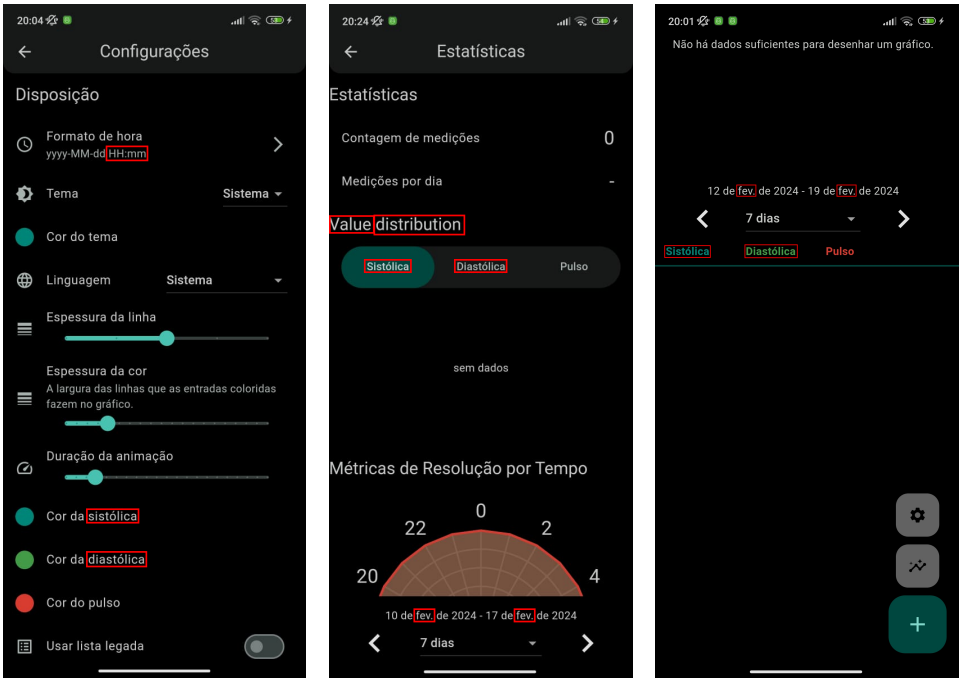



Figura 5.6: Falhas encontradas no aplicativo Blood Pressure Monitor sem o arquivo ignore

