

PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, s/n - Ondina
Salvador, BA, Brasil, 40170-110

<http://pgcomp.dcc.ufba.br>
pgcomp@ufba.br

O mercado de dispositivos móveis tem crescido exponencialmente nos últimos anos, assim como a necessidade de aplicativos que primem pela qualidade e que ofereçam funcionalidades que ampliem a retenção de usuários. Nos últimos anos percebe-se uma crescente quantidade de estudos que apresentam soluções para problemas inerentes à demanda supracitada. A atividade de testes de software tem um papel de grande importância no processo de garantia da qualidade de software. Em particular, os testes de regressão apresentam-se como uma estratégia viável para lidar com a complexidade e com a constante evolução dos aplicativos, visto que o seu principal objetivo é garantir que as mudanças realizadas entre versões não alterem o comportamento do sistema. Embora a literatura tenha dedicado esforços para o desenvolvimento de novas técnicas de teste de regressão para Android - o sistema operacional mais popular para os dispositivos móveis - os estudos existentes são limitados no que diz respeito a demonstrar quais são as técnicas de teste de regressão utilizadas por profissionais. A presente investigação tem como objetivo realizar uma síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android. A pesquisa foi organizada em quatro etapas: (i) realização de uma revisão estruturada da literatura sobre técnicas de teste de regressão de aplicativos para Android, (ii) aplicação de um survey, (iii) realização de entrevistas com profissionais da indústria, e (iv) a construção de uma síntese temática. O survey obteve 100 respostas e proporcionou uma visão preliminar sobre como é realizado o processo de teste durante e após a manutenção dos aplicativos, quão automatizado é o teste de regressão de aplicativos para Android na prática e os motivos pelos quais os profissionais não realizam teste de regressão após atualizarem os aplicativos. As entrevistas realizadas com 16 profissionais da indústria contribuíram com a pesquisa no sentido de identificar o nível de conhecimento dos profissionais e sua relação com a automação dos testes, as linguagens mais utilizadas no desenvolvimento dos aplicativos para Android e qual a relação delas com o processo de automação, as ferramentas utilizadas para automação de teste e os requisitos esperados pelos profissionais para executarem teste de regressão automatizado e um entendimento preliminar sobre como os profissionais executam teste de regressão na prática. A síntese temática apresentou um modelo resultante da comparação dos resultados obtidos nas diferentes fontes de dados - literatura, survey e entrevistas, sobre o uso de técnicas de teste de regressão. Como trabalhos futuros pretende-se: replicar as entrevistas, propor um modelo formal de ferramenta de teste de regressão, validar o modelo temático proposto e investigar como a academia pode contribuir na formação de acordo com a necessidade do mercado de trabalho.

Palavras-chave: Teste de Regressão; Testes para Android; Engenharia de Software Empírica; Pesquisa multi-método.

Síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android

Sara Mendes Oliveira Lima

Dissertação de Mestrado

Universidade Federal da Bahia

Programa de Pós-Graduação em
Ciência da Computação

Outubro | 2021

MSC | 122 | 2021

Síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android

Sara Mendes Oliveira Lima

UFBA





Universidade Federal da Bahia
Instituto de Computação

Programa de Pós-Graduação em Ciência da Computação

**SÍNTESE TEMÁTICA SOBRE A ADOÇÃO
DE TÉCNICAS DE TESTE DE REGRESSÃO
EM PROJETOS DE SOFTWARE PARA A
PLATAFORMA ANDROID**

Sara Mendes Oliveira Lima

DISSERTAÇÃO DE MESTRADO

Salvador
Outubro/2021

SARA MENDES OLIVEIRA LIMA

**SÍNTESE TEMÁTICA SOBRE A ADOÇÃO DE TÉCNICAS DE
TESTE DE REGRESSÃO EM PROJETOS DE SOFTWARE PARA
A PLATAFORMA ANDROID**

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Ivan do Carmo Machado

Salvador
Outubro/2021

Ficha catalográfica elaborada pela Biblioteca Universitária de
Ciências e Tecnologias Prof. Omar Catunda, SIBI - UFBA.

L732 Lima, Sara Mendes Oliveira.

Síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android/
Sara Mendes Oliveira Lima. – Salvador, 2021.

191f.

Orientador: Prof. Dr. Ivan do Carmo Machado

Dissertação (Mestrado) – Universidade Federal da Bahia.
Instituto de Computação, 2021.

1. Engenharia de Software. 2. Teste de Software. 3. Android.
I. Machado, Ivan do Carmo. II. Universidade Federal da Bahia.
III. Título.


CDU 004.415.5

“Síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android”

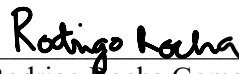
Sara Mendes Oliveira Lima

Dissertação apresentada ao Colegiado do Programa de Pós-Graduação em Ciência da Computação na Universidade Federal da Bahia, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação.

Banca Examinadora



Prof. Dr. Ivan do Carmo Machado (Orientador-UFBA)



Prof. Dr. Rodrigo Rocha Gomes e Souza (UFBA)



Prof. Dr. Alcemir Rodrigues Santos (UESPI)

Dedico este trabalho a minha mãe Rose Marie Mendes de Lima, minha primeira e eterna professora.

AGRADECIMENTOS

A Deus pelo dom da vida e seu amor incondicional em todos os instantes, em especial nos momentos mais difíceis, e por me conceder os dons necessários para concretizar este sonho. A Nossa Senhora pela sua poderosa intercessão e colo materno. E a São Miguel Arcanjo pela sua proteção.

A minha mãe, amiga, companheira e grande incentivadora. Sem você nada seria possível. Obrigada por lutar pela minha vida desde o seu ventre até os dias atuais e por não desistir de mim.

A minha irmã Rani pelas orações e pelas aulas de inglês na infância e adolescência.

A Norival Filho e seus familiares pelo apoio, orações e desejo de sucesso nesta jornada acadêmica.

Ao meu filho Logan Lima que transformou minha vida desde o dia em que tive a oportunidade de conhecê-lo. Com seus miados ou até mesmo com seu silêncio renova minhas energias e me enche de alegria.

Aos meus familiares, amigos(as), padrinhos, madrinhas e afilhados(as) pelas orações, energias positivas, mensagens de apoio e tantas formas de se fazerem presentes nesta jornada do mestrado e em tantas outras.

As amigas que tive a oportunidade de fazer durante o período que residi em Salvador. Em especial, a amiga Jocilene Moraes por tantos momentos bons compartilhados.

A família que a UFBA me deu, amizades que levo para a vida: Railana Santana, Franklin Silva, Tássio Virgínio, Nildo Jr, Joselito Mota, Denivan Campos, Virginia Venegani, Luana Martins, Tiago Motta, Renata Souza, Gláucia Boechat, Mayka Lima, Brunna Caroline, Samia Capistrano, Verônica Luzia, Liviany Reis, José Augusto e demais amigos(as) do LES, INES e do grupo de pesquisa ARIES Lab. Gratidão por cada momento compartilhado.

Ao meu psicólogo Ubirajara Martins por me auxiliar no processo de auto-conhecimento, resiliência, foco e persistência.

Aos professores e toda equipe de servidores/colaboradores da Universidade Federal da Bahia por contribuírem com esta etapa do meu processo formativo.

Ao professor e orientador Ivan Machado pelos conhecimentos compartilhados que contribuíram na minha formação profissional e pessoal.

A professora Larissa Soares pelos conhecimentos compartilhados e por me inspirar com seu exemplo de mulher, profissional e pesquisadora.

Aos pesquisadores e profissionais que colaboraram participando do *Survey* e das entrevistas.

A todos(as) muito obrigada. A trajetória com vocês ficou mais leve, mais feliz e repleta de aprendizado.

O teste de regressão constitui a grande maioria dos esforços de teste no desenvolvimento de software comercial, e é uma parte essencial de qualquer processo de desenvolvimento de software viável.

—AMMANN AND OFFUTT (2008)

RESUMO

O mercado de dispositivos móveis tem crescido exponencialmente nos últimos anos, assim como a necessidade de aplicativos que primem pela qualidade e que ofereçam funcionalidades que ampliem a retenção de usuários. Nos últimos anos percebe-se uma crescente quantidade de estudos que apresentam soluções para problemas inerentes à demanda supracitada. A atividade de testes de software tem um papel de grande importância no processo de garantia da qualidade de software. Em particular, os testes de regressão apresentam-se como uma estratégia viável para lidar com a complexidade e com a constante evolução dos aplicativos, visto que o seu principal objetivo é garantir que as mudanças realizadas entre versões não alteram o comportamento do sistema. Embora a literatura tenha dedicado esforços para o desenvolvimento de novas técnicas de teste de regressão para Android - o sistema operacional mais popular para os dispositivos móveis - os estudos existentes são limitados no que diz respeito a demonstrar quais são as técnicas de teste de regressão utilizadas por profissionais. A presente investigação tem como objetivo realizar uma síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android. A pesquisa foi organizada em quatro etapas: (i) realização de uma revisão estruturada da literatura sobre técnicas de teste de regressão de aplicativos para Android, (ii) aplicação de um *survey*, (iii) realização de entrevistas com profissionais da indústria, e (iv) a construção de uma síntese temática. O *survey* obteve 100 respostas e proporcionou uma visão preliminar sobre como é realizado o processo de teste durante e após a manutenção dos aplicativos, quão automatizado é o teste de regressão de aplicativos para Android na prática e os motivos pelos quais os profissionais não realizam teste de regressão após atualizarem os aplicativos. As entrevistas realizadas com 16 profissionais da indústria contribuíram com a pesquisa no sentido de identificar o nível de conhecimento dos profissionais e sua relação com a automação dos testes, as linguagens mais utilizadas no desenvolvimento dos aplicativos para Android e qual a relação delas com o processo de automação, as ferramentas utilizadas para automação de teste e os requisitos esperados pelos profissionais para executarem teste de regressão automatizado e um entendimento preliminar sobre como os profissionais executam teste de regressão na prática. A síntese temática apresentou um modelo resultante da comparação dos resultados obtidos nas diferentes fontes de dados - literatura, *survey* e entrevistas, sobre o uso de técnicas de teste de regressão. Como trabalhos futuros pretende-se: replicar as entrevistas, propor um modelo formal de ferramenta de teste de regressão, validar o modelo temático proposto e investigar como a academia pode contribuir na formação de acordo com a necessidade do mercado de trabalho.

Palavras-chave: Teste de Regressão; Testes para Android; Engenharia de Software Empírica; Pesquisa multi-método.

ABSTRACT

The mobile device market has grown exponentially in recent years, as the need for apps that strive for quality and that offer *features* that increase user retention. In recent years, a growing number of studies have provided solutions to problems inherent to the demand mentioned above. The software testing activity plays an essential role in the software quality assurance process. In particular, regression tests are presented as a viable strategy to deal with the complexity and constant evolution of applications since its main objective is to ensure that changes made between versions of a product do not change the system's behavior. Although the literature has devoted efforts to developing new regression testing techniques for Android - the most popular operating system for mobile devices - existing studies do not address how professionals use testing techniques. This investigation aims to carry out a thematic synthesis on the adoption of regression testing techniques in software projects for the Android platform. We organized the research in four steps: (i) carrying out a structured review of the technical literature on regression testing of Android applications, (ii) applying a *survey*, (iii) carrying out an identifier with professionals from the industry, and (iv) the construction of a thematic synthesis. The *survey* got 100 responses and provided preliminary insight into how the testing process is performed during and after application maintenance, how automated regression testing of Android applications is in practice, and why the professionals do not perform regression testing after updating applications. In the interview study, we inquired 16 industry professionals. This study contributed to the research as it could unveil the level of knowledge of professionals and its relationship with test automation. In addition, we could identify the most used languages in the development of Android applications and the tools used for testing automation. Furthermore, the study provided us with a preliminary understanding of how professionals perform regression testing in practice. Finally, the thematic synthesis presents a model resulting from the comparison of evidence gathered from different data sources - literature, *survey* and declares about the use of regression test techniques. As future work, we intend to: replicate the interviews, propose a formal model of a regression test tool, validate the proposed thematic model and investigate how academia can contribute to training according to the needs of the labor market.

Keywords: Regression Testing; Android Testing; Empirical Software Engineering; Multi-method Research.

SUMÁRIO

Lista de Figuras	xix
Lista de Tabelas	xxi
Lista de Acrônimos	xxiii
Capítulo 1—Introdução	1
1.1 Motivação	1
1.2 Problema	2
1.3 Objetivo e Questões de Pesquisa	3
1.4 Metodologia	4
1.5 Estrutura do Trabalho	6
Capítulo 2—Referencial Teórico	7
2.1 Evolução e Qualidade de Software	7
2.2 Teste de Software	8
2.2.1 Validação, Verificação e Teste	8
2.2.2 Atividade de Teste de Software	9
2.2.3 Técnicas de Teste de Software	9
2.2.4 Tipos de Teste	10
2.2.5 Testes de aplicativos para Android	11
2.3 Teste de Regressão	12
2.3.1 Definição	12
2.3.2 Técnicas de Teste de Regressão	13
2.3.2.1 Técnicas de Seleção	13
2.3.2.2 Técnicas de Minimização	14
2.3.2.3 Técnicas de Priorização	14
2.3.3 Classificação dos casos de teste	15
2.4 Síntese do Capítulo	15
Capítulo 3—Trabalhos Relacionados	17
3.1 Técnicas de teste de regressão para Android	17
3.2 Testes para aplicações Android	20
3.3 Aplicação industrial de testes de regressão	21

3.4	Síntese do Capítulo	21
Capítulo 4—Survey		23
4.1	Apresentação e Questões de Pesquisa	23
4.2	Metodologia de Pesquisa	24
4.2.1	Identificação do Público Alvo	24
4.2.2	Design do Questionário	24
4.2.3	Estudo Piloto	26
4.2.4	Distribuição do Questionário	26
4.3	Resultados	27
4.3.1	Perfil dos participantes	27
4.3.2	Questões de Pesquisa	33
4.3.2.1	Quão automatizado é o teste de regressão de aplicativos Android na prática? (QP1)	33
4.3.2.2	Como é realizado o processo de teste durante e após a manutenção dos aplicativo? (QP2)	37
4.3.2.3	Quais são os motivos para não realizar testes de regressão após a manutenção de aplicativos para Android? (QP3)	38
4.4	Discussão	39
4.4.1	Foco da atividade de teste	40
4.4.2	Automação do processo de teste	40
4.4.3	Processo de teste de regressão	41
4.5	Ameaças à Validade	41
4.6	Síntese do capítulo	42
Capítulo 5—Entrevistas		43
5.1	Apresentação e Questões de Pesquisa	43
5.2	Metodologia da Pesquisa	44
5.2.1	Identificação do Público Alvo	44
5.2.2	Definição do Protocolo	45
5.2.3	Estudo Piloto	45
5.2.4	Seleção dos Participantes	45
5.2.5	Condução das Entrevistas	46
5.2.6	Transcrição das Entrevistas	47
5.2.7	Codificação	47
5.3	Resultados	48
5.3.1	Perfil dos Participantes	48
5.3.2	Questões de Pesquisa	49
5.3.2.1	Qual a relação entre o nível de conhecimento do profissional e a forma como ele realiza teste de regressão em aplicativos para Android? (QP1)	49

5.3.2.2	Quais são as linguagens de programação utilizadas no desenvolvimento de aplicativos para Android, e qual relação entre linguagens e automação dos testes? (QP2)	51
5.3.2.3	Quais são as ferramentas utilizadas para automação dos testes e quais são as características do ferramental de apoio que os profissionais necessitam para automatizar os testes? (QP3)	52
5.3.2.4	Como os profissionais executam teste de regressão nos projetos em que atuam? (QP4)	55
5.4	Discussão	58
5.4.1	Nível de Conhecimento e Automação de Teste	58
5.4.2	Linguagens e Automação	58
5.4.3	Ferramentas para Automação	59
5.4.4	Teste de Regressão	59
5.5	Ameaças à Validade	60
5.6	Síntese do Capítulo	61
Capítulo 6—Síntese Temática		63
6.1	Apresentação	63
6.2	Metodologia da Pesquisa	63
6.2.1	Extração de Dados	63
6.2.2	Codificação	66
6.2.3	Tradução de códigos em temas	66
6.2.3.1	Teste de regressão	66
6.2.3.2	Técnicas	67
6.2.3.3	Aplicativos	68
6.2.3.4	Profissional	69
6.2.4	Criação de um modelo temático	70
6.3	Discussão	70
6.3.1	Teste de Regressão	71
6.3.2	Técnicas	71
6.3.3	Aplicativos	72
6.3.4	Profissional	73
6.3.5	O uso das técnicas de teste de regressão por profissionais que atuam em projetos de aplicativos para Android	73
6.4	Síntese do Capítulo	74
Capítulo 7—Considerações Finais		75
7.1	Resultados	75
7.2	Contribuições	75
7.3	Trabalhos Futuros	76

Apêndice A—Questionário survey	83
Apêndice B—Survey: Perfil dos Participantes	93
Apêndice C—Protocolo da Entrevista	97
Apêndice D—Roteiro da Entrevista	99
Apêndice E—Pesquisa sobre Teste em Cenário de Evolução de Aplicativos para Android/Formulário Online	101
Apêndice F—Entrevista: Perfil dos Participantes	107
Apêndice G—Transcrição das Entrevistas	109

LISTA DE FIGURAS

1.1	Distribuição mundial do uso de sistemas operacionais para aplicativos. (GS.STATCOUNTER, 2021)	3
2.1	Procedimento de Teste (HIRAMA, 2011)	9
4.1	Distribuição geográfica dos participantes do survey.	27
4.2	Distribuição da idade dos participantes do survey.	28
4.3	Distribuição do nível de escolaridade dos participantes do survey.	28
4.4	Distribuição da área de formação acadêmica dos participantes do survey .	29
4.5	Distribuição de participantes por tipo de certificação na área de testes de software.	29
4.6	Distribuição de participantes por tipo de curso na área de testes de software.	30
4.7	Distribuição de participantes por área de atuação profissional.	30
4.8	Distribuição de participantes por tempo de experiência profissional na área de testes de software.	31
4.9	Distribuição de participantes por tempo de experiência profissional na área de testes para Android.	31
4.10	Auto-avaliação dos participantes a respeito do nível de conhecimento em testes para Android comparados a outros profissionais da área.	32
4.11	Distribuição de participantes por atividade realizada no processo de teste de software.	32
4.12	Distribuição de participantes por técnica de teste utilizada.	33
4.13	Distribuição de tipos de teste realizados pelos participantes do survey. . .	34
4.14	<i>Survey</i> : Os testes são realizados de forma manual e/ou automatizada . .	34
4.15	Distribuição de participantes por ferramentas.	35
4.16	Distribuição de participantes por estratégia de teste de regressão.	35
4.17	Distribuição de ferramentas utilizadas para automatizar o teste de regressão	36
4.18	Distribuição de dados sobre como os participantes consideram as ferramentas disponíveis	37
4.19	Distribuição da frequência de realização de testes durante a fase de manutenção.	38
4.20	Distribuição de dados sobre os procedimentos adotados pelos participantes para testar a versão atualizada dos aplicativos Android.	38
4.21	Distribuição de fatores que levam à não-realização de testes em aplicativos nas fases de manutenção.	39
5.1	Entrevistas: Etapas Metodologia da Pesquisa	44

5.2	Entrevistas: Fontes de Aprendizado sobre teste de aplicativos para Android	50
5.3	Entrevistas: Fontes de Aprendizado sobre teste de regressão	51
5.4	Entrevistas: Ferramentas para Automação dos Testes	53
6.1	Síntese Temática: Tema Teste de Regressão	67
6.2	Síntese Temática: Tema Técnicas	68
6.3	Síntese Temática: Tema Aplicativos	69
6.4	Síntese Temática: Tema Profissional	70
6.5	Síntese Temática: Modelo Temático	70

LISTA DE TABELAS

2.1	Teste de Regressão Corretivo & Progressivo (LEUNG; WHITE, 1989) . . .	13
4.1	Design do <i>survey</i>	24
4.2	Ferramentas de teste de aplicativos para Android	25
5.1	Entrevista: Roteiro X Questões de Pesquisa	45
5.2	Entrevistas: Estratégias utilizadas para seleção de participantes	46
5.3	Entrevistas: Critérios utilizados para codificação das respostas	48
5.4	Entrevista: Requisitos funcionais elencados pelos participantes	54
5.5	Entrevistas: Requisitos Não-Funcionais	54
6.1	Síntese Temática: Critérios PICOC	64
6.2	Síntese Temática: Bases de Dados	64
6.3	Síntese Temática: String de Busca	64
6.4	Síntese Temática: Artigos encontrados na literatura	65
6.5	Síntese Temática: Códigos x Fonte de Dados	66
B.1	<i>Survey</i> : Perfil dos Participantes	93
B.2	<i>Survey</i> : Perfil dos Participantes	94
B.3	<i>Survey</i> : Perfil dos Participantes	95
F.1	Entrevistas: Perfil dos Participantes	107

LISTA DE ACRÔNIMOS

APPS	Aplicativos para dispositivos móveis	83
GUI	<i>Graphical User Interface</i>	59
JVM	<i>Java Virtual Machine</i>	12
PO	<i>Product Owner</i>	56
QA	<i>Quality Assurance</i>	48
QP	Questões de Pesquisa	43
RTS	<i>Regression Test Selection</i>	14
SUT	<i>System Under Test</i>	14
VV	Verificação e Validação	8
VVT	Verificação, Validação e Teste	8

1

INTRODUÇÃO

1.1 MOTIVAÇÃO

Desenvolver software não é uma tarefa simples e envolve diversos fatores, tais como: escopo, prazo e recursos disponíveis. Um problema em potencial é a entrega de produtos que diferem das expectativas dos usuários finais (ou clientes). Segundo Delamaro et al. (2007) para que esses problemas não perdurem e para que se possa identificá-los antes da implantação do software, existe uma série de atividades, coletivamente chamadas de Verificação, Validação e Teste (VVT).