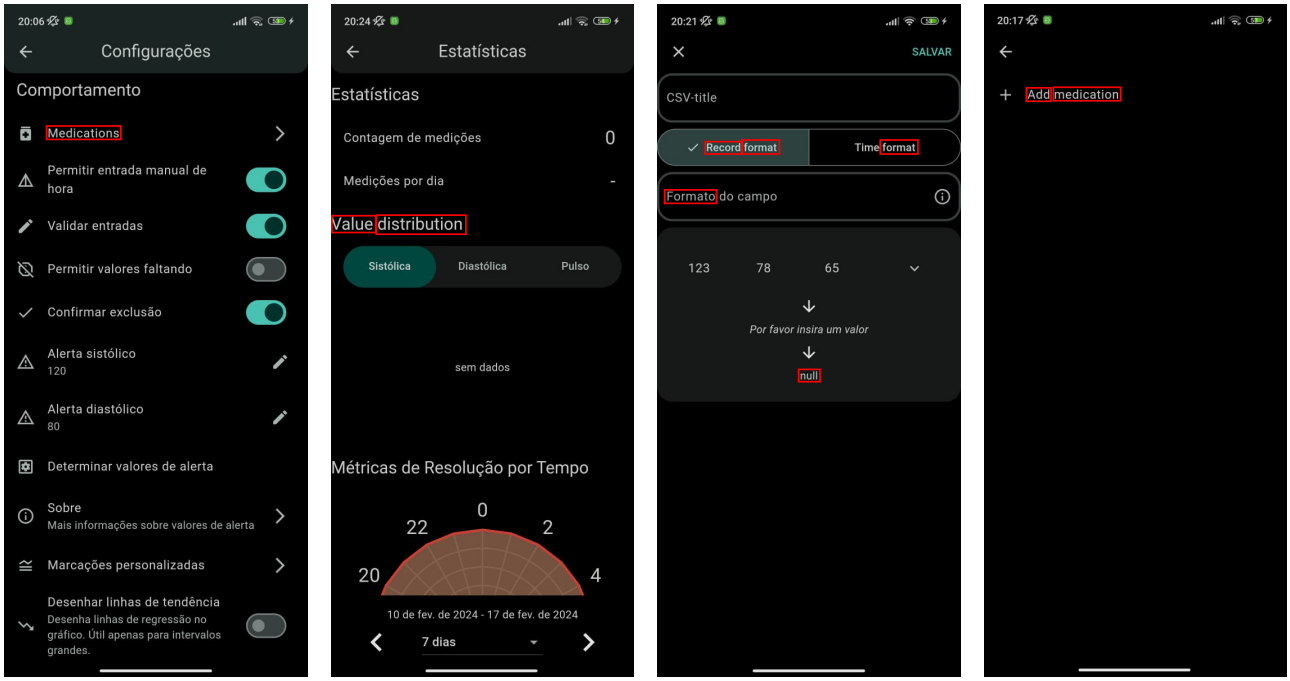



Figura 5.7: Falhas encontradas no aplicativo Blood Pressure Monitor com o arquivo ignore

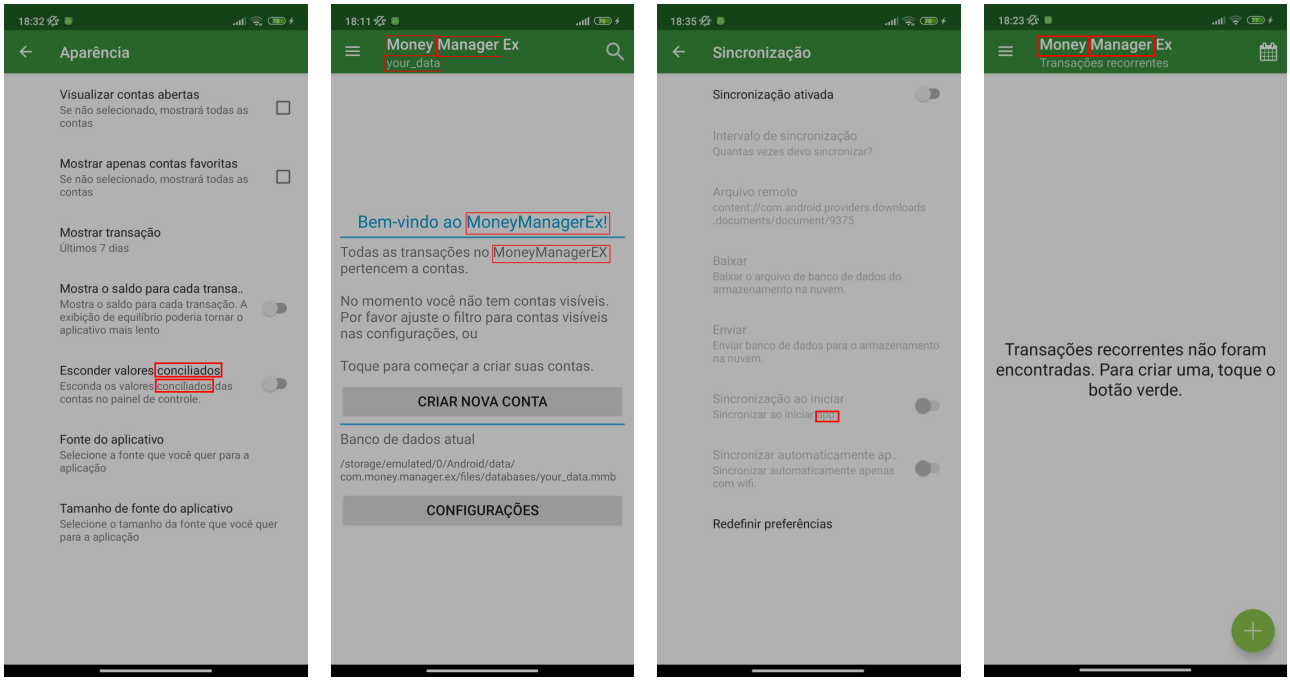



Figura 5.8: Falhas encontradas no aplicativo MoneyManagerEX sem o arquivo *Ignore*

## 5.5 Resultados MoneyManagerEX

As Figuras 5.8 e 5.9 mostram falhas identificadas no aplicativo MoneyManagerEX pela ferramenta, antes e após a aplicação do arquivo Ignore, respectivamente. As palavras identificadas pela ferramenta como não traduzidas foram: automobile, bills, education, food, gifts, healthcare, homeneeds, insurance, leisure, miscellaneous, expenses, read, update, and, area, place, showing, them, toast, or, new, screen, populate, number, send, us, sending, about, initial, saturday, february, database, currency