Neste sentido, as atividades de VVT auxiliam na garantia de qualidade, do termo original e mais comumente utilizado *Quality Assurance* (QA) (em inglês). Segundo HIRAMA (2011) QA refere-se a definição de como a qualidade de software pode ser alcançada e como as empresas que trabalham com desenvolvimento sabem que o software alcançou o nível de qualidade necessário, estabelecendo processos, procedimentos e padrões que conduzem a um software de qualidade.

O presente trabalho tem como foco teste de software. Dentre as etapas do procedimento de teste proposto por HIRAMA (2011) está a geração dos casos de teste. De acordo com (WONG et al., 1995) um caso de teste é uma sequência de valores inseridos em um programa durante uma execução, e um conjunto de teste consiste em um ou mais casos de teste que serão executados para testar o programa. Visto a inviabilidade de testar todas as possíveis entradas de um programa, são estabelecidos critérios de cobertura para decidir quais entradas de teste usar, proporcionando que os profissionais detectem as falhas e forneçam assim um software com qualidade e confiabilidade (AMMANN; OFFUTT, 2008). Segundo Delamaro et al. (2007) a atividade de teste pode ser caracterizada em três níveis (ou fases): teste de unidade, teste de integração e teste de sistema.

Visto que o teste tem como objetivo encontrar erros no software (HIRAMA, 2011), antes de se disponibilizar uma versão estável do software faz-se necessária que uma grande massa de testes seja executada. Entretanto, existe um tipo de teste, denominado teste de regressão, que deve ser considerado quando é necessário realizar modificações (manutenções) no software após entrega, seja para corrigir falhas (corretiva), melhorar o

desempenho ou implementar novas funcionalidades (perfectiva), adaptar o produto a um novo ambiente (adaptativa) (IEEE, 1998) e prevenir problemas antes que eles ocorram (preventiva) (MENS; DEMEYER, 2008). O teste de regressão tem como objetivo fornecer confiança de que as alterações recém-introduzidas não comprometem o comportamento funcional da parte existente e inalterada do software (YOO; HARMAN, 2012).

Ao realizar o teste de regressão tende-se a reexecutar todos os casos de teste da versão original do software para testar a versão modificada. Porém, esse procedimento tende a gerar um grande esforço da equipe de teste, a depender da quantidade de casos de teste. A literatura então apresenta uma série de técnicas para lidar com este problema, em particular no sentido de propor mecanismos para reduzir o número de reexecuções de casos de teste, sem perder a cobertura do código fonte e a capacidade de detecção de defeitos (GRAVES et al., 2001; ENGSTRÖM et al., 2010; KAZMI et al., 2017; ROMANO et al., 2018).

Segundo YOO e HARMAN (2012) as técnicas de teste de regressão são classificadas como: técnicas de *seleção*, técnicas de *minimização* e técnicas de *priorização*. As técnicas de *seleção* buscam reduzir a quantidade de casos de teste a serem reexecutados, selecionando-os de acordo com a parte do software que foi alterada. As técnicas de *minimização* propõem a remoção de casos de teste redundantes. Por sua vez, as técnicas de *priorização* tem como objetivo encontrar o conjunto mais representativo de casos de teste a serem reexecutados.

Testes de regressão apresentam-se como uma estratégia viável para lidar com a complexidade - e a constante evolução - dos aplicativos, tais como: experiência do usuário, portabilidade, segurança, conectividade e desempenho (DO et al., 2016; CHOI et al., 2018; JIANG et al., 2018).

1.2 PROBLEMA

Os dispositivos móveis têm sido cada vez mais utilizados no cotidiano. Pesquisas de mercado indicam que o **Android** é o sistema operacional mais popularmente em uso em dispositivos móveis, com cerca de 72% dos usuários¹, conforme ilustra a Figura 1.1.

Considerando a constante evolução dos aplicativos, a necessidade de avaliar se as alterações realizadas não afetam seu comportamento e os testes de regressão se apresentarem como uma estratégia viável na detecção de defeitos (JIANG et al., 2018), nos últimos anos, a literatura tem apresentado estudos sobre a aplicação das técnicas de teste de regressão em aplicativos, tais como: (AMALFITANO et al., 2011), (SZABÓ et al., 2012), (BAUERSFELD, 2013), (DO et al., 2016), (GRØNLI; GHINEA, 2016), (GÓMEZ et al., 2016), (HU; NEAMTIU, 2016), (LI et al., 2017), (CHOI et al., 2018), (JIANG et al., 2018), (CHANG et al., 2018). Alguns desses estudos apresentam propostas de ferramentas de apoio, cujo objetivo é automatizar o processo de teste de regressão. Tais ferramentas propostas são avaliadas através de estudos experimentais, que indicam evidências empíricas acerca da eficiência/eficácia da ferramenta desenvolvida.

Embora a literatura apresenta estudos sobre evidências empíricas acerca das técnicas de teste de regressão aplicadas a projetos Android, há pouca evidência disponível na

¹<http://gs.statcounter.com/os-market-share/mobile/worldwide> - Acessado em 04/09/2021.

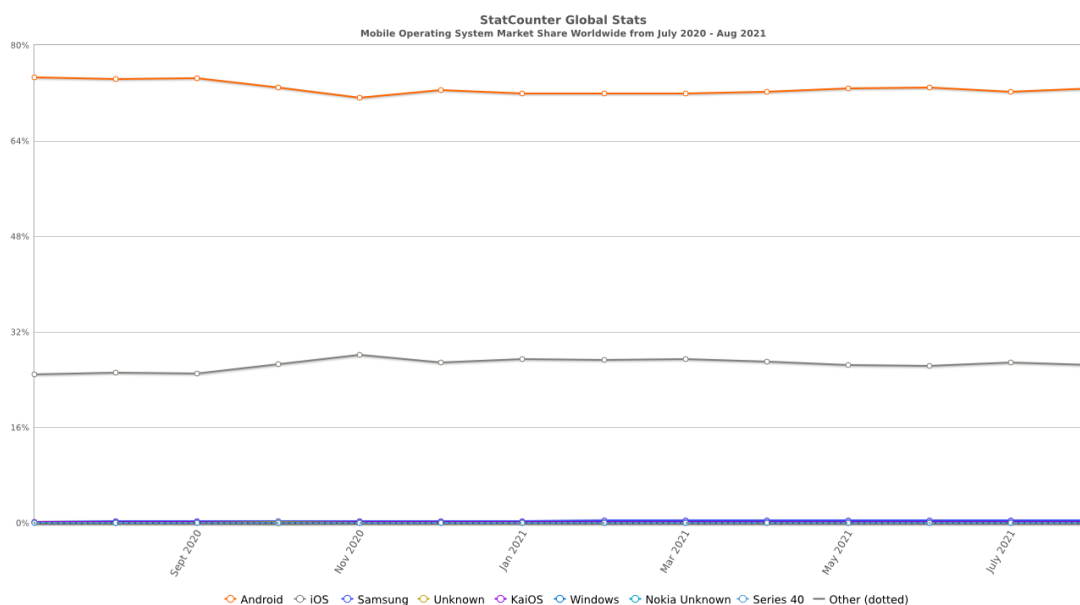


Figura 1.1 Distribuição mundial do uso de sistemas operacionais para aplicativos. (GS.STATCOUNTER, 2021)

literatura no que diz respeito a prover evidências empíricas sobre quais são as técnicas de teste de regressão mais aplicáveis no desenvolvimento de aplicativos Android (JIANG et al., 2018; DO et al., 2016), e também, como estas técnicas são utilizadas por profissionais da indústria de aplicativos (VÁSQUEZ et al., 2017; KOCHHAR et al., 2015; CHOUDHARY et al., 2015). Neste sentido, é fundamental que novas pesquisas sejam desenvolvidas, de modo a ampliar o conjunto de evidências sobre essa aplicabilidade.

Visto que as técnicas de teste de regressão utilizam critérios para redução dos casos de teste a serem reexecutados ao testar uma nova *release* de um software, que os aplicativos para Android passam por constantes manutenções e dado a complexidade do problema ora descrito, identificamos a necessidade de investigar a seguinte questão de pesquisa: **Como o teste de regressão tem sido utilizado em projetos de desenvolvimento de software para a plataforma Android?**

1.3 OBJETIVO E QUESTÕES DE PESQUISA

O presente trabalho tem como objetivo prover uma síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android. Para responder a pergunta principal, propusemos as seguintes Questões de Pesquisa (QP):

QP1 Como os profissionais que atuam em projetos de desenvolvimento de software para a plataforma Android realizam teste de regressão?

A literatura apresenta diversos estudos sobre teste de regressão (ENGSTRÖM et al., 2010; KONG et al., 2019; ALI et al., 2019). Porém, não encontramos estudos que apresentam resultados sobre como os profissionais que atuam em projetos de software para a plataforma Android realizam teste de regressão no

cotidiano. Assim, nossa pesquisa busca coletar evidências sobre o uso prático das técnicas de teste de regressão por profissionais da indústria.

QP2 Quais técnicas disponíveis na literatura são adotadas na prática? Ao longo dos anos a literatura apresenta diversos estudos que tratam sobre testes de regressão, técnicas de teste de regressão e ferramentas de suporte que implementam as técnicas no sentido de automatizar o teste de regressão. As técnicas de teste de regressão estão relacionadas a redução do número de caso de testes a serem reexecutados para testar novas versões de um software sem reduzir a cobertura de teste. Ao conhecer e aplicar essas técnicas, os profissionais que atuam em projetos de desenvolvimento de software para a plataforma Android podem melhorar a eficácia da detecção de defeitos.

QP3 Quais são os requisitos necessários em ferramentas para automação de teste de regressão? Alguns autores, tais como (DO et al., 2016; JIANG et al., 2018; CHOI et al., 2018) apresentam técnicas de teste de regressão implementadas por ferramentas para automação de teste de regressão de aplicativos para Android. Nosso estudo propõe identificar quais são os requisitos funcionais e não-funcionais, necessários a uma ferramenta de automação de teste de regressão na percepção dos profissionais que atuam nos projetos de desenvolvimento de software para a plataforma Android.

1.4 METODOLOGIA

Este estudo tem um caráter qualitativo, por envolver a análise de dados oriundos da literatura, de um *survey* e de entrevistas semi-estruturadas. Neste sentido, para alcançar os objetivos definidos para a pesquisa, empregamos um percurso metodológico que engloba as seguintes fases:

- **Fase 1: Levantamento conceitual.** Esta fase tem como objetivo levantar os conhecimentos essenciais referentes ao tema proposto dessa dissertação, para compreendermos o estado-da-arte e identificarmos os trabalhos relacionados ao nosso estudo. Ela foi dividida em duas etapas:
 1. **Levantamento bibliográfico:** Leitura mais ampla de livros e artigos em busca de um conhecimento inicial sobre os tópicos abordados na pesquisa, são eles: evolução e qualidade de software, teste de software e teste de regressão.
 2. **Revisão estruturada:** Leituras específicas de textos com relação direta a temática de uso de técnicas de teste de regressão em projetos de aplicativos para Android. A revisão foi realizando o método proposto por (ARKSEY; O'MALLEY, 2005), seguindo as seguintes etapas:
 - (a) Identificação das questões de pesquisa;
 - (b) Identificação dos estudos relevantes;
 - (c) Seleção dos estudos;

- (d) Mapeamento dos dados;
- (e) Agrupamento e sumarização dos dados.

Os dados deste estudo estão apresentados na Subseção 6.2.1.

- **Fase 2: *Survey*.** Esta fase contempla a aplicação de um *Survey* com profissionais que trabalham em projetos de aplicativos para Android. O *Survey* tem como objetivo compreender como testadores e desenvolvedores realizam o teste de regressão em projetos de aplicativos para Android. A pesquisa se baseia no método proposto por Kasunic (2005) e aplica os princípios de pesquisa definidos por Kitchenham e Pfleeger (2002). Essa fase foi composta pelas seguintes etapas:
 1. **Planejamento:** Refere-se ao planejamento considerando: identificação do público-alvo, design do questionário, projeto piloto e distribuição do questionário;
 2. **Execução:** Refere-se à aplicação do *Survey*;
 3. **Análise dos Resultados:** Nesta etapa os resultados obtidos são apresentados considerando o perfil dos participantes e as questões de pesquisa do estudo;
 4. **Discussão dos Resultados:** Refere-se as discussões realizadas de acordo com os resultados obtidos de acordo com três aspectos: (i) foco da atividade de teste, (ii) automação do processo de teste e (iii) processo de teste de regressão;
- **Fase 3: Entrevistas.** Esta fase refere-se a realização de entrevistas com profissionais que possuem experiência no desenvolvimento de aplicativos Android. O estudo visa complementar os resultados obtidos com o *Survey*, explorando lacunas ainda inexploradas na etapa anterior, tais como identificar quais requisitos são esperados pelos profissionais em ferramentas de teste de aplicativos para Android. Além disso, o estudo busca compreender como os profissionais fazem teste de regressão. O método definido em Hove e Anda (2005) será empregado neste estudo. Essa fase foi composta pelas seguintes etapas:
 1. **Metodologia da Pesquisa:** Refere-se ao processo metodológico utilizado para realização do estudo, considerando: identificação do público alvo, definição do protocolo, aplicação do projeto piloto, seleção dos participantes, condução das entrevistas, transcrição das entrevistas, transcrição das entrevistas e *coding* das entrevistas.
 2. **Análise dos Resultados:** Nesta etapa serão apresentados os resultados obtidos com as entrevistas considerando o perfil dos participantes e as questões de pesquisa.
 3. **Discussão dos Resultados:** Refere-se as discussões realizadas de acordo com os resultados obtidos de acordo com quatro aspectos: (i) nível de conhecimento e automação de teste, (ii) linguagens e automação, (iii) ferramentas para automação e (iv) teste de regressão.

- **Fase 4: *Síntese Temática*.** Esta etapa apresenta a síntese temática sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android, utilizando como fontes de dados a literatura, o *Survey* e as entrevistas. Será adotada a metodologia definida por (CRUZES; DYBÅ, 2011a) e refinada por (PRATES, 2015). Essa fase foi composta pelas seguintes etapas:
 1. **Extração de Dados:** Esta etapa refere-se ao processo de extração de dados dos estudos utilizados.
 2. **Codificação:** Esta etapa refere-se a identificação e codificação dos conceitos, categorias, achados e resultados interessantes do conjunto de dados analisado.
 3. **Tradução de códigos em temas:** Nesta etapa apresentaremos o processo de tradução de códigos em temas.
 4. **Criação de um modelo temático:** Esta etapa refere-se ao processo de estabelecer as relações entre os temas e criação de um macro tema.

1.5 ESTRUTURA DO TRABALHO

Os demais capítulos desta dissertação estão estruturados como segue:

- **Capítulo 2:** Referencial Teórico;
- **Capítulo 3:** Trabalhos Relacionados;
- **Capítulo 4:** *Survey*;
- **Capítulo 5:** Entrevistas;
- **Capítulo 6:** Síntese Temática;
- **Capítulo 7:** Considerações Finais.

2

REFERENCIAL TEÓRICO

O presente capítulo apresenta os conceitos fundamentais necessários para a compreensão do trabalho proposto: evolução e qualidade de software, teste de software e teste de regressão.

2.1 EVOLUÇÃO E QUALIDADE DE SOFTWARE

O desenvolvimento de um software não termina com a entrega do sistema, mas, refere-se a um processo contínuo durante toda a sua vida útil. Após a fase de entrega, para que um software se mantenha útil, faz-se necessário a aplicação de atividades e processos de manutenção. Como consequência, tem-se a geração de novas versões que possuem alterações em relação a versão anterior, onde o período entre versões pode ser de curto, médio e longo prazo. Este processo denomina-se evolução de software (CHAPIN et al., 2001).

A norma IEEE 1219-1998 (*IEEE Standard for Software Maintenance*) (IEEE, 1998) define manutenção de software como a modificação de um produto de software após a entrega para corrigir falhas, melhorar o desempenho ou outros atributos ou para adaptar o produto a um ambiente modificado. Há quatro principais tipos de manutenção de software: *perfectiva*, *corretiva*, *adaptativa* ou *preventiva*. A manutenção *perfectiva* é qualquer modificação de um produto de software após a entrega cujo objetivo é melhorar o desempenho ou a capacidade de manutenção. A manutenção *corretiva* diz respeito às modificações reativas de um produto de software executadas após a entrega para corrigir falhas descobertas. A manutenção *adaptativa* é a modificação de um produto de software executada após a entrega para manter um programa de computador utilizável em um ambiente modificado ou em mudança. A manutenção *preventiva* refere-se a modificações de software realizadas com o objetivo de prevenir problemas antes que eles ocorram (MENS; DEMEYER, 2008).

Um dos desafios do processo de evolução de software é manter a qualidade do produto modificado. O termo qualidade é apresentado com diferentes conceitos na literatura. A norma ISO 9000 define qualidade como o grau em que um conjunto de características

inerentes preenchem os requisitos (SCHMAUCH, 1995). Para Juran e Gryna (1970) e Crosby (1979), o conceito de qualidade está relacionado à “conformidade com requisitos” e “adequação para uso”, respectivamente. Para HIRAMA (2011), qualidade é algo difícil de ser definido. Ainda mais difícil é garantir a qualidade do produto, visto que a percepção de qualidade passa pelas expectativas que cada pessoa tem sobre produto ou serviço.

Garantir a qualidade de um software durante seu desenvolvimento e evolução é uma tarefa relevante. Segundo HIRAMA (2011), garantia de qualidade é o processo geral de definição de como a qualidade de software pode ser atingida e como a organização de desenvolvimento sabe que o software tem o nível de qualidade necessário. Para o autor, a qualidade de um software está relacionada à quantidade de defeitos descobertos (e corrigidos). Quanto mais defeitos são descobertos e corrigidos ainda durante as fases de desenvolvimento, maior é o potencial de entrega de software com qualidade.

Na literatura, a atividade de teste apresenta-se como uma estratégia viável para descobertas de defeitos em um software. A próxima seção introduz os conceitos associados a esta atividade.

2.2 TESTE DE SOFTWARE

2.2.1 Validação, Verificação e Teste

As atividades de Verificação e Validação (VV) são comumente categorizadas em estáticas e dinâmicas. As estáticas são as que não requerem a execução ou mesmo a existência de um programa ou modelo executável para serem conduzidas. As dinâmicas são as que se baseiam na execução de um programa ou de um modelo (DELAMARO et al., 2007).

Atividades de VV permitem que se determine sistematicamente se os requisitos de um sistema estão sendo corretamente tratados e implementados. Por sua vez, a atividade de testes tem por objetivo descobrir defeitos no software, considerando aspectos estruturais e lógicos. De modo complementar, todas essas atividades contribuem diretamente para o atendimento dos prazos e custos do projeto. Portanto, conhecer os conceitos associados é de suma importância para a obtenção de produtos de maior qualidade, bem como o alcance de melhorias no processo de desenvolvimento, e até mesmo em termos de produtividade. As atividades de Verificação, Validação e Teste (VVT) auxiliam na garantia da qualidade do software (HIRAMA, 2011).

No que tange à atividade de testes, é importante compreender a distinção entre alguns conceitos relacionados, conforme apresentado no padrão de testes de software da IEEE (IEEE, 1990). Um **erro**, do inglês (*error, mistake*), refere-se a uma ação humana que produz um resultado incorreto. Os desenvolvedores cometem erros (enganos) quando interpretam mal as necessidades dos clientes e os usuários cometem erros (enganos) quando operam um sistema em desacordo com as intenções dos desenvolvedores. Um **defeito**, do inglês (*bug, fault, defect*) refere-se a algo implementado dentro de um artefato. São requisitos inconsistentes com as necessidades dos clientes e requisitos funcionais do sistema em desacordo com os requisitos de negócio. E uma **falha**, do inglês *failure*, é a incapacidade de um sistema ou componente em executar as funções requeridas dentro de um nível de desempenho requerido, como exemplo, a famosa “tela azul” do sistema

operacional Windows, da Microsoft (IEEE, 1990).

2.2.2 Atividade de Teste de Software

A atividade de teste deve acontecer desde o momento da concepção do software, sendo realizada durante todo o processo de desenvolvimento. Um procedimento de teste completo é composto por algumas etapas, conforme apresentado na Figura 2.1. O teste procura descobrir defeitos quando o comportamento do software não está correto ou não está em conformidade com o que foi especificado.

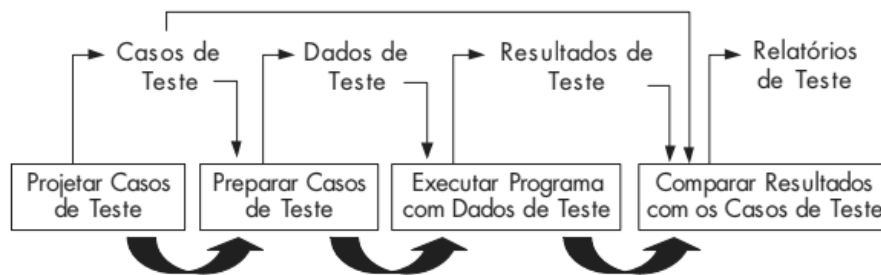


Figura 2.1 Procedimento de Teste (HIRAMA, 2011)

Um dos resultados do procedimento de teste é gerar os casos de teste para que posteriormente possam ser executados. Um bom caso de teste é aquele que tem uma grande probabilidade de revelar um defeito ainda não descoberto (HIRAMA, 2011).

O documento básico para a atividade de teste é o plano de testes, no qual se define: objetivos para cada tipo (ou fase de teste), estratégias de teste, cronograma e responsabilidades, procedimentos e padrões a serem usados na execução e na elaboração do relatório de testes, critérios para a conclusão do teste, bem como o sucesso de cada teste (HIRAMA, 2011).

O processo de execução dos testes pode ser conduzido de forma manual, por um profissional que testa o software a partir de um roteiro a ser seguido, ou de forma automatizada, com a utilização de ferramentas que executam os testes e reportam os resultados. Para Ammann e Offutt (2008), o teste de software é caro, exige mão de obra intensiva e requer até 50% dos custos de desenvolvimento de software. Para Sommerville (2016), aproximadamente 60% dos custos de software são custos de desenvolvimento e 40% são custos de teste. Para software customizado, os custos de evolução geralmente excedem os custos de desenvolvimento. Assim, faz-se necessário buscar automatizar o processo de execução dos testes de software para reduzir o custo, minimizar o erro humano e facilitar o teste de regressão.

2.2.3 Técnicas de Teste de Software

Segundo Delamaro et al. (2007), as duas técnicas mais amplamente aceitas na academia e na indústria são as técnicas de testes **funcionais** e as **estruturais**, comumente conhecidas como *teste de caixa preta* e *teste de caixa branca*, respectivamente.

A técnica **funcional** é utilizada para projetar casos de teste em que o programa ou sistema de software é considerado uma caixa preta. Para testá-lo, fornece-se entradas e avalia-se as saídas geradas para verificar se estão em conformidade com os objetivos especificados. No teste funcional os detalhes de implementação não são considerados e o software é avaliado segundo a visão do usuário. Segundo HIRAMA (2011) nesta técnica os testes precisam atender às seguintes características:

- O projeto de casos de teste usa a estrutura de controle procedimental do software (fluxo de controle do software) para derivar casos de teste;
- Deve garantir que todos os caminhos independentes dentro de um módulo tenham sido exercitados pelo menos uma vez;
- Deve exercitar todas as decisões lógicas para valores falsos ou verdadeiros;
- Deve executar todos os laços em suas fronteiras e dentro de limites operacionais;
- Deve exercitar as estruturas de dados internas para garantir a sua validade.

Na técnica **estrutural**, os requisitos de teste são estabelecidos com base em uma dada implementação, e requer a execução de partes ou de componentes elementares do programa. A técnica busca testar os caminhos lógicos do software, utilizando casos de teste para provar conjuntos específicos, estruturas condicionais, definições e uso de variáveis. Segundo HIRAMA (2011) nesta técnica os testes precisam atender às seguintes características:

- Concentram-se nos requisitos funcionais do software;
- São uma abordagem complementar aos testes estruturais.

Existe ainda a técnica denominada grey-box, onde os profissionais possuem acesso parcial ao código do software a ser testado. Nesta técnica, embora o estado interno de uma implementação ainda permaneça oculto como na técnica funcional, as entradas habilitadas no estado de implementação atual pode ser revelado (SACHTLEBEN; PELESKA, 2021).

2.2.4 Tipos de Teste

A atividade de testes é dividida em fases com objetivos distintos (DELAMARO et al., 2007). Segundo HIRAMA (2011), os tipos de teste mais comuns realizados para cobrir os testes estruturais e funcionais são: teste de unidade, teste de integração, teste de validação e teste de sistema.

- **Teste de unidade:** tem como foco as menores unidades de um programa, podendo ser funções, procedimentos, métodos ou classes. O objetivo é testar os componentes isoladamente, para verificar o funcionamento conjunto dos algoritmos e as estruturas de dados. Espera-se então que sejam identificados erros relacionados a algoritmos incorretos ou mal implementados, estruturas de dados incorretas, ou simples erros de programação.

- **Teste de integração:** deve ser realizado após serem testadas as unidades individuais, e a ênfase é dada na construção da estrutura do sistema. O objetivo é testar um conjunto de módulos verificando o seu funcionamento com foco nas suas interfaces entre os módulos. Quando as diversas partes do software são colocadas para trabalhar juntas, verifica-se se a interação entre elas funciona de maneira adequada e não ocasiona erros. Faz-se necessário um grande conhecimento das estruturas internas e das interações existentes entre as partes do sistema.
- **Teste de validação:** tem como objetivo testar o software como um todo verificando se todas as exigências funcionais, comportamentais e de desempenho foram atendidas. A referência usada para os testes é o documento de requisitos, que descreve tanto os requisitos funcionais quanto os não-funcionais do software. O teste é realizado pela equipe de desenvolvimento de software;
- **Teste de sistema:** tem como objetivo medir o sistema em diferentes cenários verificando se todos os elementos do sistema (hardware, software, banco de dados e pessoas) foram adequadamente integrados e realizam as funções requeridas. A referência usada para os testes é a especificação arquitetural do sistema, um documento que contém a descrição funcional do sistema, dos subsistemas e do seu desempenho no ambiente operacional.

Existem ainda outros testes que podem ser aplicados a sistemas de software. São eles: teste de recuperação, teste de proteção, teste de estresse, teste de desempenho, dentre outros, que visam validar requisitos não-funcionais do software, i.e., aqueles ligados aos atributos de qualidade esperados. O teste de recuperação força o sistema a apresentar falhas de diversas maneiras e verifica se a recuperação (reiniciação do sistema e recuperação de dados) é adequadamente executada. O teste de proteção tenta verificar se todos os mecanismos de proteção embutidos em um sistema funcionam contra acessos indevidos. O teste de estresse confronta o software com situações anormais (alta exigência de recursos, alto número de interrupções, alta taxa de entrada de dados, alta busca de dados em disco). E o teste de desempenho é utilizado no contexto de um sistema integrado (tempos envolvidos, ciclos de processador, interrupções) (HIRAMA, 2011).

Outros tipos de teste de software relevantes são os testes alfa e beta, que estão relacionados ao teste de aceitação. Em virtude das dificuldades encontradas pelos desenvolvedores para preverem como o cliente usará o programa, ou como ele irá interpretar os manuais de usuários, alguns testes são realizados para compreender como será a aceitação do cliente em relação ao software desenvolvido.

2.2.5 Testes de aplicativos para Android

Segundo Glauber (2019), o Android não é apenas um sistema operacional, mas um conjunto completo de software para dispositivos móveis que inclui: um Sistema Operacional, um *middleware* e aplicações-chave. O Android é *open-source* e seu código fonte está disponível para *download*¹. Assim como o código-fonte, a documentação para desenvol-

¹<https://source.android.com/> - Acessado em 04/09/2021.

vimento e teste também está disponível para consulta².

Desenvolver e testar aplicativos para Android apresentam alguns desafios específicos, tais como concorrência, segurança, performance, eficiência energética, compatibilidade, detecção de defeitos (KONG et al., 2019). Assim, testar um aplicativo para Android é uma tarefa que tende a exigir um esforço da equipe.

Os testes de aplicativos devem ser realizados em paralelo ao longo de todo ciclo de desenvolvimento, com o objetivo de identificar falhas de forma breve e assim entregar um software estável e confiável. Neste contexto, existem técnicas que podem ser utilizadas, tais como seguir um projeto de testes onde sejam identificados os cenários de teste ou, com a realização de testes exploratórios, onde o testador investiga cada funcionalidade da aplicação (GLAUBER, 2019).

Com o objetivo de reduzir o esforço de teste no processo de desenvolvimento de aplicativos para Android, tanto a literatura quanto o mercado propõem ferramentas para automatizar o processo de testes em aplicativos Android. As ferramentas podem ser gratuitas ou proprietárias, e estão focadas em diferentes tipos de testes no processo de desenvolvimento dos aplicativos. A Tabela 4.2 apresentará uma lista de ferramentas utilizadas para automação de testes para aplicativos Android.

Segundo Glauber (2019), os testes automatizados no Android podem ser divididos basicamente em dois tipos:

- **Unit tests (testes de unidade):** Estes testes rodam na máquina virtual do Java, do inglês *Java Virtual Machine* (JVM) e tem como foco os componentes independentes do *framework* do Android;
- **Instrumentation tests (testes instrumentados):** Em virtude da dependência de componentes específicos do Android, são executados em um dispositivo real ou através do uso de emuladores.

2.3 TESTE DE REGRESSÃO

2.3.1 Definição

Segundo a norma IEEE Std 610.12-1990, que apresenta o glossário padrão de termos da Engenharia de Software IEEE (1990), teste de regressão pode ser conceituado como a repetição seletiva de um sistema ou componente para verificar se as modificações não causaram efeitos não-intencionais e se o sistema ou componentes ainda estão em conformidade com o requisito especificado. Segundo Ammann e Offutt (2008), o teste de regressão é o processo de testar novamente o software que foi modificado. Para Silveira-Neto et al. (2010), o teste de regressão é uma forma eficaz de testar o software após modificações.

À medida que os desenvolvedores mantêm um sistema de software, eles periodicamente fazem teste de regressão, na esperança de encontrar erros causados por suas alterações e fornecer confiança de que suas modificações estão corretas. Para realizar essa atividade, os desenvolvedores geralmente criam um conjunto inicial de testes, que são reutilizáveis. A estratégia de teste de regressão mais simples é a *reexecução de todos os casos de*

²<https://developer.android.com/> - Acessado em 04/09/2021.

teste. Essa abordagem, porém, torna-se provavelmente inviável por exigir uma quantidade muito grande de tempo para a sua execução completa (GRAVES et al., 2001). Para Ammann e Offutt (2008), o teste de regressão constitui a grande maioria dos esforços de teste no desenvolvimento de software e é uma parte essencial de qualquer processo de desenvolvimento de software viável.

Segundo Leung e White (1989), os testes de regressão podem ser **corretivos** ou **progressivos**. A Tabela 2.1 apresenta algumas diferenças fundamentais entre ambos.

Tabela 2.1 Teste de Regressão Corretivo & Progressivo (LEUNG; WHITE, 1989)

Teste de Regressão Corretivo	Teste de Regressão Progressivo
<ul style="list-style-type: none"> • A especificação não é alterada • Envolve pequenas modificações no código (por exemplo, adicionar e excluir instruções) • Geralmente feito durante o desenvolvimento e a manutenção corretiva • Muitos casos de teste podem ser reutilizados • Chamado em intervalos irregulares 	<ul style="list-style-type: none"> • A especificação é alterada • Envolve modificação importante (por exemplo, adicionar e excluir módulos) • Geralmente feito durante a manutenção adaptativa e aperfeiçoada • Menos casos de teste podem ser reutilizados • Chamado em intervalos regulares

2.3.2 Técnicas de Teste de Regressão

Para Ammann e Offutt (2008), uma técnica de teste é precisa na medida em que omite testes de regressão que não são reveladores de modificação. Uma técnica é eficiente na medida em que determina o subconjunto apropriado do conjunto de testes de regressão. Por sua vez, uma técnica é geral no grau que se aplica a uma ampla variedade de situações práticas.

As técnicas podem ser categorizadas, para que sejam avaliadas e comparadas, em quatro categorias: **inclusividade**, **precisão**, **eficiência** e **generalidade**. A *inclusividade* mede o quanto uma técnica escolhe testes que farão com que o programa modificado produza uma saída diferente do programa original expondo falhas causadas por modificações. A *precisão* mede a capacidade de uma técnica evitar a escolha de testes que não farão com que o programa modificado produza resultados diferentes do programa original. A *eficiência* mede o custo computacional e, portanto, a praticidade de uma técnica. A *generalidade* mede a capacidade de uma técnica lidar com construções de linguagem realistas e diversificadas, modificações de código arbitrariamente complexas e aplicativos de teste realistas (ROTHERMEL; HARROLD, 1996).

YOO e HARMAN (2012) classificam as técnicas de teste de regressão como técnicas de **seleção**, técnicas de **minimização** e técnicas de **priorização**. Essas técnicas serão apresentadas a seguir.

2.3.2.1 Técnicas de Seleção

As técnicas de seleção, do original em inglês *Regression Test Selection* (RTS), buscam reduzir o conjunto de casos de teste a serem reexecutados, definindo um subconjunto de casos de teste do conjunto de testes. Nessas técnicas os casos de teste são selecionados porque sua execução é relevante para as mudanças entre a versão anterior e a versão atual do sistema, do inglês, *System Under Test* (SUT) (YOO; HARMAN, 2012).

Dado: O programa P , a versão modificada de P , P' e um conjunto de testes T .

Problema: Encontre um subconjunto de T , T' , com o qual testar P' .

Essas técnicas reduzem o custo do teste de um programa modificado, pois, reutilizam testes existentes e identificam partes do programa modificado ou sua especificação que deve ser testada. Para Leung e White (1991), uma técnica de seleção é mais econômica que a técnica de reexecução de todos os casos de teste, se o custo de selecionar um subconjunto reduzido de testes a executar for menor que o custo de executar os testes que a técnica de seleção omitir.

2.3.2.2 Técnicas de Minimização

As técnicas de minimização do conjunto de testes visam identificar casos de teste redundantes e removê-los do conjunto de testes para reduzir o tamanho do conjunto de testes (YOO; HARMAN, 2012).

Dado: Um conjunto de testes T , um conjunto de requisitos de teste r_1, \dots, r_n , que deve ser satisfeito para fornecer o teste “adequado” desejado do programa e subconjuntos de T , T_1, \dots, T_n , um relacionado a cada um dos r_i s de tal forma que qualquer um dos casos de teste t_j pertencentes ao T pode ser usado para atingir o requisito r_i .

Problema: Encontre um conjunto representativo, T' , de casos de teste de T que satisfaça todos os r_i s.

As técnicas de minimização buscam reduzir o custo do teste de regressão, encontrando um subconjunto mínimo em termos do número de casos de teste que preserva a cobertura com relação a um determinado critério de cobertura do conjunto de teste original (WONG et al., 1995).

2.3.2.3 Técnicas de Priorização

As técnicas de priorização procuram o subconjunto ideal de casos de teste para que o testador obtenha o máximo benefício, mesmo se o teste for prematuramente interrompido em algum ponto arbitrário (YOO; HARMAN, 2012).

Dado: Um conjunto de testes T , o conjunto de permutações de T , PT e uma função de PT para números reais, $f : PT \rightarrow R$.

Problema: Para encontrar $T' \in PT \rightarrow (\forall T'')(T'' \in PT)(T'' \neq T')[f(T') \geq f(T'')]$.

As técnicas de priorização trazem benefícios para os testadores, tais como: aumentar a taxa de detecção de defeitos revelando as falhas mais cedo em uma execução de teste de regressão, aumentar a cobertura do código fonte permitindo que um critério de cobertura de código seja atendido no processo de teste, aumentar a confiabilidade do sistema, aumentar a taxa em que as falhas de alto risco são detectadas por um conjunto de casa de testes e aumentar a probabilidade de revelar falhas relacionadas a alterações de códigos específicas no início do processo de teste de regressão (ROTHERMEL et al., 2001).

2.3.3 Classificação dos casos de teste

YOO e HARMAN (2012) categorizam os casos de teste em cinco classes. As três primeiras classes consistem em casos de teste que já existem em T .

- **Reutilizável:** Os casos de teste reutilizáveis executam apenas as partes do programa que permanecem inalteradas entre duas versões, ou seja, as partes do programa que são comuns a P e P' . Não é necessário executar esses casos de teste para testar P' ; no entanto, eles são chamados de reutilizáveis porque ainda podem ser retidos e reutilizados para o teste de regressão das futuras versões de P ;
- **Retestável:** Casos de teste retestáveis executam as partes de P que foram alteradas em P' . Assim, os casos de teste retestáveis devem ser reexecutados para testar P' ;
- **Obsoleto:** Os casos de teste podem se tornar obsoletos porque (1) sua relação de entrada/saída não é mais correta devido a mudanças nas especificações, (2) eles não testam mais o que foram projetados para testar devido a modificações no programa ou (3) casos de teste “estruturais” que não mais contribuem para a cobertura estrutural do programa. As duas classes restantes consistem em casos de teste que ainda precisam ser gerados para o teste de regressão de P' ;
- **Novo-estrutural:** Novos casos de teste estruturais testam as construções modificadas do programa, fornecendo cobertura estrutural das partes modificadas em P' ;
- **Nova especificação:** Casos de teste de nova especificação testam as especificações do programa modificado, testando o novo código gerado a partir das partes modificadas das especificações de P' .

2.4 SÍNTESE DO CAPÍTULO

Neste capítulo foram apresentados os conceitos de evolução e qualidade de software, teste de software e teste de regressão.

O uso de técnicas de teste de regressão corrobora com a redução do esforço e do custo da atividade de teste de software em cenários de evolução de aplicativos, diminuindo o número de casos de teste a serem reexecutados. O conhecimento e a aplicação destas técnicas auxilia na garantia da qualidade das releases dos aplicativos.

No próximo capítulo serão apresentados os trabalhos relacionados a esta dissertação.

3

TRABALHOS RELACIONADOS

Neste capítulo serão apresentados os trabalhos relacionados a esta dissertação, sendo categorizados em três temáticas: técnicas de teste de regressão para Android, testes para aplicações Android, e aplicação industrial de testes de regressão.

3.1 TÉCNICAS DE TESTE DE REGRESSÃO PARA ANDROID

Nesta seção apresentamos os trabalhos relacionados que abordam técnicas de teste de regressão implementadas por ferramentas para Android.

1. **(JHA et al., 2019)** Neste trabalho os autores apresentam um *framework* para testar aplicativos Android reutilizando casos de teste. O *framework* aproveita casos de teste e conhecimento de domínio de aplicativos Android de código aberto existentes para testar novos aplicativos Android, utilizando casos de teste que são extraídos pela mineração de aplicativos Android de código aberto existentes e analisa os aplicativos que contêm casos de teste para obter o conhecimento do domínio. Com base no conhecimento do domínio, o *framework* categoriza os casos de teste extraídos, que são generalizados para reutilização. O *framework* então analisa o aplicativo em teste para obter o conhecimento do domínio. Finalmente, o *framework* testa o aplicativo reutilizando os casos de teste dos aplicativos existentes que correspondem ao domínio do aplicativo em teste.
2. **(KONG et al., 2019)** Neste trabalho os autores fornecem uma visão geral dos trabalhos sobre teste de aplicativos Android destacando as principais tendências, apontando as principais metodologias aplicadas e enumerando os desafios enfrentados pelas abordagens de teste do Android. Eles conduziram uma revisão sistemática da literatura, identificando 103 artigos de pesquisa relevantes publicados nas principais conferências e periódicos até 2016. Este trabalho elenca um conjunto de ferramentas produzidas em trabalhos acadêmicos para testes de aplicativos para plataforma Android. O estudo levou a várias descobertas e destacou os desafios

que os pesquisadores de testes do Android devem se esforçar para abordar no futuro. Apresentaram como proposta algumas direções de pesquisa concretas em que abordagens de teste são necessárias para resolver problemas recorrentes em atualizações de aplicativos, aumentos contínuos de tamanhos de aplicativos, bem como a fragmentação do ecossistema Android.