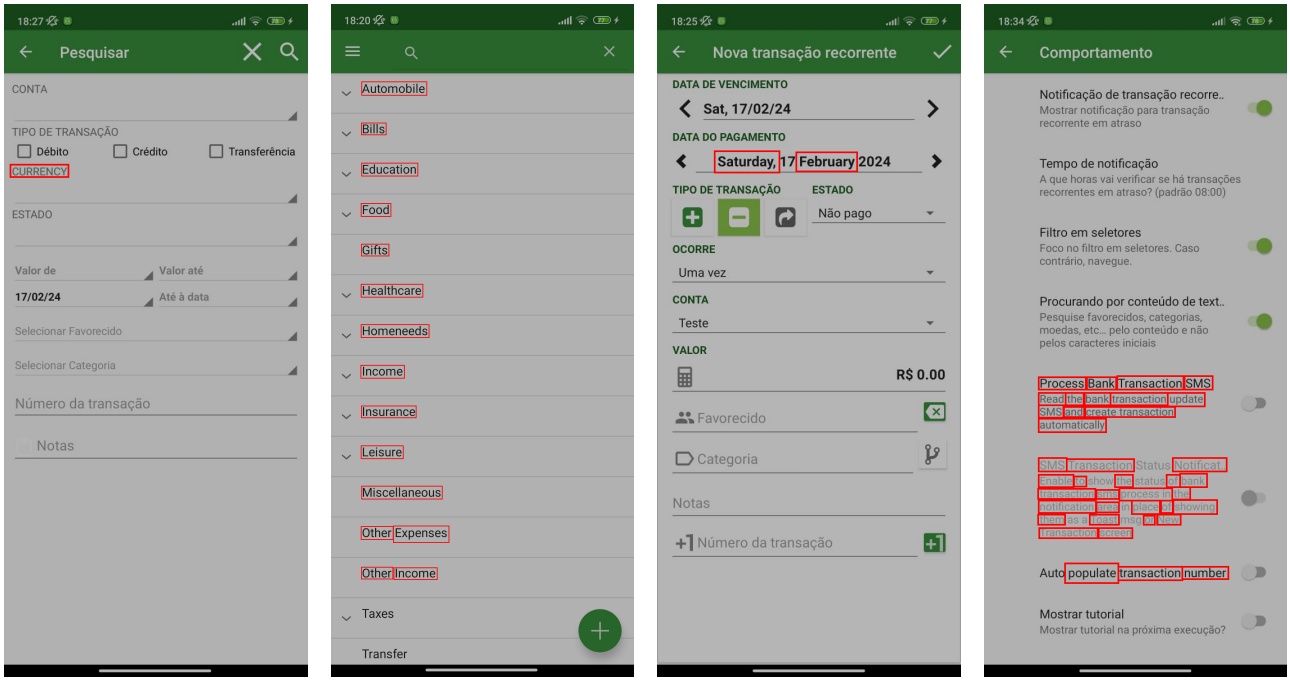



Figura 5.9: Falhas encontradas no aplicativo MoneyManagerEX com o arquivo *Ignore*

CAPÍTULO 6

# Discussão

## 6.1 Discussão Geral

A Tabela 5.1 ilustra como o tempo de exploração dos aplicativos é significativamente influenciado pelo número de telas. Isso indica que, quanto maior o número de telas, maior o tempo necessário para capturar as telas e os arquivos XML correspondentes. A integração da ferramenta com exploradores automáticos de aplicativos, como o Droidbot [14], demonstrou ser crucial para reduzir o tempo de exploração e aprimorar a experiência do usuário. Observou-se que o tempo de execução da ferramenta permanece estável, embora varie conforme o número de telas e a quantidade de palavras em cada captura de tela, influenciando diretamente o tempo de processamento pelo OCR.

A Tabela 5.2 destaca a drástica redução na quantidade de falhas após a utilização da ferramenta com o arquivo *ignore*. Isso evidencia a importância desta funcionalidade para a ferramenta, especialmente considerando que a maioria dos aplicativos contém uma quantidade significativa de palavras que não necessitam de tradução. Contudo, a personalização do arquivo *ignore* depende especificamente de cada aplicativo, requerendo uma configuração única para cada caso. Após a implementação com o arquivo *ignore*, em média, apenas 32.33% das palavras não traduzidas restaram, indicando que a ferramenta sem o arquivo ignore detecta muitas palavras que não precisam de tradução.

## 6.2 Kahoot

Os resultados da ferramenta no aplicativo Kahoot revelaram algumas inconsistências interessantes. A Figura 5.1 destaca palavras em português que foram erroneamente identificadas como falhas de tradução, como "potencialize", "aleatoriamente", "desativada", e "organizador". Essas ocorrências não estavam presentes no dicionário fornecido à ferramenta, o que sugere que a performance da ferramenta está diretamente ligada à qualidade e abrangência do dicionário utilizado. Além disso, foi observado que abreviações não constam no dicionário, resultando em falsos positivos, o que implica na necessidade do usuário adicionar tais abreviações ao arquivo *ignore*.