3. **(CHOI et al., 2018)** Neste trabalho os autores propõe uma técnica heurística que ajuda a criar um pequeno conjunto de testes de regressão para um aplicativo Android a partir de um grande conjunto de testes gerado por uma ferramenta de teste Android *Graphical User Interface* (GUI) automatizada. A proposta é que se for possível identificar e remover algumas formas comuns de redundâncias introduzidas por ferramentas existentes de teste de GUI, pode-se reduzir o tempo necessário para minimizar um conjunto de testes de GUI. Os autores implementaram o algoritmo em um protótipo de ferramenta chamada DetReduce. Aplicaram DetReduce a vários aplicativos Android e descobriram que DetReduce reduz o tempo de execução de um conjunto de testes.
4. **(CHANG et al., 2018)** Neste trabalho os autores propõe uma abordagem para automatizar a manutenção do *script* de teste da GUI para aplicativos Android, denominada CHATEM. A abordagem CHATEM extrai automaticamente as mudanças entre as duas GUIs e gera ações de manutenção para cada mudança, que são então combinadas para formar as ações de manutenção para o teste afetado *scripts* e os *scripts* de teste originais, utilizando como entrada os modelos para as GUI do aplicativo de base e versão atualizada. Eles realizaram uma avaliação experimental em 16 aplicativos Android, onde a abordagem CHATEM foi capaz de manter automaticamente os *scripts* de teste para que no geral mais de 95% dos comportamentos restantes testados antes ainda sejam testados e quase 80% das ações de teste reutilizáveis sejam retidas nos testes de resultados.
5. **(JIANG et al., 2018)** Este trabalho apresenta um estudo sobre técnicas de seleção de teste de regressão. Os pesquisadores propõem a criação de uma técnica denominada **ReTestDroid** que serve para modelar invocações de tarefas assíncronas, ciclo de vida de atividades baseado em fragmentos e código nativo dentro do gráfico de fluxo de controle de uma aplicação Android. A técnica é implementada e testada em cinco Aplicativos para dispositivos móveis (APPS) Android.
6. **(LI et al., 2017)** Neste trabalho os autores propõe uma nova abordagem para manter automaticamente *scripts* de teste de GUI de aplicativos móveis para testes de regressão, denominada de ATOM. ATOM usa um modelo de sequência de eventos para abstrair possíveis sequências de eventos em uma GUI e um ESM delta para abstrair as alterações feitas na GUI. Dado ambos os modelos como entrada, o ATOM atualiza automaticamente os *scripts* de teste escritos para a versão base de um aplicativo para refletir as mudanças. Os autores realizaram um experimento com 22 versões de 11 aplicativos Android. Neste experimento, a abordagem ATOM atualizou todos os *scripts* de teste afetados pela mudança de versão; os scripts

atualizados alcançam mais de 80% da cobertura dos *scripts* originais no aplicativo da versão base; exceto um conjunto de *scripts* atualizados, todos preservam mais de 60% das ações nos *scripts* de teste originais.

7. **(DO et al., 2016)** Este trabalho apresenta um estudo de *bugs* reais em aplicativos para Android que demonstra a existência de erros de regressão. Como solução para este problema, os pesquisadores propõem a técnica de seleção de teste de regressão denominada REDROID que aproveita a combinação de análise de impacto estático e cobertura dinâmica de código, para identificar um subconjunto de casos de teste para re-execução na versão modificada do aplicativo. A técnica foi implementada e testada em um estudo empírico.
8. **(GRØNLI; GHINEA, 2016)** Este trabalho apresenta uma arquitetura extensível e um protótipo conceitual que mostra e combina a execução de teste para multiplataformas mobile com medições de desempenho, contribuindo para um processo de garantia de qualidade, automatizando partes de um teste de regressão para aplicativos mobile multiplataforma.
9. **(GÓMEZ et al., 2016)** Neste trabalho os autores apresentam uma abordagem denominada DUNE, que tem como objetivo detectar automaticamente degradações de desempenho da UI em aplicativos Android, levando em consideração as diferenças de contexto. Inicialmente, a abordagem DUNE constrói um modelo de conjunto das métricas de desempenho da UI de um aplicativo a partir de um repositório de execuções de teste históricas que são conhecidas como aceitáveis, para diferentes configurações de contexto. Em seguida, a DUNE usa esse modelo para sinalizar desvios de desempenho da UI (regressões e otimizações) em novas execuções de teste. A abordagem DUNE foi avaliada em defeitos reais de desempenho da UI relatados em dois aplicativos Android e um defeito injetado manualmente em um terceiro aplicativo. Assim, foi possível observar que este conjunto de ferramentas pode ser usado com sucesso para detectar regressões de desempenho da UI em uma granularidade fina.
10. **(HU; NEAMTIU, 2016)** Neste trabalho os autores apresentam VALERA, uma ferramenta de gravação e reprodução versátil, mas leve, para Android. VALERA pode ser usada como uma ferramenta de *replay* eficaz em telefones reais e emuladores. Ela foi avaliada em mais de 50 aplicativos Android populares. O estudo demonstra que VALERA pode ser usada em muitos cenários de desenvolvimento: reprodução de bug, teste de regressão, reprodução e verificação de corrida orientada por evento, teste de mutação por meio de *replay* difuso e teste de aplicativo cruzado.
11. **(BAUERSFELD, 2013)** Neste trabalho os autores propõem uma ferramenta de teste de regressão GUI chamada GUIdiff. A ferramenta executa duas versões diferentes de um *System Under Test (SUT)* lado a lado, compara os estados da GUI entre si e apresenta a lista dos desvios detectados ao testador. A ferramenta é semiautomática no sentido de que encontra as diferenças completamente automáticas e que o testador as rotula como falhas ou falsos positivos.

12. **(SZABÓ et al., 2012)** Neste trabalho os autores apresentam a aplicação de refatoração e teste de regressão, que suporta a modificação de software de forma sistemática e controlada. A refatoração do banco de dados e o teste de regressão são apresentados no desenvolvimento de um aplicativo móvel experimental baseado em Android. Os autores avaliam os resultados dos experimentos e verificam se estão corretos.
13. **(AMALFITANO et al., 2011)** Este trabalho aborda o problema de teste automático de aplicativos móveis desenvolvidos para a plataforma Google Android e apresenta uma técnica para teste de travamento rápido e teste de regressão de aplicativos Android. A técnica é baseada em um rastreador que constrói automaticamente um modelo da GUI do aplicativo e obtém casos de teste que podem ser executados automaticamente. A técnica é suportada por uma ferramenta para rastrear o aplicativo e gerar os casos de teste. No artigo os autores apresentam um exemplo de uso da técnica e da ferramenta para testar um aplicativo Android de tamanho real que mostra preliminarmente a eficácia e usabilidade da abordagem de teste proposta.

Os trabalhos de (JHA et al., 2019), (CHOI et al., 2018), (CHANG et al., 2018), (JIANG et al., 2018), (LI et al., 2017), (DO et al., 2016), (GRØNLI; GHINEA, 2016), (GÓMEZ et al., 2016), (HU; NEAMTIU, 2016), (BAUERSFELD, 2013), (SZABÓ et al., 2012) e (AMALFITANO et al., 2011) apresentam estudos sobre técnicas de teste de regressão para APPS Android, e propõem como solução a criação de ferramenta, realizando estudos para prover evidências empíricas sobre sua eficácia. O trabalho de (KONG et al., 2019) apresenta um conjunto de ferramentas apresentadas na literatura para automação de testes de aplicativos para Android.

3.2 TESTES PARA APLICAÇÕES ANDROID

Nesta seção apresentamos os trabalhos relacionados referente a estudos feitos com a participação de profissionais sobre testes para aplicações Android.

1. **(VÁSQUEZ et al., 2017)** Neste estudo, os autores investigaram colaboradores de projetos Android de código aberto sobre práticas e preferências para projetar e gerar casos de teste, práticas de teste automatizadas e percepções de métricas de qualidade, como cobertura de código. Eles apresentaram os resultados de um estudo empírico, com 102 respostas, de aplicativos móveis de código aberto hospedados no GitHub, revelando as opiniões dos desenvolvedores sobre projetos de código aberto. Os resultados da pesquisa mostram a necessidade de ferramentas / abordagens para testar aplicativos com base em modelos. Segundo eles, a cobertura do código fonte não é uma medida importante da qualidade dos casos de teste. Esta pesquisa também mostra que, embora existam ferramentas, o uso delas para automatizar o processo de teste ainda é baixo.
2. **(KOCHHAR et al., 2015)** Este trabalho apresenta um estudo empírico conduzido em duas etapas. Na primeira etapa, os autores analisaram o código fonte de

teste de 600 projetos open source para compreender o estado atual dos testes de software na comunidade de desenvolvimento Android. Na segunda etapa, eles investigaram os desenvolvedores de aplicativos da Microsoft, obtendo 127 respostas. Eles apontaram a importância da atividade de teste de software, especificamente, com o crescimento da comunidade de aplicativos e a importância da atividade de teste para a criação de aplicativos de qualidade. Eles observaram que os desenvolvedores de aplicativos preferem usar estruturas de teste padrão, como JUnit, e também ferramentas de teste, como Monkeyrunner, Robotium e Robolectric. Além disso, muitos desenvolvedores do Android executam um processo de teste manual, sem a ajuda de qualquer estrutura ou ferramenta de teste.

3. (**CHOUDHARY et al., 2015**) Neste estudo, os autores realizaram uma comparação das principais ferramentas de geração de entradas de teste para Android, a fim de entender os pontos fortes e fracos dessas abordagens. Os entrevistados avaliaram a eficácia dessas ferramentas e técnicas de acordo com quatro métricas: facilidade de uso, capacidade de trabalhar em várias plataformas, cobertura de código e capacidade de detectar falhas. Seus resultados indicaram que a ferramenta Monkey obteve a melhor cobertura, em média, relatou o número mais significativo de falhas, foi a mais fácil de usar e funcionou para todas as plataformas. Além disso, os autores apresentaram um conjunto de recursos relevantes que a ferramenta de teste deve conter, como Gerar eventos do sistema, Minimizar reinicializações e Evitar efeitos colaterais entre diferentes execuções.

Os trabalhos de (VÁSQUEZ et al., 2017), (KOCHHAR et al., 2015) e (CHOUDHARY et al., 2015) investigaram o perfil dos profissionais que trabalham com desenvolvimento e testes de aplicativos para Android.

3.3 APLICAÇÃO INDUSTRIAL DE TESTES DE REGRESSÃO

Nesta seção apresentamos um trabalho relacionado referente a um estudo feito com a participação de profissionais sobre a aplicação industrial de teste de regressão.

1. (**ALI et al., 2019**) Neste estudo os autores realizaram uma revisão sistemática da literatura com dois objetivos: permitir que pesquisadores projetassem e apresentassem pesquisas que tinham como foco a aplicabilidade e a relevância industrial do teste de regressão, e para facilitar a adoção industrial destas pesquisas. Para alcançar os objetivos propostos os pesquisadores consultaram profissionais durante o processo de realização da revisão sistemática. Os autores mapearam 38 artigos relatando a avaliação de técnicas de teste de regressão em ambientes industriais.

O trabalho de (ALI et al., 2019) busca compreender a aplicabilidade industrial do teste de regressão e suas técnicas.

3.4 SÍNTESE DO CAPÍTULO

Neste capítulo apresentamos estudos com temáticas relacionadas a esta dissertação, categorizando-os em três macro temáticas: Técnicas de teste de regressão para Android, Testes para

aplicações Android, e Aplicação industrial de testes de regressão. Esta dissertação diferencia-se dos trabalhos acima citados, visto que tem como foco investigar a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android. Na nossa metodologia utilizamos a literatura, assim como, a consulta direta com profissionais que atuam neste domínio específico, aplicações para Android, por meio do *Survey* e das entrevistas. Ainda, nosso estudo elenca os requisitos necessários para ferramentas que executam teste de regressão identificando a oportunidade de implementação de novas ferramentas baseadas nas técnicas apresentadas pela literatura.

No próximo capítulo será apresentado o *Survey* realizado com profissionais e pesquisadores que atuam em projetos de aplicativos para Android.

4

SURVEY

Com o objetivo de compreender como testadores e desenvolvedores realizam o teste de regressão, realizamos um estudo empírico do tipo *survey* com a participação de 100 profissionais e pesquisadores que atuam em projetos de desenvolvimento de software para a plataforma Android (LIMA et al., 2020). Este capítulo apresenta esse estudo empírico, com detalhes sobre o planejamento, a execução e a análise dos resultados obtidos.

4.1 APRESENTAÇÃO E QUESTÕES DE PESQUISA

Segundo Pfleeger e Kitchenham (2001), um *survey* é um sistema abrangente que tem como objetivo coletar informações para descrever, comparar ou explicar conhecimentos, atitudes e comportamentos.

As seguintes Questões de Pesquisa (QP) foram definidas para este estudo:

- QP1: Quão automatizado é o teste de regressão de aplicativos Android na prática?** A literatura e a indústria fornecem ferramentas para automação de testes de aplicativos para Android. Esta QP busca (i) identificar se os testadores utilizam ferramentas para testar aplicativos Android e (ii) elencar quais são as ferramentas usadas.
- QP2: Como é realizado o processo de teste durante e após a manutenção dos aplicativos Android?** A literatura apresenta diversos estudos sobre testes em cenário de evolução. Esta QP busca identificar se os testadores realizam teste de regressão e elencar quais são as estratégias de seleção empregadas para reduzir a quantidade de casos de teste a serem reexecutados.
- QP3: Quais são os motivos para não realizar testes de regressão após a manutenção de aplicativos para Android?** Esta QP tem como objetivo entender os motivos pelos quais os testadores não realizam testes de regressão em sua prática cotidiana.

4.2 METODOLOGIA DE PESQUISA

Esta seção abrange os detalhes do planejamento e procedimentos de execução. Neste estudo, utilizamos a metodologia proposta por Kasunic (2005) e os princípios de pesquisa definidos por Kitchenham e Pfleeger (2002).

4.2.1 Identificação do Público Alvo

Para obter resultados válidos foram selecionados como público-alvo estudantes e profissionais de diferentes estados do Brasil que trabalham com testes de aplicativos para Android.

4.2.2 Design do Questionário

O *survey* foi dividido em quatro seções, conforme apresentado na Tabela 4.1.

Tabela 4.1 Design do *survey*

Seção	Design
I	Refere-se ao termo de consentimento livre e esclarecido.
II	Refere-se à identificação do participante, a qual consiste em 13 perguntas, sendo a maioria obrigatória (8 perguntas de múltipla escolha, 3 subjetivas e 2 de seleção).
III	Trata do processo de teste de aplicativos Android. Consiste em 16 perguntas, sendo a maioria obrigatória (7 perguntas de múltipla escolha, 3 perguntas subjetivas e 6 perguntas na caixa de seleção).
IV	Refere-se às considerações finais.

A seguir, os objetivos de cada seção do questionário.

- **Perfil dos participantes:** identificar o perfil dos participantes, com informações sobre gênero, idade, escolaridade, área de formação, certificações de teste, nível de conhecimento, experiência em testes de software e experiência em teste de aplicativos para Android.
- **Foco da atividade de teste:** (i) identificar se os testes realizados são funcionais ou estruturais e (ii) identificar os tipos de testes realizados nos projetos atuais - teste de interface (do inglês *Graphical User Interface (GUI)*), teste de unidade, teste de integração.
- **Automação do processo de teste:** (i) identificar se os testes são realizados de forma manual ou automatizada e (ii) identificar o suporte ferramental utilizado pelos participantes. No questionário foram elencadas as ferramentas para testar aplicativos Android encontradas na literatura, assim como, as produzidas pela indústria. Assim, foi possível investigar se as ferramentas listadas na literatura também são conhecidas e utilizadas pelos participantes. A Tabela 4.2 mostra o conjunto de ferramentas de teste para aplicativos Android apresentadas aos participantes. Essas ferramentas foram identificadas com a utilização de três critérios: (i) ferramentas

para teste de aplicativos para Android disponíveis na indústria, (ii) ferramentas identificadas na literatura como resultado de implementação de técnicas de teste de regressão, e (iii) ferramentas apontadas no estudo de (KONG et al., 2019).

Tabela 4.2 Ferramentas de teste de aplicativos para Android

Ferramenta	Tipo de Teste							Fonte	Licença
	<i>Func.</i>	<i>GUI</i>	<i>Reg.</i>	<i>Perf.</i>	<i>Stress</i>	<i>Unit.</i>	<i>Est.</i>		
APPIUM	•	•						M	G
ATOM			•					L	G
CALABASH	•	•						M	G
CHATEM			•					L	G
DETREDUCE			•					L	G
ESPRESSO	•	•						M	G
GUIDIFF			•					L	G
KATALON	•	•						M	G
KMAX				•				M	P
KOBITON	•	•		•				M	P
MONKEY					•			M	G
MONKEY RUNNER	•	•	•					M	G
RANOREX	•	•	•					M	P
REDROID			•					L	G
RETESTDROID			•					L	G
ROBOLECTRIC						•	•	M	G
ROBOTIUM	•	•						M	G
SEE TEST	•	•		•				M	P
SQUISH	•	•	•					M	P
TELERIK	•	•						M	P
TEMA			•					L	G
TESTCOMPLETE	•	•	•					M	P
TESTINGBOT	•	•		•				M	P
UFT	•	•	•	•				M	P
UI AUTOMATOR	•	•						M	G

Legenda: Fonte (**L**: Literatura, **M**: Mercado), Tipo de Licença (**G**: Gratuita, **P**: Proprietária), *Func.*: Teste Funcional, *Reg.*: Teste de Regressão, *Perf.*: Teste de Performance, *Unit.*: Teste de Unidade, *Est.*: Teste Estrutural

- **Processo de teste de regressão:** identificar se os participantes realizam testes de regressão após realizar manutenção em aplicativos Android. Busca-se descobrir: (i) se o processo é feito manualmente ou de modo automatizado (com o suporte de ferramenta de apoio), (ii) se é aplicada alguma técnica para selecionar os casos de teste a serem reexecutados, (iii) se são utilizadas ferramentas encontradas na

literatura ou produzidas pela indústria, (iv) se os participantes estão satisfeitos com as ferramentas existentes, e (v) se e por que consideram a execução de testes de regressão como uma atividade relevante.

4.2.3 Estudo Piloto

A fase inicial do estudo contemplou a aplicação de um estudo-piloto. Para esse estudo, contamos com a participação de dois profissionais da indústria e um profissional que atua na academia (pesquisador). Todos possuíam experiência prévia com testes de aplicativos para Android. Para a versão piloto do questionário foi adicionada uma seção de avaliação na qual os participantes poderiam prover *feedbacks* sobre a estrutura do questionário, bem como indicar sugestões de melhoria. A seção de avaliação apresentou questões relacionadas à adequação das questões em relação ao tema da pesquisa, o número de perguntas a serem respondidas, o tempo necessário para responder o formulário e a relevância da pesquisa, como segue:

1. Em uma escala de 1 a 5, onde 1 é igual a ruim e 5 é igual a ótimo, como você avalia o número de perguntas no formulário?
2. Em uma escala de 1 a 5, onde 1 é igual a ruim e 5 é igual a ótimo, como você avalia a adequação das perguntas do formulário ao tema “Processo de teste de aplicativos para Android”?
3. Cite pontos para melhorias.
4. Cite pontos positivos do questionário.
5. Gostaria de prover comentários adicionais sobre o questionário?

O *feedback* sugeriu pequenas modificações na estrutura do questionário. As mudanças incluíram: (i) adicionar opções de resposta a várias perguntas, (ii) alterar palavras para melhorar o entendimento e (iii) destacar o objetivo da pergunta para diferenciar questões que têm contextos semelhantes. Com base nas melhorias identificadas foi gerada uma segunda versão do questionário, que foi utilizada no estudo principal.