A Figura 5.2 ilustra os verdadeiros problemas de tradução identificados, evidenciando a utilidade da ferramenta na identificação de falhas que podem impactar negativamente a experiência do usuário. Isso inclui elementos de interface como dropdowns não traduzidos e outros componentes importantes para a compreensão do aplicativo.

## 6.3 Notion

Os resultados obtidos no aplicativo Notion revelaram detalhes interessantes quanto à identificação de falhas de tradução pela ferramenta. A Figura 5.3 destaca que algumas palavras,

embora em inglês, como "Wiki" e "Link", foram marcadas como falhas. No entanto, esses termos já são amplamente utilizados e compreendidos no contexto do português, não necessitando de tradução.

Além disso, a ferramenta marcou identificadores únicos, como "0bc11203a853" e nomes próprios como "Matheus" e "Lucena", como falhas de tradução. Esses casos evidenciam a importância de ajustar o arquivo *ignore*, demonstrando a necessidade de uma personalização cuidadosa do arquivo para cada aplicativo analisado.

A Figura 5.4 ilustra falhas reais de tradução identificadas, como botões "Notify Me" e "Connect To" não traduzidos e telas que permanecem sem localização para o português. Isso sublinha a utilidade da ferramenta em revelar áreas que necessitam de atenção na localização de aplicativos, ao mesmo tempo em que destaca a importância de uma análise crítica dos resultados para identificar verdadeiras falhas de tradução versus uso aceitável de termos em inglês ou identificadores únicos.

## 6.4 Blood Pressure Monitor

O aplicativo Blood Pressure Monitor apresentou desafios únicos. Conforme ilustrado na Figura 5.6, a ferramenta não conseguiu reconhecer formatos de hora (como HH:mm) e termos médicos específicos (por exemplo, "diastólica"), marcando-os como falhas. Isso ressalta a importância de um dicionário bem ajustado e a necessidade de personalização do arquivo *ignore* para incluir termos e formatos específicos utilizados no aplicativo. A identificação de abreviações como "fev" (para fevereiro) também indica a sensibilidade da ferramenta a variações linguísticas.

A Figura 5.7 mostra os resultados promissores da ferramenta ao identificar falhas genuínas, demonstrando sua capacidade de destacar problemas que podem não ser imediatamente óbvios para os testadores, mas que são cruciais para a usabilidade e acessibilidade do aplicativo. No entanto, um problema específico surgiu com a palavra "formato", que foi indevidamente sinalizada como uma falha na segunda imagem da figura, por conta da sua proximidade com a palavra em inglês "format". Este erro ocorreu por conta da imprecisão do OCR.

## 6.5 MoneyManagerEX

Finalmente, a análise realizada no aplicativo MoneyManagerEX, como mostrado na Figura 5.9, evidencia a eficácia da ferramenta em identificar telas com falhas significativas de tradução. A capacidade de sinalizar uma tela inteira que não estava localizada para o português destaca o potencial da ferramenta em auxiliar no processo de localização de aplicativos.

## 6.6 Ameaças à validade

A validade dos resultados deste estudo é limitada principalmente pelo escopo restrito dos testes e pela natureza da execução manual, que pode introduzir viés. Além dessas limitações, o estudo enfrenta desafios adicionais relacionados à diversidade linguística e cultural. A ferramenta foi avaliada apenas com aplicativos em português, o que restringe nossa compreensão de sua eficácia em outros contextos linguísticos e culturais. Idiomas com características únicas, como diferentes direções de escrita e particularidades culturais, podem apresentar desafios não capturados neste estudo. Além disso, a variedade de gêneros de aplicativos testados foi limitada, potencialmente não abrangendo todos os desafios de localização específicos de cada categoria

de aplicativo.

As limitações inerentes ao reconhecimento óptico de caracteres (OCR), representam ameaças adicionais à validade dos resultados. Mudanças na interface do usuário ou na apresentação de dados podem impactar a eficácia da ferramenta. A precisão do OCR, essencial para a detecção de falhas de localização, pode ser afetada por vários fatores, incluindo a qualidade das imagens e o layout da tela. A subjetividade na avaliação de falhas de localização e variações na qualidade e cobertura do dicionário utilizado também podem influenciar os resultados, levando a interpretações divergentes sobre o que constitui uma falha de localização.

Essas considerações sublinham a importância de testes mais abrangentes e de estratégias de mitigação para futuras pesquisas. Expandir os testes para incluir uma gama mais ampla de idiomas e gêneros de aplicativos, atualizar regularmente os dicionários e explorar técnicas avançadas para melhorar a precisão do OCR são etapas essenciais para aumentar a validade e a aplicabilidade da ferramenta em diversos contextos de localização de aplicativos.

CAPÍTULO 7

# Conclusão e Trabalhos Futuros

O presente trabalho apresentou o desenvolvimento de uma ferramenta automatizada destinada a facilitar a identificação de falhas de tradução em aplicativos Android, empregando técnicas de reconhecimento óptico de caracteres (OCR) e análise de arquivos XML. Embora o projeto tenha enfrentado desafios significativos, refletindo as complexidades inerentes à tradução e localização de aplicativos em um contexto global, ele também teve êxitos pontuais ao identificar falhas verdadeiras de tradução, demonstrando o potencial da ferramenta para auxiliar no processo de testes de localização de aplicativos.

Para trabalhos futuros, duas melhorias são sugeridas: a primeira é integrar a ferramenta com sistemas automáticos de exploração de aplicativos, como o Droidbot. Isso pode automatizar e agilizar o processo de testes, evitando a necessidade de navegação manual. A segunda é expandir as funcionalidades da ferramenta para detectar uma variedade maior de erros de tradução, além da simples ausência de tradução. Essas melhorias poderiam tornar a ferramenta mais versátil e valiosa no processo de localização de aplicativos.

# Referências Bibliográficas


[1] S. Gross *et al.*, "Internationalization and localization of software," *Eastern Michigan University, Department of Computer Science, Ypsilanti, Michigan, Thesis June*, vol. 19, 2006 (cit. on pp. 1, 2).

[2] M. Couto and B. Miranda, "An industrial experience report on the challenges in training localization and internationalization testers," in *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*, 2023, pp. 96–98 (cit. on p. 1).

[3] A. M. A. Awwad and W. Slany, "Automated bidirectional languages localization testing for android apps with rich gui," *Mob. Inf. Syst.*, vol. 2016, 2872067:1–2872067:13, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:29777522 (cit. on p. 1).

[4] L. Eikvil, "Optical character recognition," *citeseer. ist. psu. edu/142042. html*, vol. 26, 1993 (cit. on pp. 1, 4).

[5] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 7, pp. 1480–1500, 2015. DOI: 10.1109/TPAMI.2014.2366765 (cit. on p. 5).

[6] A. Alameer, S. Mahajan, and W. G. J. Halfond, "Detecting and localizing internationalization presentation failures in web applications," in *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2016, pp. 202–212. DOI: 10.1109/ICST.2016.36 (cit. on p. 5).

[7] C. Escobar-Velásquez, A. Donoso-Diaz, and M. Linares-Vásquez, "Itdroid: A tool for automated detection of i18n issues on android apps," in *2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft)*, 2021, pp. 52–55. DOI: 10.1109/MobileSoft52590.2021.00012 (cit. on p. 5).

[8] M. R. L. de Couto and B. Miranda, "Towards improving automation support for internationalization and localization testing," in *Anais Estendidos do XXI Simpósio Brasileiro de Qualidade de Software*, SBC, 2022, pp. 9–14 (cit. on p. 5).

[9] M. Couto and B. Miranda, "L10n-trainer: A tool to assist in the training of localization (l10n) and internationalization (i18n) testers," in *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, 2023, pp. 277–282 (cit. on p. 5).

[10] L. P. Felipe and B. Miranda, "Supporting localization testing through automated application navigation," in *Anais Estendidos do XXII Simpósio Brasileiro de Qualidade de Software*, SBC, 2023, pp. 25–30 (cit. on p. 5).

[11] *Lista de todas as palavras do português brasileiro*, https://www.ime.usp.br/~pf/dicios/, Acessado em 13/02/2024 (cit. on p. 7).

[12] *List of all spanish words. source rae*, https://github.com/JorgeDuenasLerin/diccionario-espanol-txt, Acessado em 13/02/2024 (cit. on p. 7).

[13] *Easyocr: A simple and powerful optical character recognition (ocr) tool*, https://github.com/JaidedAI/EasyOCR, Acessado em 13/02/2024 (cit. on p. 7).

[14] Y. Li, Z. Yang, Y. Guo, and X. Chen, "Droidbot: A lightweight ui-guided test input generator for android," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 23–26. DOI: 10.1109/ICSE-C.2017.8 (cit. on p. 19).