4.2.4 Distribuição do Questionário

O questionário foi elaborado através da ferramenta online Google Forms¹. O processo de ampla divulgação desta pesquisa foi realizado em 20 de janeiro de 2020 e ficou disponível por aproximadamente dois meses, durante o primeiro semestre de 2020. O link do questionário foi enviado por e-mail (a detinatários individuais, e através de listas de e-mail) e mídias sociais, tais como o LinkedIn. O formulário foi enviado juntamente com um texto de convite, que apresentava informações essenciais sobre o objetivo e a importância do estudo. Os participantes também foram informados sobre as políticas de privacidade do estudo de maneira clara e detalhada.

¹<http://docs.google.com/forms>

4.3 RESULTADOS

O *survey* recebeu 106 respostas, sendo 6 descartadas pois os participantes não tinham experiência anterior com teste de software nem trabalhavam com projetos de aplicativos Android, ou por terem respondido o formulário de forma duplicada. Assim, obtivemos um conjunto final de 100 respostas válidas. A Figura 4.1 apresenta a distribuição geográfica dos respondentes.

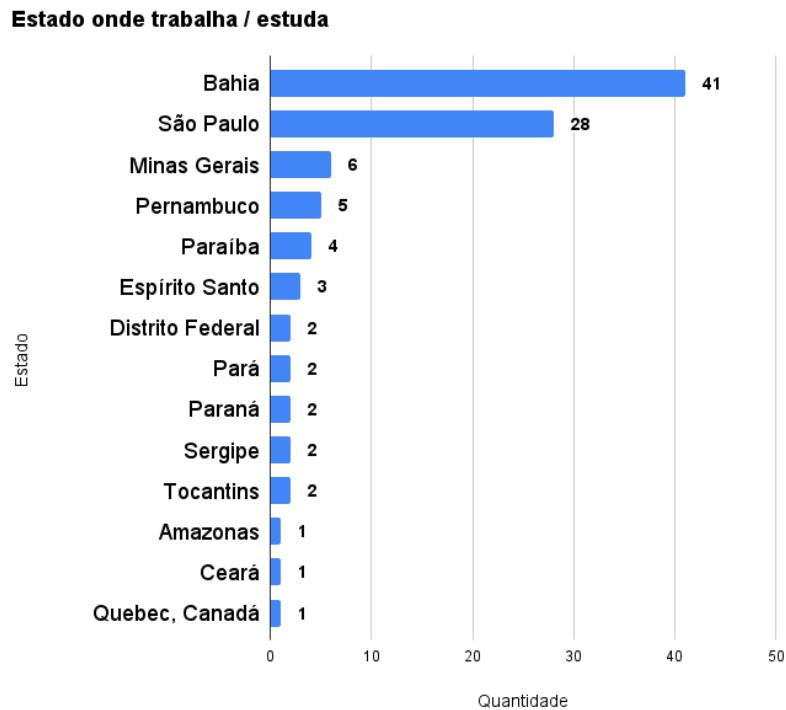


Figura 4.1 Distribuição geográfica dos participantes do survey.

As Tabelas B.1, B.2 e B.3 apresentam os dados dos participantes desse estudo.

4.3.1 Perfil dos participantes

Quanto ao gênero dos participantes, 81% eram homens e 19% mulheres. Sobre a idade dos participantes, 66% estavam na janela entre 20 e 30 anos (29% de 20 a 25 e 37% de 26 a 30 anos), 22% na janela entre 31 e 35 anos, e 12% possuíam mais de 35 anos, como apresenta a Figura 4.2.

Quanto ao nível de escolaridade, 9% possuíam apenas o ensino médio completo, 5% possuíam nível técnico, 7% eram estudantes de graduação, 55% eram graduados, 1% era estudante de pós-graduação e 23% possuíam diplomas mais altos, como MBA, mestrado ou doutorado, como apresenta a Figura 4.3.

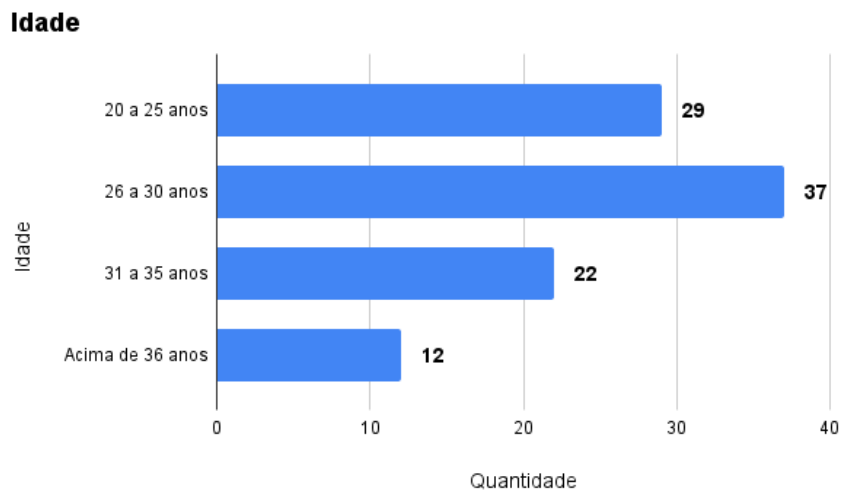


Figura 4.2 Distribuição da idade dos participantes do survey.

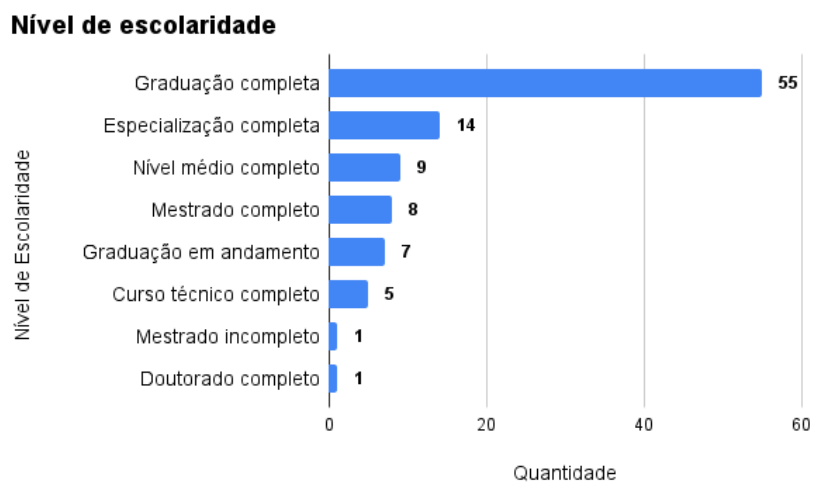


Figura 4.3 Distribuição do nível de escolaridade dos participantes do survey.

Dentre os participantes, 89% possuíam formação em Informática, Computação ou áreas afins. Essa formação refere-se a Nível Técnico, Graduação, Especialização, Mestrado e Doutorado. 9% eram formados em Engenharia Elétrica ou Engenharia de Produção e 2% possuíam diplomas em outras áreas, vide Figura 4.4.

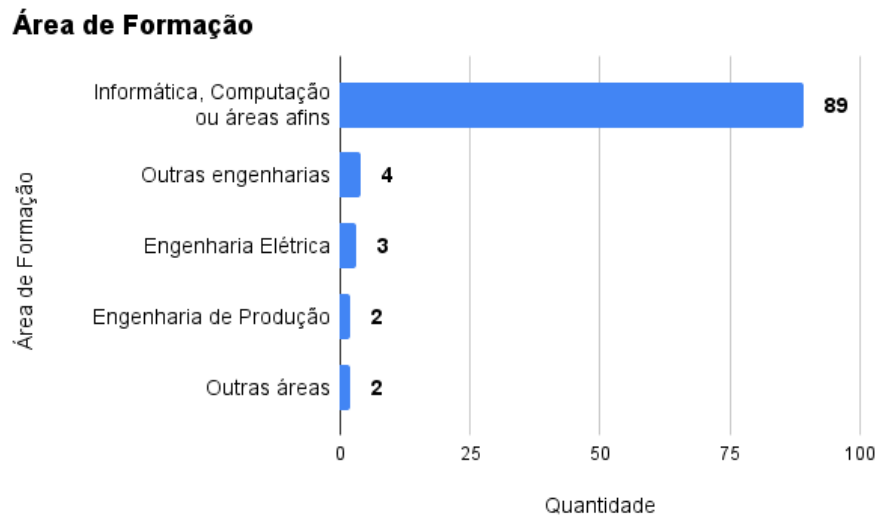


Figura 4.4 Distribuição da área de formação acadêmica dos participantes do survey

Ao considerar certificações ou cursos na área de teste de software, 56 dos 100 participantes possuíam pelo menos um certificado em cursos de teste de software, como o CTFL (*Certified Tester Foundation Level*) e o *Certified in Tester Foundation Level - Agile Tester (CTFL-AL)*. As Figuras 4.5 e 4.6 apresentam a distribuição de participantes por certificação/curso na área de testes de software.

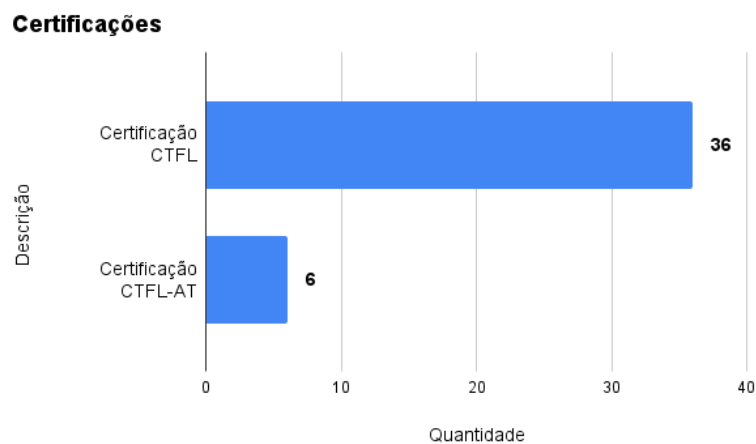


Figura 4.5 Distribuição de participantes por tipo de certificação na área de testes de software.

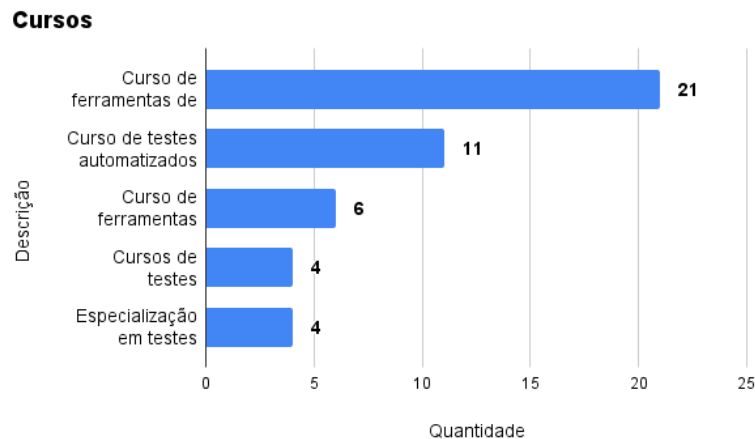


Figura 4.6 Distribuição de participantes por tipo de curso na área de testes de software.

Ao questioná-los sobre a atuação profissional, 81% dos participantes afirmaram ser estudantes e/ou profissionais que atuavam em uma empresa, 21% eram estudantes em dedicação integral, e 18% eram profissionais autônomos, como apresenta a Figura 4.7.

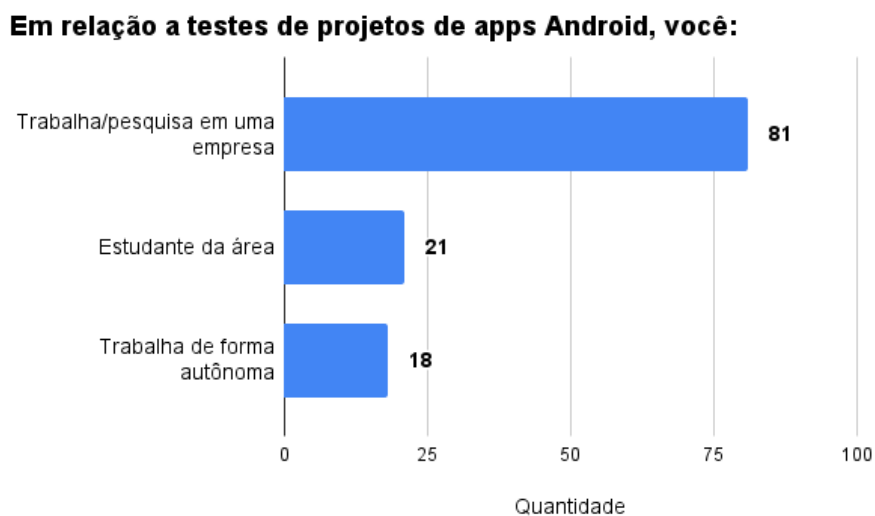


Figura 4.7 Distribuição de participantes por área de atuação profissional.

Em termos de experiência profissional em teste de software, a maioria dos participantes possuía até 5 anos de experiência, e apenas 7% possuía mais de 10 anos de experiência, como apresenta a Figura 4.8. No entanto, quando questionados sobre a experiência em testes de aplicativos Android, cerca de metade dos participantes (48%) possuía apenas 1 ano de experiência, como apresenta a Figura 4.9.

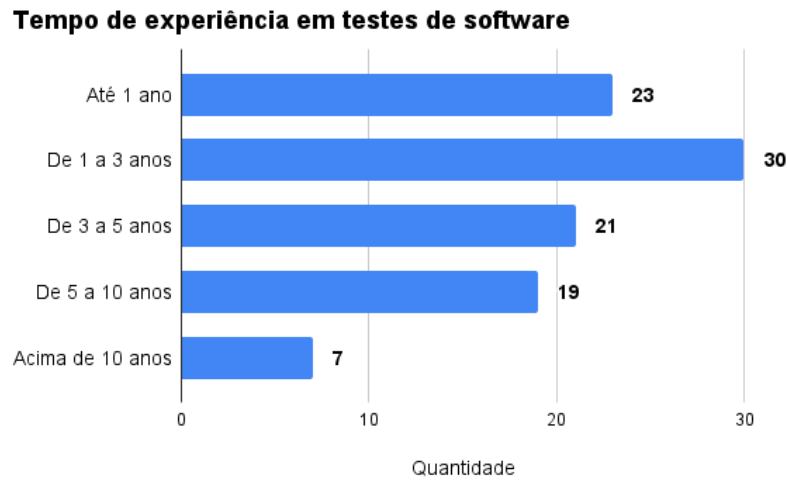


Figura 4.8 Distribuição de participantes por tempo de experiência profissional na área de testes de software.

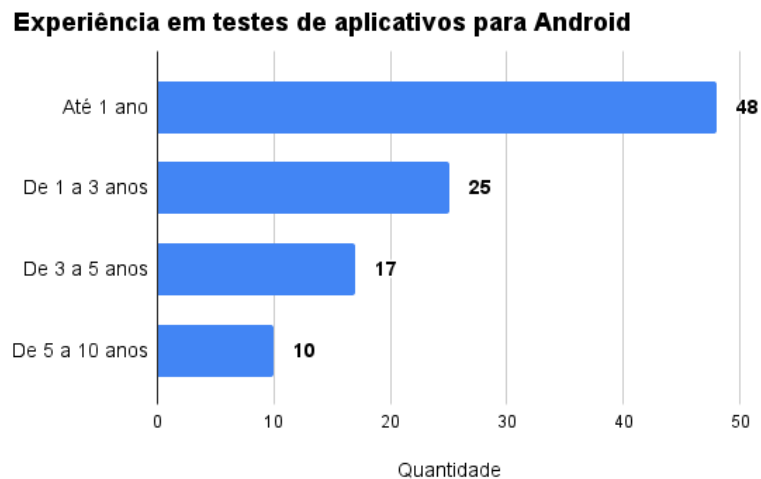


Figura 4.9 Distribuição de participantes por tempo de experiência profissional na área de testes para Android.

Quando solicitados a proverem uma auto-avaliação sobre o nível de conhecimento em testes para Android em comparação a outros profissionais, cerca de metade dos participantes indicou possuir nível bom ou excelente, e 29% indicaram possuir nível baixo ou muito baixo, como apresenta a Figura 4.10.

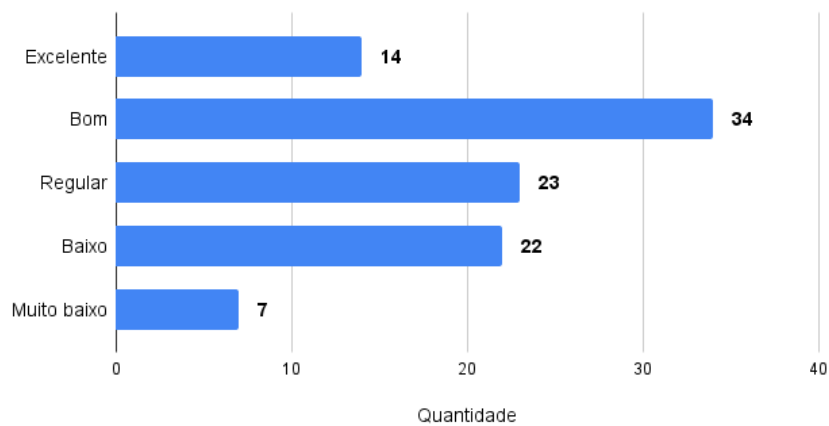
Nível de conhecimento do participante em testes de aplicativos Android

Figura 4.10 Auto-avaliação dos participantes a respeito do nível de conhecimento em testes para Android comparados a outros profissionais da área.

Em relação à posição principal do projeto atual na empresa, os participantes consideraram uma das seguintes descrições: analista de qualidade, desenvolvedor, analista de teste, desenvolvimento móvel com Android, desenvolvedor para Android, analista de sistemas, gerente de projeto, gerente técnico e engenheiro de teste.

Com relação às principais tarefas exercidas nos projetos atuais, 72% dos participantes trabalham com criação e design de caso de teste, e 85% deles trabalham apenas com execução de casos de teste, como apresenta a Figura 4.11.

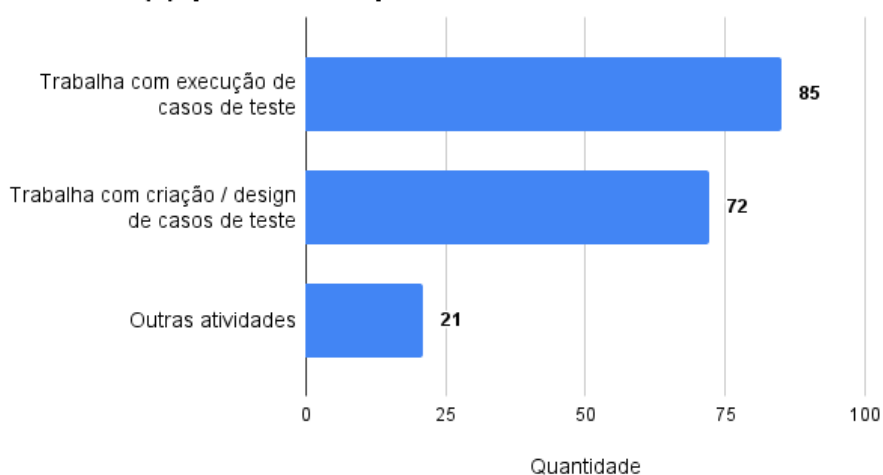
Atividade(s) que realiza no processo de teste de software

Figura 4.11 Distribuição de participantes por atividade realizada no processo de teste de software.

Quando perguntados sobre as técnicas de teste utilizadas, 77% responderam que faziam testes funcionais, 44% testes não-funcionais, e 27% *grey-box*, quando há acesso parcial ao código fonte, como apresenta a Figura 4.12.

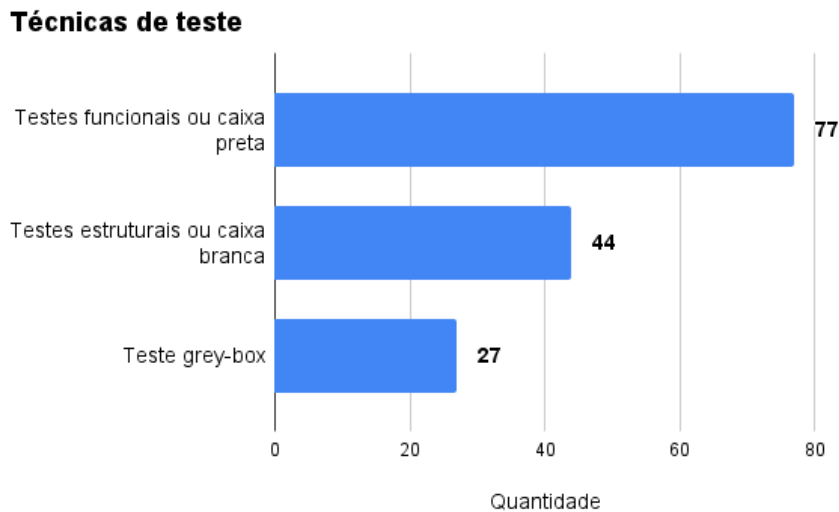


Figura 4.12 Distribuição de participantes por técnica de teste utilizada.

A respeito dos tipos de testes realizados, 60 participantes realizam teste de validação, 57 teste de sistema, 56 teste de regressão, 56 teste de integração, 52 teste de GUI, 44 teste de unidade, 32 teste de estresse, 30 teste de desempenho, 25 teste de conectividade, 15 teste de segurança, 9 teste de condições naturais, 7 teste de recuperação, 7 teste de proteção e 7 teste de certificação. A Figura 4.13 apresenta tal distribuição, em detalhes. Nesta questão, os participantes poderiam assinalar mais de uma opção.

4.3.2 Questões de Pesquisa

4.3.2.1 Quão automatizado é o teste de regressão de aplicativos Android na prática? (QP1)

Os participantes foram perguntados se eles costumavam executar testes de forma manual, automatizada ou se usavam as duas abordagens. 52% dos participantes indicaram que adotavam testes automatizados e manuais. 30% dos participantes responderam que realizam apenas testes manuais e 18% seguiam um processo de teste apoiado por ferramentas, como apresenta a Figura 4.14.

Tipos de teste

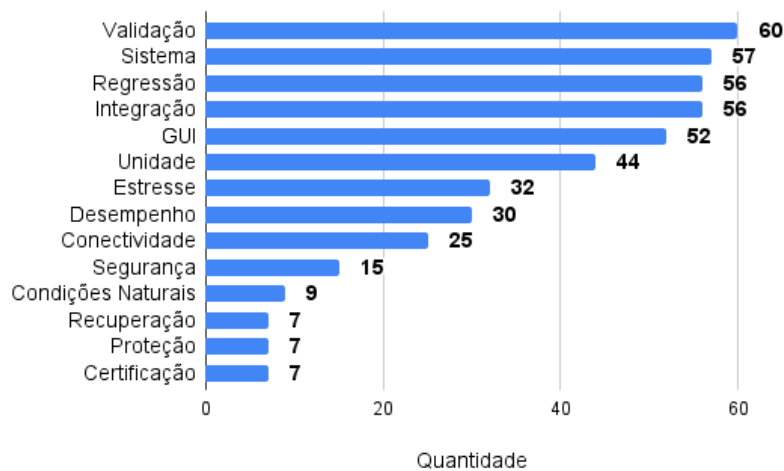


Figura 4.13 Distribuição de tipos de teste realizados pelos participantes do survey.

Os testes são realizados de forma manual e/ou automatizada

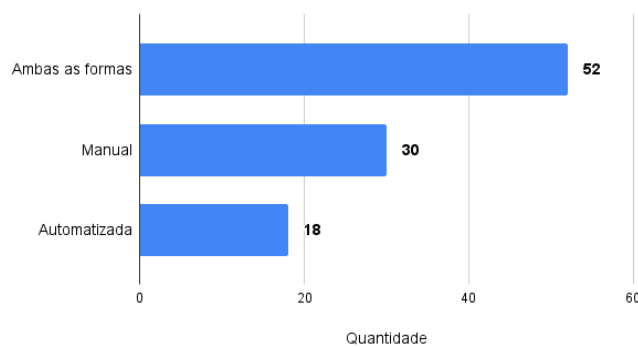


Figura 4.14 Survey: Os testes são realizados de forma manual e/ou automatizada

Em outra questão, referente ao uso de ferramentas, 35% dos participantes informaram que não usavam ferramentas para testar seus aplicativos. No entanto, essa resposta contrasta com a anterior quando 30% afirmaram que só realizavam testes manualmente. Em relação às ferramentas usadas com frequência, as mais mencionadas foram Appium, Espresso, UI Automator, Robolectric e Monkey. A Figura 4.15 apresenta a distribuição de participantes por ferramenta indicada como resposta no formulário.

Quando questionados se o processo de teste durante a manutenção (atualização do aplicativo) era feita de forma manual ou automatizada, 46% informaram que executavam teste de regressão de forma manual e automatizada, 38% responderam que testavam somente de forma manual, 15% responderam que testavam somente de forma automatizada e 1% responderam que não testavam o aplicativo após manutenção. A Figura 4.16 apresenta a distribuição de participantes por estratégia de teste regressão adotada na prática.

Ferramentas para automação de testes

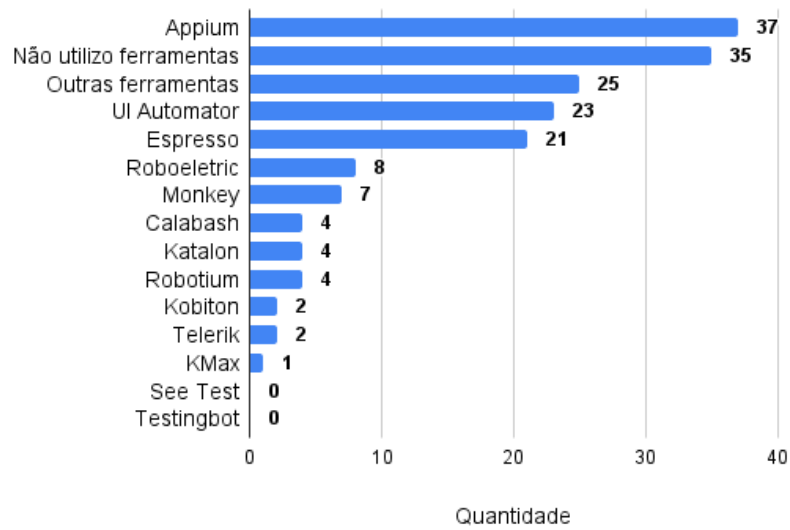


Figura 4.15 Distribuição de participantes por ferramentas.

Os testes de regressão são realizados de forma manual e/ou automatizada

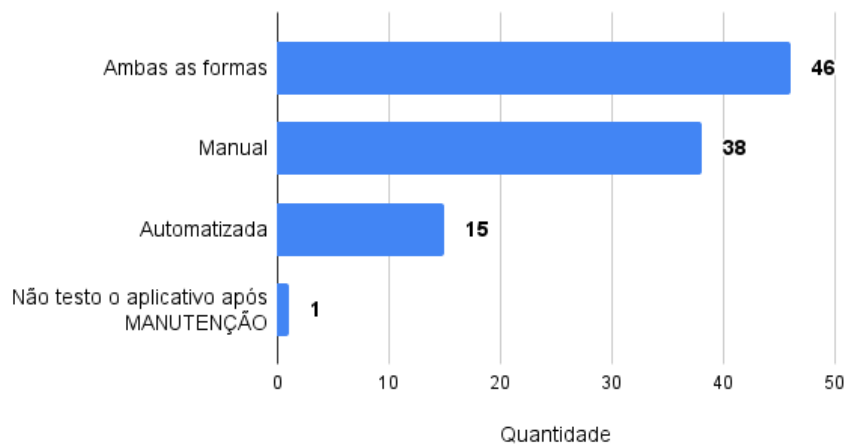


Figura 4.16 Distribuição de participantes por estratégia de teste de regressão.

No entanto, ao questioná-los sobre o uso de ferramentas de automação de testes ao realizar atividades de manutenção, 67% responderam que não utilizavam ferramentas. Dos 33 participantes que informaram que utilizavam ferramentas para automatizar o teste de novas versões, 33% dos participantes afirmaram que automatizam o teste de regressão com as mesmas ferramentas que utilizam durante as outras etapas de teste. Outros 27% informaram que utilizavam ferramentas propostas pela literatura (ATOM, REDROID e GUIDIFF), 24% informaram que utilizavam ferramentas disponíveis na

indústria (MONKEYRUNNER, TEST COMPLETE, UFT e SQUISH) e 18% informaram que utilizavam outras ferramentas como TestNG² e ferramentas internas, ou seja, aquelas desenvolvidas por sua própria empresa. A Figura 4.17 apresenta essa distribuição.

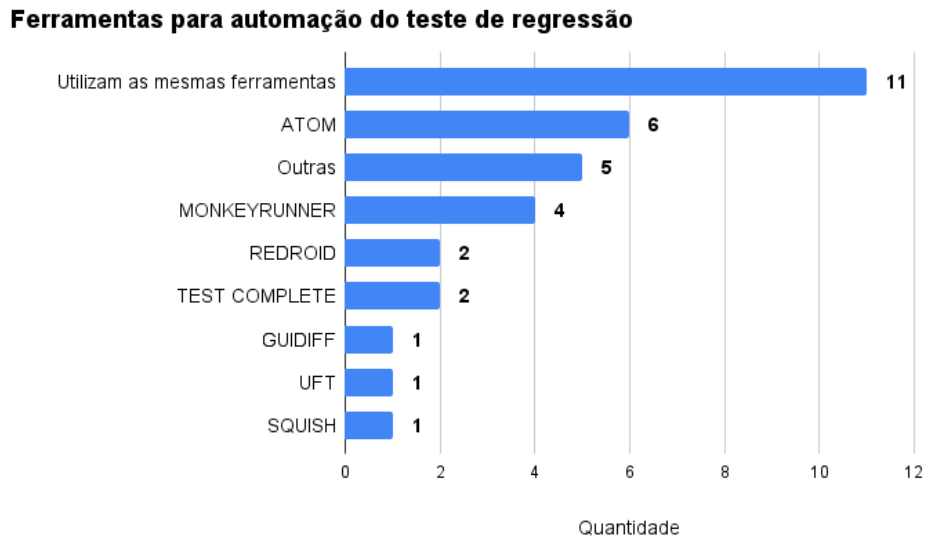


Figura 4.17 Distribuição de ferramentas utilizadas para automatizar o teste de regressão

Os participantes foram questionados como eles poderiam avaliar a relevância do suporte de ferramentas existente para testes de regressão, no sentido de que eles poderiam indicar se as ferramentas atendiam às suas necessidades específicas ou não. Para 10% dos participantes as ferramentas nunca atendem às suas necessidades. Para 8% elas raramente atendem às suas necessidades. Para 13% as ferramentas às vezes atendem às suas necessidades. Para 24% eles geralmente atendem às suas necessidades. 18% dos participantes responderam que as ferramentas sempre atendem às suas necessidades e 27% dos participantes não testam o aplicativo após a manutenção. A Figura 4.18 apresenta essa distribuição.

²<https://testng.org/doc/>

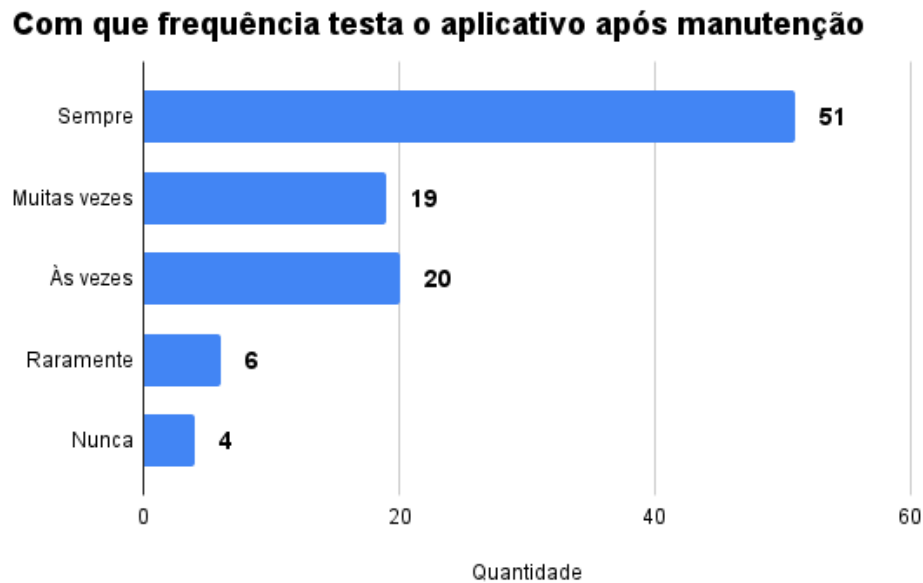


Figura 4.18 Distribuição de dados sobre como os participantes consideram as ferramentas disponíveis

Quando questionados se estavam satisfeitos com as funcionalidades oferecidas pelas ferramentas de testes de aplicativos para Android existentes 58% dos participantes responderam que sim e 42% responderam que não.

4.3.2.2 Como é realizado o processo de teste durante e após a manutenção dos aplicativos? (QP2)

Os participantes foram questionados sobre a prática de realizar testes nas fases de manutenção dos aplicativos, visando garantir que as manutenções realizadas não alteraram o comportamento funcional do produto.

Para 51% dos participantes, é “sempre” necessário testar novamente os aplicativos Android após a manutenção ou atualização para garantir que a manutenção realizada não altere o comportamento funcional do aplicativo. Para os outros participantes, 20% responderam que “às vezes” testam, 19% “frequentemente”, 6% “raramente” e apenas 4% “nunca”, como apresenta a Figura 4.19.

Os participantes também foram questionados sobre os procedimentos de teste que geralmente executavam após a manutenção. Esta pergunta buscava compreender se os participantes utilizavam estratégias para reduzir a quantidade de casos de teste a serem reexecutados para testar uma nova versão do aplicativo. A maioria dos participantes (77%) seleciona e reutiliza os casos de teste que se referem às alterações entre a versão original e a versão atualizada do aplicativo Android. Essa estratégia pode reduzir o tempo necessário para testar o aplicativo. No entanto, 59% responderam que reexecutam todos os casos de teste da versão original para testar a versão atualizada do aplicativo Android. 58% dos participantes analisam se é necessário criar novos casos de teste para

Com que frequência testa o aplicativo após manutenção

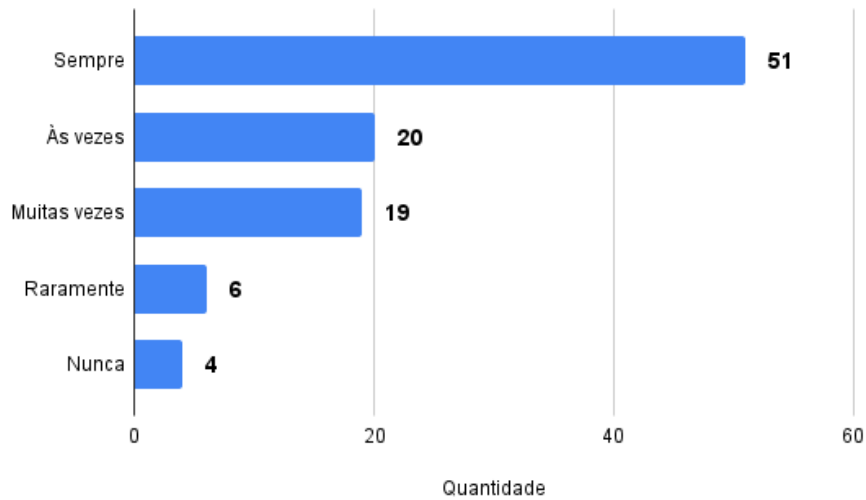


Figura 4.19 Distribuição da frequência de realização de testes durante a fase de manutenção.

a versão atualizada do aplicativo. 10% deles excluem um conjunto de casos de teste que não detectam falhas na versão atualizada do aplicativo Android, vide Figura 4.20. Vale ressaltar que nesta pergunta em particular, os participantes podiam selecionar mais de uma opção.

Estratégias referentes aos casos de teste a serem reexecutados para testar uma nova versão do aplicativo

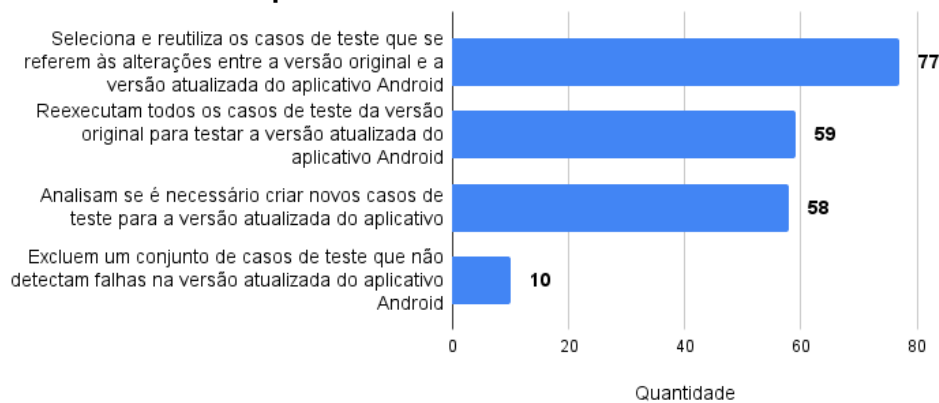


Figura 4.20 Distribuição de dados sobre os procedimentos adotados pelos participantes para testar a versão atualizada dos aplicativos Android.

4.3.2.3 Quais são os motivos para não realizar testes de regressão após a manutenção de aplicativos para Android? (QP3)

Embora 11% dos participantes indicaram não realizar testes após a manutenção ou atualização de software, 99% consideraram essa tarefa relevante.

Os participantes foram questionados sobre os principais motivos pelos quais as empresas não testam aplicativos após a manutenção. Para 71%, o motivo é o tempo de entrega reduzido para o lançamento da nova *release*. 28% dos participantes responderam que o problema se deve ao alto esforço da equipe de teste. Para 23% a detecção de erros na nova versão tem pouca importância. Para 22% o motivo é a falta de um processo suportado por ferramentas. Esses dados estão sintetizados na Figura 4.21.



Figura 4.21 Distribuição de fatores que levam à não-realização de testes em aplicativos nas fases de manutenção.

Os participantes (P) também compartilharam suas opiniões, com declarações como:

- “Os clientes geralmente pressionam a equipe para lançar o produto no mercado para não perder o tempo de colocação no mercado” - P19.
- “Temos uma cultura no processo de desenvolvimento, que acreditamos que a fixação de uma área não afetará outra” - P20.

4.4 DISCUSSÃO

Esta seção apresenta as discussões desse estudo, com base nos resultados obtidos com a aplicação do survey. A análise leva consideração três principais aspectos: foco da atividade de teste, automação do processo de teste e entendimento sobre o processo de teste de regressão.

4.4.1 Foco da atividade de teste

Os resultados indicaram que os participantes realizam mais testes funcionais, nos quais não precisam acessar o código-fonte do aplicativo. O estudo mostrou que os funcionários executam principalmente testes de validação, testes de integração, testes de sistema, testes de regressão, testes de GUI e testes de unidade. Esses testes podem ser essenciais para garantir a qualidade do aplicativo. No contexto de aplicações para dispositivos móveis os testes de GUI tem um impacto direto na qualidade do aplicativo desenvolvido. Com relação à preferência por testes funcionais em vez de não funcionais, pode ser necessário atrair especialistas em tecnologias de software e código fonte específicos para realizar testes não funcionais. Além disso, ferramentas podem ser adotadas para facilitar a execução e criação do teste.

Pesquisas futuras poderão ser realizadas para entender: (i) a relação entre os tipos de testes (validação, unidade, GUI, sistema) e o quão são automatizados, (ii) as abordagens e estratégias usadas para realizar testes funcionais ou não-funcionais e (iii) quais razões levam os profissionais a escolher uma técnica de teste específica em detrimento de outra.

4.4.2 Automação do processo de teste

A automação dos testes não é executada na maioria das empresas de software pesquisadas, apenas 18% dos participantes realizam testes automatizados. No entanto, eles poderiam reduzir significativamente o custo do processo de teste, minimizar o erro humano e facilitar o teste de regressão (AMMANN; OFFUTT, 2008). Portanto, o desenvolvimento de ferramentas apropriadas pode trazer grandes benefícios para a área.

A maioria dos participantes faz uso de ferramentas gratuitas que supostamente não atendem totalmente às suas necessidades profissionais. As ferramentas acadêmicas podem ter poucos requisitos e normalmente não são disponíveis para uso comercial.

Uma pesquisa ou comparação da eficácia das ferramentas de teste pode ser realizada para orientar e auxiliar os testadores durante as atividades de teste. Neste estudo, também foi perguntado aos participantes sobre quais recursos poderiam ser aprimorados ou implementados nas ferramentas que eles estão usando atualmente.

As respostas mais relevantes dos participantes (P) foram:

- *“Os testes de UI no Android ainda são difíceis e demorados para construir.”* - P1.
- *“Acho que poderia ser emitido relatórios de tempo de execução de testes em relação a uma versão anterior e informar se teve algum caso de teste que não foi coberto na nova versão e que estava sendo coberto na versão anterior.”* - P2.
- *“A criação e execução das suítes de teste precisam ser automatizadas para permitir uma verificação mais rápida da qualidade da aplicação frente a alterações.”* - P18.
- *“Serem mais User Friendly.”* - P25.
- *‘Difícilmente uma ferramenta vai conseguir fornecer tudo que o projeto precisa em relação a testes, então não acho que seja uma feature/funcionalidade que venha a*

resolver todos os problemas. O que a equipe realmente precisa ter é uma pessoa que entenda de testes e que quando falta alguma coisa em alguma ferramenta pronta, consegue mobilizar a equipe para criar uma lib do zero para resolver aquele problema. Isto é bem melhor do que ficar importando várias ferramentas de teste para o projeto, porque naturalmente cada uma é melhor em um aspecto. Então chega um certo ponto, que ao invés de ficar experimentando ferramentas novas para ver se resolve um problema, é melhor a equipe desenvolver esta nova ferramenta.” - P29.

Tais respostas apresentam dificuldades encontradas pelos participantes no uso de ferramentas de automação de teste para aplicativos Android que representam alguns dos motivos pelos quais eles não costumam usar as ferramentas existentes. Estudos futuros podem ser feitos para mapear a relação entre automação e tipo de teste, identificando: (i) quais tipos de testes possuem uma necessidade maior de ferramental de apoio para sua execução e (ii) quais recursos são esperados por esses profissionais para cada tipo específico de teste.

4.4.3 Processo de teste de regressão

Embora 99% dos participantes considerem relevante realizar testes de regressão, a quantidade de profissionais que realmente os realiza é muito menor. A seleção dos casos de teste a serem reexecutados para testar novas versões dos aplicativos pode reduzir o custo do software e o esforço da equipe. A pesquisa não incluiu a palavra regressão em nenhuma pergunta ou resposta. Assim, alguns dos participantes podem não ter entendido que testar o aplicativo após executar sua atualização (manutenção) referia-se a testes de regressão (LEUNG; WHITE, 1989). Foram observadas divergências nas respostas dos participantes. Por exemplo, quando perguntados se eles realizavam algum tipo de teste após a manutenção do aplicativo para garantir que as alterações não afetem o comportamento funcional do aplicativo 4% responderam que nunca realizam esses testes. No entanto, em uma pergunta relacionada 12% responderam que não testam o aplicativo após a manutenção.

Embora a literatura apresente muitos estudos sobre técnicas de seleção de casos de teste como uma maneira de reduzir a quantidade de casos de teste no processo de teste de regressão (LEUNG; WHITE, 1989; LEUNG; WHITE, 1991; ROTHERMEL; HARROLD, 1996; ENGSTRÖM et al., 2010; KAZMI et al., 2017) muitos profissionais ainda reexecutam todos os casos de teste existentes para testar a nova versão. Como a maioria dos participantes não usa o suporte de ferramentas, o custo e o esforço para aplicar uma estratégia de seleção manualmente podem ser maiores do que reexecutar o conjunto completo de casos de teste. Estes resultados reforçam a necessidade de ferramentas que forneçam uma seleção adequada de casos de teste.

4.5 AMEAÇAS À VALIDADE

Validade interna. Uma importante ameaça à validade interna diz respeito ao possível viés na seleção dos participantes do estudo. O nosso objetivo inicial era o de mapear profissionais de teste com larga experiência industrial, de modo a nos prover dados rele-

vantes sobre a prática de teste. Entretanto, boa parte dos participantes possuía pouca experiência, ou mesmo não fazia uso de ferramentas de automação de testes. E isso pode ter limitado o nosso raio de percepção de *insights*.

Validade externa. Embora este estudo seja baseado em uma amostra que considerou pesquisadores e profissionais de teste de software, os resultados não podem ser generalizados para outros cenários, dado o pequeno quantitativo de participantes. No entanto, este estudo é um passo inicial para fornecer mais evidências sobre a consciência e percepção dos profissionais sobre o uso de teste de regressão na manutenção de aplicativos Android.

Validade de construto. No questionário, evitou-se usar o termo teste de regressão. Em vez disso, foi usado implicitamente a definição do teste de regressão como uma estratégia para evitar viés e impedir que os participantes pesquisassem o conceito, antes de responder ao questionário. Apesar dos esforços, alguns participantes podem não ter relacionado a definição ao teste de regressão, e essa é uma ameaça de construto bastante sensível ao presente estudo.

4.6 SÍNTESE DO CAPÍTULO

O presente capítulo apresentou o *Survey* utilizado como instrumento de pesquisa para compreender qual a perspectiva de acadêmicos e profissionais sobre técnicas de seleção de teste de regressão implementadas por ferramentas para projetos de aplicativos Android. O Apêndice A apresenta o questionário que foi disponibilizado para preenchimento dos participantes. No próximo capítulo serão apresentadas as entrevistas que foram realizadas com profissionais que atuam no contexto de desenvolvimento mobile.

No próximo capítulo será apresentado o estudo empírico realizado com profissionais e pesquisadores que atuam em projetos de aplicativos para Android, utilizando a metodologia de entrevistas semi-estruturadas.

5

ENTREVISTAS

Com o objetivo de compreender como os profissionais de Engenharia de Software, em particular testadores e desenvolvedores, lidam com teste de regressão nos projetos para a plataforma Android, e complementar os resultados obtidos com o Survey, realizamos um estudo empírico do tipo *entrevista* semi-estruturada com a participação de 16 profissionais que atuam em projetos de desenvolvimento de software para a plataforma Android. Este capítulo apresenta esse estudo empírico, com detalhes sobre o planejamento, a execução e análise dos resultados obtidos.

5.1 APRESENTAÇÃO E QUESTÕES DE PESQUISA

Segundo Hove e Anda (2005), o propósito de utilizar-se de entrevistas semi-estruturadas é coletar dados sobre fenômenos que não podem ser obtidos por meio de medidas quantitativas. Para a presente investigação, foram definidas as seguintes Questões de Pesquisa (QP):

- QP1: Qual a relação entre o nível de conhecimento do profissional e a forma como ele realiza teste de regressão em aplicativos para Android?** Deseja-se identificar se a trajetória e o nível de conhecimento do profissional em testes de aplicativos e teste de regressão impactam na forma como ele realiza o processo de teste de regressão na sua prática.
- QP2: Quais são as linguagens de programação utilizadas no desenvolvimento de aplicativos para Android e qual a relação entre linguagens e automação dos testes?** O desenvolvimento de aplicativos para Android pode ser feito utilizando-se algumas linguagens de programação que são categorizadas como linguagens nativas, quando propostas pela empresa que fornece o Android e não-nativas ou híbridas, que são linguagens utilizadas que permitem o desenvolvimento de aplicativos para multiplataformas. Esta QP busca elencar as linguagens utilizadas no desenvolvimento dos aplicativos e identificar

se na percepção dos participantes existe relação entre a linguagem utilizada no desenvolvimento do aplicativo e a automatização dos testes, de acordo com o ferramental de apoio disponível.

QP3: Quais são as ferramentas utilizadas para automação dos testes e quais são as características do ferramental de apoio que os profissionais necessitam para automatizar os testes? Tanto a literatura quanto a indústria produzem ferramentas para automação de testes de aplicativos para Android. A Tabela 4.2 elencou algumas dessas ferramentas. Esta QP busca identificar as ferramentas utilizadas por profissionais, bem como identificar as funcionalidades que os profissionais esperam encontrar em ferramentas de automação de testes.

QP4: Como os profissionais executam teste de regressão nos projetos em que atuam? A literatura apresenta diversos estudos sobre teste de regressão e suas técnicas para redução de casos de testes. Esta QP busca (i) elencar o passo-a-passo no planejamento e execução dos testes em cenário de evolução dos aplicativos realizados pelos profissionais, (ii) compreender se na percepção do profissional é importante sistematizar o teste de regressão e (iii) identificar se para os profissionais a execução do teste de regressão impacta na qualidade da *release* do aplicativo.

5.2 METODOLOGIA DA PESQUISA

Esta seção abrange os detalhes do planejamento, procedimentos de execução, resultados e discussões. Foram aplicados os princípios de pesquisa definidos por Hove e Anda (2005), como apresenta a Figura 5.1.



Figura 5.1 Entrevistas: Etapas Metodologia da Pesquisa

5.2.1 Identificação do Público Alvo

Foi definido como público-alvo deste estudo profissionais que atendessem dois critérios:

- Trabalhar com testes de aplicativos para Android;
- Possuir no mínimo um ano de experiência na área de desenvolvimento/teste de aplicativos.

5.2.2 Definição do Protocolo

Foi definido o protocolo a ser seguido neste estudo conforme apresentado no Apêndice C. O objetivo do protocolo é estabelecer orientações e processos a serem seguidos, antes, durante e depois das entrevistas.

O Apêndice D apresenta o roteiro da entrevista que possui 11 perguntas de caráter subjetivo. Cada pergunta do roteiro tinha como objetivo prover dados para responder as questões de pesquisa, conforme demonstrado na Tabela 5.3, exceto a pergunta 11 que tinha como objetivo obter o *feedback* dos participantes referente a percepção deles sobre a entrevista.

Tabela 5.1 Entrevista: Roteiro X Questões de Pesquisa

Pergunta do Roteiro	Questão de Pesquisa (QP)
Pergunta 1	QP 1
Pergunta 2	QP 1
Pergunta 3	QP 2
Pergunta 4	QP 4
Pergunta 5	QP 4
Pergunta 6	QP 1
Pergunta 7	QP 3
Pergunta 8	QP 2
Pergunta 9	QP 4
Pergunta 10	QP 4

5.2.3 Estudo Piloto

Com o objetivo de validar se as questões a serem respondidas pelos participantes, conforme apresentado no Apêndice D, responderiam às questões de pesquisa definida neste estudo, foi realizado um estudo piloto com um profissional que atua com desenvolvimento e testes de aplicativos para Android e possui mais de cinco de anos de experiência na área de testes de software. A entrevista durou cerca de 40 minutos. A aplicação do piloto foi essencial para calibrar o instrumento.

5.2.4 Seleção dos Participantes

Neste estudo buscou-se por profissionais que atuassem em empresas onde o processo de teste fosse sistematizado e o aplicativo fosse um produto essencial da empresa. Assim, para encontrar potenciais participantes foram utilizadas as seguintes estratégias:

1. Foram contatados por e-mail 13 participantes do *survey* realizado que preenchiam o perfil profissional que definimos como público-alvo;
2. A pesquisa foi divulgada em um grupo fechado de Automação e *Networking* do WhatsApp que possui 257 participantes;
3. A pesquisa foi divulgada em um grupo fechado do LinkedIn *Android Brasil* que possui 7.721 integrantes;
4. Foram contatados por mensagem direta 104 profissionais pelo LinkedIn que preenchiam o perfil profissional definido como público-alvo;
5. Foram solicitadas indicações de potenciais participantes para os profissionais que participavam da entrevista.

Tabela 5.2 Entrevistas: Estratégias utilizadas para seleção de participantes

Tipo de estratégia	Participantes alcançados
Linkedin - Mala Direta	13
Grupo de WhatsApp de Automação e <i>Networking</i> - Mala Direta	2
Indicação de outros participantes	1

O grupo de WhatsApp e a comunidade do LinkedIn, nos quais divulgamos nossa pesquisa, são grupos fechados, e nem todos os participantes são profissionais que atuam diretamente com desenvolvimento e teste de Android. Assim, não foi possível estabelecer um percentual de pessoas alcançadas por cada estratégia que utilizamos para selecionar os participantes deste estudo.

O processo de agendamento das entrevistas foi realizado por e-mail, WhatsApp ou pelo *direct* do LinkedIn.

5.2.5 Condução das Entrevistas

O processo de condução das entrevistas foi realizado utilizando o Roteiro de Entrevista apresentado no Apêndice D. Durante as entrevistas, as perguntas podiam ser explicadas ou refeitas ao participante de modo que ficasse compreensível. Não foram acrescentadas perguntas que não estavam no roteiro.

Inicialmente foram fornecidas aos participantes informações e orientações relevantes sobre a entrevista. As entrevistas foram realizadas utilizando a plataforma Google Meet¹. A duração média das entrevistas foi de 22 minutos, com desvio padrão de 8 minutos.

Com o objetivo de obter as informações necessárias e ao mesmo tempo ter a atenção focada durante o diálogo com os participantes, as entrevistas foram gravadas mediante autorização prévia feita por um formulário online utilizando a plataforma Google Forms², conforme demonstrado no Apêndice E.

¹<https://meet.google.com>

²<http://docs.google.com/forms>

5.2.6 Transcrição das Entrevistas

As transcrições das entrevistas foram realizadas em duas etapas. Na primeira foi utilizada a ferramenta *open-source* Transcriber Bot do Telegram³, na qual os áudios eram inseridos e a saída era a transcrição em texto das entrevistas. Tanto a literatura cinza^{4,5}, quanto a literatura acadêmica (MILLAR et al., 2005) apresentam estudos referentes a métricas utilizadas para calcular a acurácia de ferramentas que trabalham com transcrição de arquivos de áudio em texto, tais como, o percentual de palavras erradas, ou de sentenças erradas. Alguns desafios referente a essas métricas, referem-se a qualidade do áudio e a nasalidade da fala. Não encontramos trabalhos que apresentam a acurácia da ferramenta que utilizamos Transcriber Bot.

Na segunda etapa foi feito um refinamento das transcrições geradas automaticamente pela ferramenta, quando foi comparado o áudio com o texto e corrigido os erros encontrados.

5.2.7 Codificação

O tipo realizado de codificação foi aberta (HODA, 2021), sem uso de ferramentas, identificando as palavras-chave nas respostas dos participantes de acordo com cada pergunta no formulário. Também foi utilizado como critério agrupar as respostas em busca de fornecer dados para responder as questões de pesquisa deste estudo, conforme apresentado na Tabela 5.3.

³https://telegram.me/Transcriber_Bot

⁴<https://www.3playmedia.com/blog/how-accurate-is-your-transcription-subtitling-service/>

⁵<https://medium.com/descript/challenges-in-measuring-automatic-transcription-accuracy-f322bf5994f>

Tabela 5.3 Entrevistas: Critérios utilizados para codificação das respostas

#	Critérios utilizados para codificação
Pergunta 1	Identificar padrões encontrados na trajetória profissional do participante.
Pergunta 2	Identificar as fontes de aprendizado sobre teste de aplicativos para Android do participante.
Pergunta 3	Identificar quais são as linguagens de desenvolvimento utilizadas nos projetos (linguagens nativas x híbridas) e elencar quais são as ferramentas utilizadas para automação dos testes.
Pergunta 4	Identificar se existe um processo definido para testar as novas <i>releases</i> dos aplicativos e elencar quais são as etapas realizadas pelos participantes para testar as novas <i>releases</i> .
Pergunta 5	Identificar se o participante utiliza algum critério para reduzir o número de casos de teste a serem reexecutados no processo de teste das novas <i>releases</i> .
Pergunta 6	Compreender se o participante sabe o que é teste de regressão, elencar quais foram as fontes de aprendizagem e identificar se possuem alguma dúvida sobre esse tema.
Pergunta 7	Identificar requisitos funcionais e não-funcionais necessários para uma ferramenta de teste de regressão para aplicativos Android.
Pergunta 8	Identificar se na percepção dos participantes existe relação entre a linguagem utilizada no desenvolvimento do aplicativo e o processo de teste de regressão ser automatizado.
Pergunta 9	Identificar se na percepção dos participantes é importante sistematizar o processo de teste de regressão.
Pergunta 10	Identificar se na percepção do participante executar teste de regressão auxilia na qualidade da nova release do aplicativo.

5.3 RESULTADOS

5.3.1 Perfil dos Participantes

A Tabela F.1 apresenta uma síntese do perfil dos participantes deste estudo. Quanto ao gênero, 87% são homens e 12% são mulheres. Em relação a faixa etária, 50% dos participantes possuem entre 20 e 30 anos e os demais possuem entre 31 e 40 anos. 50% dos participantes trabalham no estado de São Paulo, os demais, encontram-se distribuídos entre os estados do Amazonas, Goiás, Minas Gerais, Rio de Janeiro e o Distrito Federal.

Mais de 80% dos participantes são graduados em cursos da área de Informática. O nível máximo de formação (coluna "Educação" na Tabela F.1) dos participantes está dividido prioritariamente entre graduados e especialistas (ESP). 43% informaram que possuem cursos ou certificações na área de teste. Apesar de mais de 80% dos participantes terem formação na área de Informática, este estudo também apontou profissionais de outras áreas que atuam hoje na área de teste de aplicativos para dispositivos móveis, seja como analista de teste, profissional da área de qualidade de software - atuação na área de *Quality Assurance* (QA) -, ou como desenvolvedor. Foi identificado que nos processos seletivos para contratação, a qualificação do profissional e sua performance prática em relação as atividades desenvolvidas são critérios valorizados pelos empregadores. Algumas empresas possuem um plano de formação próprio para capacitar seus profissionais.

Quanto à experiência profissional em teste de aplicativos para Android (em anos), 75% possuem entre 1 a 3 anos de experiência, 12% possuem entre 4 e 5 anos e 12% possuem menos de 1 ano. Embora tenha sido estabelecido como critério inicial o participante ter mais de um ano de experiência na área de testes de aplicativos para Android, os dados obtidos nestas entrevistas foram considerados relevantes, por isso não foram descartados. Mais de 65% dos participantes trabalham em empresas que possuem mais de 100 funcionários. Quanto ao tempo que atuam na empresa, 43% possuem menos de 1 ano, 50% entre 1 a 3 anos e 6% entre 4 e 5 anos. Quanto a área de atuação, mais de 60% trabalham diretamente na área de testes de software, enquanto os demais atuam na área de desenvolvimento.

Em relação a experiência profissional em teste de aplicativos para Android, há uma concentração na faixa de 1 a 3 anos de experiência. Esses dados corroboram com os resultados obtidos com o *Survey*, com uma leve indicação de que esta é uma área em expansão na indústria de software brasileira. Foi observado também que, embora haja uma diversidade de cargos, na prática os profissionais executam atividades similares.

5.3.2 Questões de Pesquisa

5.3.2.1 Qual a relação entre o nível de conhecimento do profissional e a forma como ele realiza teste de regressão em aplicativos para Android? (QP1)

Dos 16 participantes apenas um iniciou a carreira na área de teste web e mobile. 5 participantes trabalham com desenvolvimento e executam testes, com foco, nos testes de unidade. 9 participantes iniciaram em outras áreas e migraram para a área de qualidade, com foco em automação nos domínios web e mobile. Algumas dessas migrações na trajetória profissional ocorreram por oportunidades que surgiram na empresa em que já trabalhavam.

Foi perguntado aos participantes quais as principais fontes de aprendizado sobre teste de aplicativos para Android. Nesta pergunta os participantes informaram mais de uma categoria. A Figura 5.2 apresenta a síntese das fontes de aprendizado indicadas pelos participantes. Os resultados estão descritos a seguir:

- 68% dos participantes indicaram como fonte de aprendizado os colegas de trabalho, no processo de troca de experiência, em especial, com os que já possuíam mais conhecimento e prática profissional;
- 56% dos participantes citaram como fonte de aprendizado a pesquisa na internet (e.g., sites, fóruns, artigos, vídeos de plataformas como o YouTube);
- 25% dos participantes responderam que tem como fonte de aprendizado os cursos livres;
- 25% dos participantes responderam que participaram de planos de formação da empresa onde atuam ou atuaram;
- 12% dos participantes responderam que tem como fonte de estudos as comunidades da área de testes;

Fontes de Aprendizado sobre testes de aplicativos para Android

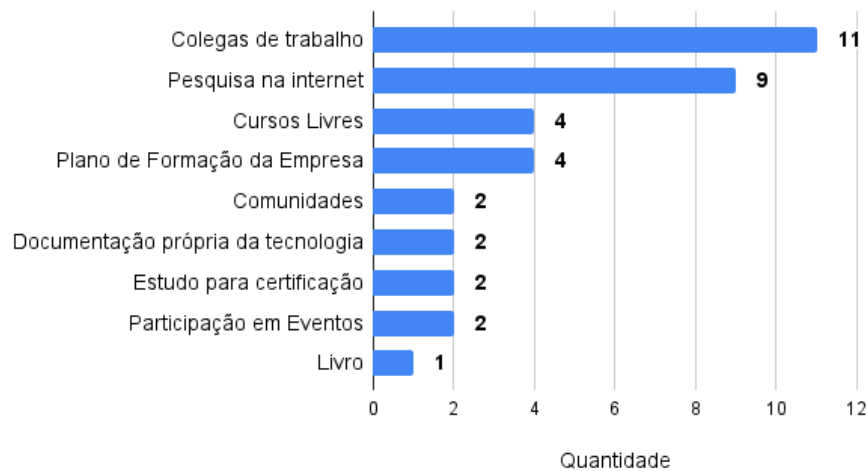


Figura 5.2 Entrevistas: Fontes de Aprendizado sobre teste de aplicativos para Android

- 12% dos participantes informaram que utilizam a documentação oficial da tecnologia utilizada;
- 12% dos participantes responderam que utilizam material de estudo para certificação;
- 12% dos participantes responderam que obtiveram como fonte de aprendizado as participação em eventos (e.g., congressos, palestras, *meetups*);
- 6% dos participantes adotaram os livros como fonte de estudo. Nenhum participante indicou a faculdade como fonte de aprendizado.

Também foi perguntado aos participantes sobre o conhecimento que eles têm sobre teste de regressão. O nosso objetivo era identificar se eles sabiam ou não o conceito, como conheceram o assunto e se tinham alguma dúvida sobre o tema. A Figura 5.3 apresenta a síntese das fontes de aprendizagem indicadas pelos participantes. Os resultados estão descritos a seguir:

- 93% dos participantes indicaram que sabem o que é teste de regressão;
- 60% dos participante apontaram que conheceram o assunto na prática profissional;
- 13% dos participantes relatam que aprenderam a definição na faculdade;
- 13% dos participantes afirmam que aprenderam sobre o conceito de teste de regressão através de leituras da literatura cinza;
- 6% dos participantes afirmam que conheceram sobre o conceito de teste de regressão participando em planos de formação da empresa; e

Fontes de aprendizado sobre Teste de Regressão

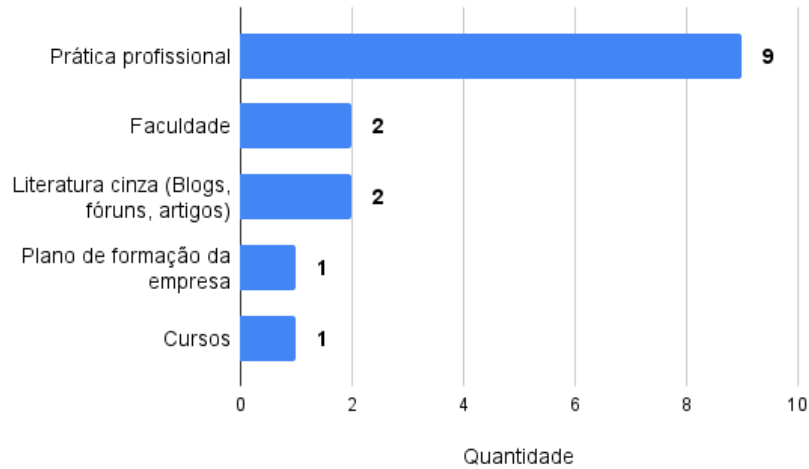


Figura 5.3 Entrevistas: Fontes de Aprendizado sobre teste de regressão

- 6% dos participantes afirmam que conheceram sobre o conceito em cursos.

Foi perguntado aos participantes se eles tinham alguma dúvida sobre teste de regressão. Dos 16 participantes, 67% participantes informaram que não possuíam dúvidas sobre o tema, 20% dos participantes não souberam responder quais seriam as dúvidas e 13% dos participantes optaram por não responder.

5.3.2.2 Quais são as linguagens de programação utilizadas no desenvolvimento de aplicativos para Android, e qual relação entre linguagens e automação dos testes? (QP2)

Esta QP possui dois objetivos: (i) identificar quais são as linguagens utilizadas no processo de desenvolvimento dos aplicativos para Android e (ii) compreender se na percepção do profissional existe uma relação entre a linguagem utilizada para desenvolver o aplicativo e o processo de automação dos testes, em função do ferramental de apoio disponível para aquela linguagem.

Foi perguntado aos participantes sobre as linguagens utilizadas no desenvolvimento dos aplicativos nos projetos em que eles atuavam. Nesta pergunta os participantes poderiam informar mais de uma linguagem. As linguagens de desenvolvimento foram divididas em duas categorias: nativas e não-nativas.

As linguagens nativas são definidas pela própria empresa que desenvolve o Android, são elas Java e Kotlin, sendo que esta última é a linguagem oficial atual para desenvolvimento Android. O desenvolvimento de aplicativos para multi-plataformas, ou seja, com um único código um aplicativo que roda tanto em Android quanto em iOS, normalmente é realizado utilizando *frameworks*, tais como: Flutter (linguagem Dart), React (biblioteca JavaScript) e React-native (biblioteca Javascript).

Quanto às linguagens nativas, 75% dos participantes indicaram utilizar Kotlin no de-

envolvimento dos aplicativos, e 43% dos participantes indicaram que os aplicativos são desenvolvidos em Java. É importante ressaltar que alguns participantes apontaram que utilizam majoritariamente a linguagem Kotlin para desenvolvimento e que utilizam a linguagem Java na manutenção dos sistemas legados, ou seja, produtos que embora não recebam novas implementações, precisam ser mantidos por possuírem dados e funcionalidades relevantes.

18% dos participantes apontaram que utilizam *frameworks* que produzem código multi-plataforma, sendo que 12% dos participantes apontaram que utilizam o *framework* React e 6% dos participantes apontaram que utilizam o *framework* React-Native.

Outra pergunta feita aos participantes referia-se a relação entre a linguagem que era adotada no desenvolvimento e o processo de automatização dos testes de regressão. Desejava-se compreender se na percepção dos participantes utilizar linguagens nativas, como Kotlin e Java, poderia favorecer a automação dos testes de regressão em virtude de possuir mais ferramentas disponíveis. Como resultados, identificamos que 43% dos participantes acreditavam não existir relação entre a linguagem de desenvolvimento e o processo de automação. 43% acreditavam que existe relação entre a linguagem de desenvolvimento e o processo de automação, visto que as linguagens nativas são mais “confiáveis” e mais “sólidas”, e conseqüentemente possuíam mais ferramentas para automatizar os testes. Outros 12% não responderam.

5.3.2.3 Quais são as ferramentas utilizadas para automação dos testes e quais são as características do ferramental de apoio que os profissionais necessitam para automatizar os testes? (QP3)

Esta QP tem como objetivos: (i) elencar quais são as ferramentas utilizadas pelos profissionais para automação de testes, em especial, teste de regressão, e (ii) identificar quais são os requisitos que estes profissionais compreendem como essenciais para uma ferramenta atender a demanda de executar teste de regressão. Foi perguntado aos participantes qual(is) ferramenta(s) eles utilizavam para automação dos testes. Os participantes poderiam indicar mais de uma ferramenta. A Figura 5.4 apresenta as ferramentas de automação utilizadas pelos participantes. Observou-se que a ferramenta Appium é mais utilizada pelos participantes. Cerca de 30% dos participantes afirmam utiliza-la. A ferramenta Appium é uma ferramenta open-source e multi-plataforma. Com esta ferramenta é possível automatizar os testes de aplicações Android independente da linguagem utilizada. Quanto aos testes de unidade, as entrevistas apontam o uso do framework JUnit⁶, assim como a própria estrutura do framework Android Studio⁷.

Em relação as ferramentas de automação do teste de regressão, os participantes não indicaram nenhuma ferramenta específica para esta finalidade. Eles indicaram que utilizam as mesmas ferramentas para automação de teste, seja durante o desenvolvimento ou durante o processo de manutenção dos aplicativos. No processo do teste de regressão destaca-se o uso de ferramentas utilizadas para o gerenciamento dos testes que precisam ser reexecutados, assim como a criação de *pipelines* para automação do processo de

⁶<https://junit.org/>

⁷<https://developer.android.com/studio>

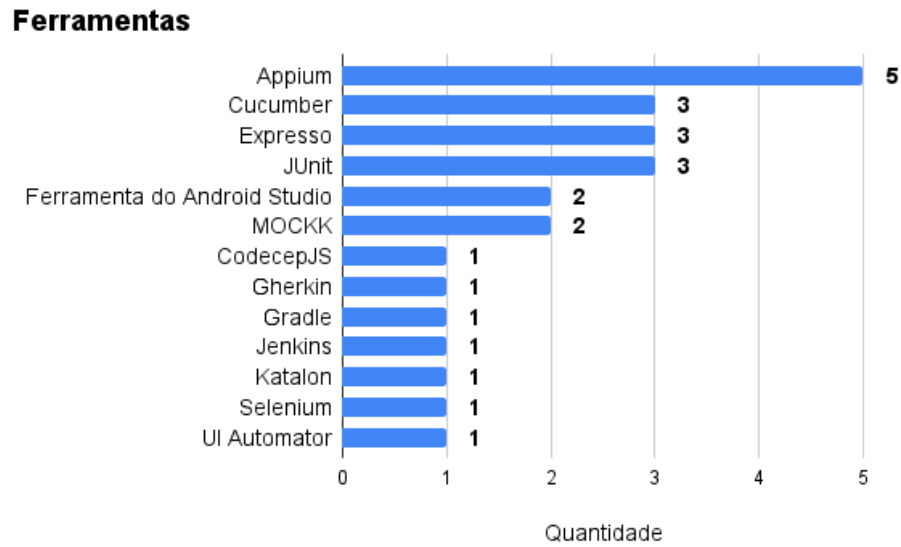


Figura 5.4 Entrevistas: Ferramentas para Automação dos Testes

reexecução.

Foi perguntado aos participantes quais requisitos eles consideravam essenciais para uma ferramenta atender satisfatoriamente a demanda de executar os testes de regressão. Buscou-se identificar requisitos para possíveis implementações nas ferramentas existentes ou na criação de uma nova ferramenta.

A Tabela 5.4 apresenta os principais requisitos funcionais indicados pelos participantes.

Os participantes elencaram também alguns requisitos não-funcionais importantes para essas ferramentas apresentados na Tabela 5.5.

Tabela 5.4 Entrevista: Requisitos funcionais elencados pelos participantes

	Descrição do Requisito Funcional
1	Mapear as modificações, para saber o que foi modificado. - P3
2	Identificar na lista de modificações mapeadas, quais tem testes vinculados, principalmente testes de unidade. - P3
3	Integrar teste de UI com testes de unidade. - P4.
4	Suportar testes em diferentes devices, em diferentes versões de Sistema Operacional, diferentes tamanhos de tela. - P5.
5	Editar scripts só que em devices diferentes. - P5.
6	Extrair relatórios de testes. - P5.
7	Medir a temperatura do <i>device</i> . - P5.
8	Fazer <i>tracking</i> de rede. Quais requisições eu mandei, o quanto eu recebi, era necessário tudo aquilo mesmo. - P5.
9	Fazer cenários negativos, tipo de falha de conexão, <i>retry</i> e afins. - P5.
10	Prover um <i>report</i> do que falhou e do que passou. - P7.
11	Possuir suporte para sistemas híbridos (multiplataforma - Android e IOS). - P11.
12	Identificar as atualizações da API do Android. - P13.
13	Trabalhar com pipelines. - P5.
14	Trabalhar com disparo automático (<i>triggers</i>). - P5.
15	Ferramenta que já escrevesse os testes de acordo com os cenários. - P16.
16	Criar uma biblioteca, uma base de execução, por exemplo, as tratativas que são necessárias para aguardar um elemento aparecer, as esperas implícitas quanto explícitas. - P9.
17	Conseguir guardar de alguma forma os passos, ou, eliminar repetição sistemática, onde eu possa validar cada efeito foi feito no cenário mais próximo do real. - P10.
18	Conseguir fazer com que esse fluxo completo fosse feito, algo próximo da integração. - P10.
19	Ferramenta que tivesse parametrização, como exemplo, parâmetros no recebimento e no envio. - P14.
20	Identificar e evitar <i>FLAKY tests</i> . - P15.

Tabela 5.5 Entrevistas: Requisitos Não-Funcionais

	Descrição do Requisito Não-Funcional
1	Ferramenta fácil de fazer alterações que possam aparecer no decorrer do teste. - P2.
2	Ferramenta de fácil uso. - P1, P2.
3	Ferramenta de fácil manutenção. - P2.
4	Ferramenta que não fosse pesada, ou seja, que não consuma muito recurso de hardware do equipamento. - P2.
5	Montar um mecanismo que ajude a fazer teste de regressão com mais agilidade. - P6.
6	Ferramenta que permitisse que o teste fosse executado por qualquer pessoa do time. - P7.
7	<i>Template</i> feito em HTML, onde contivesse os campos para colocar quais são as funcionalidades que eu quero rodar e onde estão contidas. - P7.
8	Resiliência, ou seja, que suporte intermitências causadas pelo ambiente. - P8.
9	Detalhar de forma clara, todos os passos que foram feitos. - P10.
10	Ter uma boa documentação. - P11.
11	Ter espaço para comunidade ir reforçando as falhas. - P11.
12	Ferramenta de fácil configuração. - P16.

Observa-se que os requisitos não-funcionais elencados pelos participantes não são específicos para ferramentas de automação de teste de regressão, mas, são aplicáveis a

quaisquer ferramenta.

Não foi realizada uma discussão com os participantes no que refere-se aos requisitos que eles apontaram.

5.3.2.4 Como os profissionais executam teste de regressão nos projetos em que atuam? (QP4)

Esta QP está relacionada com a compreensão de como os participantes do estudo lidam com o teste de regressão nos projetos de aplicativos Android em que eles atuam.

Foi perguntado aos participantes se existia algum passo-a-passo para testar uma nova versão do produto (e.g., fluxograma ou processo) durante a execução do teste de regressão. 50% dos participantes responderam que durante a execução do teste eles seguem um roteiro ou processo. Embora cada participante (P) tenha descrito um processo a ser seguido, observou-se um padrão de etapas, tais como:

1. Definir as funcionalidades que farão parte da *release* - P4;
2. Abrir solicitação de teste (feito pela equipe de desenvolvimento) - P1;
3. Disponibilizar a versão a ser testada (feito pela equipe de desenvolvimento) - P1;
4. Congelar a *branch* para evitar alterações enquanto o teste está sendo executado - P4;
5. Definir os casos principais e as exceções - P10;
6. Executar os testes (feito pela equipe de teste) - P4, P14;
7. Identificam-se os erros encontrados - P14.
8. Reportar os erros encontrados (feito pela equipe de teste) - P1.
9. Desenvolvedor gera uma nova versão retificada - P1, P14.
10. Após as correções serem efetuadas, a nova *release* do aplicativo sobe para produção - P1, P14.

Os participantes (P) que responderam não possuir um roteiro de execução de teste de regressão apontaram os seguintes motivos:

- A empresa possui um documento norteador de quais cenários devem ser testados - P5.
- Utilizam o conceito de metodologia Agile e não formalizam documentos - P7.
- O processo de teste de novas *releases* depende do escopo de cada projeto - P8.
- o processo de teste é feito de forma dinâmica e interativa - P9.

- A empresa trabalha com conceito de SQUAD ou Equipe, onde cada time define como será o fluxo de teste de novas *releases* - P12, P16.
- O processo de teste é feito em paralelo ao processo de desenvolvimento - P15.

Foi perguntado aos participantes se no processo de teste de uma nova *release* eles reexecutavam todos os casos de testes já existentes ou se definiam algum critério para escolher um subconjunto de casos de teste a serem reexecutados. Os resultados encontrados estão descritos a seguir:

- 31% dos participantes informaram que reexecutam todos os casos de teste já existentes para testar a nova *release*;
- 37% utilizam algum critério para selecionar, minimizar ou priorizar os casos de teste a serem reexecutados;
- 18% dos participantes indicaram que sempre que possível reexecutam todos os casos de teste, mas, utilizam algum critério para selecionar os casos de teste prioritários caso não haja tempo hábil para reexecutar tudo;
- 6% dos participantes informaram que depende do que o cliente definir;
- 6% dos participantes informaram que depende do *framework* de teste que estão utilizando.

Quanto aos critérios utilizados para selecionar, minimizar ou priorizar os casos de teste, encontramos os seguintes resultados:

- 44% dos participantes informaram que escolhem os casos de teste de acordo com o impacto e relação com as funcionalidades principais;
- 33% dos participantes criam um *MOCK* teste;
- 22% dos participantes selecionam os casos de teste relacionados a parte que foi alterada;
- 11% dos participantes responderam que selecionam os casos de teste que revelaram erro na primeira versão;
- 11% dos participantes responderam que a escolha é feita junto ao *Product Owner* (PO).

Nosso estudo indicou ainda que a escolha dos casos de teste a serem reexecutados é feito de forma manual.

Quanto ao questionamento sobre a percepção sobre a importância de sistematizar o processo de teste de regressão, 100% dos participantes responderam que consideravam ser algo importante. Como benefícios percebidos pelos participantes (P) da sistematização do processo de teste de regressão estão:

- Aumentar a cobertura do cenário de teste - P1.
- Entregar o produto no tempo hábil - P2, P7.
- Entregar o produto com qualidade - P2, P14.
- Reduzir problemas no relacionamento com o usuário - P2.
- Garantir que o produto que está sendo entregue está correto - P3, P6, P10, P15.
- Ganho de tempo - P4.
- Reduzir as atividades repetitivas e desnecessárias - P4, P7, P8.
- Tornar o processo menos suscetível a falha - P5, P7.
- Ganho de velocidade - P9.
- Identificar os problemas de forma mais rápida - P9.
- Garantir a eficiência - P11.
- Saber o que é necessário testar - P12.
- Definir precisão - P12.
- Definir impacto dos testes para os usuários - P12.
- Auxiliar na compreensão de como as coisas funcionam - P13.
- Garantir que o produto entregue não vai quebrar na produção - P16.

Foi perguntado aos participantes se existia relação entre executar o teste de regressão e a qualidade da *release* de um aplicativo. 68% dos participantes responderam que SIM, ou seja, executar teste de regressão proporciona garantia da qualidade da *release*. E, 31% dos participantes responderam que SIM, mas, NÃO SOMENTE, ou seja, fazer teste de regressão auxilia na qualidade da *release*, mas, não é o único fator que garante a qualidade da *release*. Os participantes (P) apresentaram alguns fatores que podem ser considerados ao tratar da execução de teste de regressão em aplicativos para Android. São eles:

- O teste de regressão não pode ser feito na correria ou sobre pressão - P2.
- É necessário olhar os pontos que a automação não cobre e aplicar testes exploratórios - P7.
- É necessário definir quais são os fluxos mais importantes - P8.
- É preciso fazer uma manutenção regular nos casos de teste - P10.
- É importante considerar a complexidade da variedade de *devices* e versões do Android - P11.

- Podem existir cenários que não estão cobertos pelos testes - P12.
- É necessário ter cuidado para não criar cenários falsos positivos - P16.

5.4 DISCUSSÃO

5.4.1 Nível de Conhecimento e Automação de Teste

Os participantes, em sua maioria, não iniciaram a carreira na área específica de teste de aplicativos para Android. Observou-se que existe um processo de migração de carreira, algumas vezes até mesmo como oportunidade na própria empresa que em trabalham, visto a necessidade e expansão da área de mobile na indústria brasileira. Alguns participantes também migraram da área de desenvolvimento para a área de teste/qualidade de software. Outro ponto importante a ser ressaltado é a mudança de cultura no que refere-se ao desenvolvimento, visto que hoje, algumas empresas já implementam como boas práticas que o processo de teste da aplicação seja iniciado em paralelo ao processo de desenvolvimento.

Observou-se que as principais fontes de aprendizado sobre teste de aplicativos para Android, de acordo com as entrevistas, são a prática profissional e a pesquisa na internet (literatura cinza). Em geral, os profissionais com mais experiência, compartilham conhecimento com quem está começando. Nenhum dos participantes apontou a faculdade como fonte de aprendizado e ainda ressaltaram que o conhecimento sobre teste de software é um assunto pouco explorado na faculdade. Os participantes relatam que sentem falta de um preparo maior, no que refere a academia para atuarem no mercado de teste. A área de teste de aplicativos para Android, de acordo com as entrevistas, é uma área relativamente nova e em expansão.

Observou-se que a forma como os participantes executam o teste de regressão nos projetos em que atuam, está relacionado com as práticas e processos definidos pela empresa.

5.4.2 Linguagens e Automação

Este estudo tinha como suposição inicial que o uso de linguagens nativas, ou seja, Java e Kotlin, proporcionaria que os testes dos aplicativos desenvolvidos nestas linguagens fossem mais automatizados do que os testes dos aplicativos desenvolvidos em linguagens híbridas. Tal suposição partiu do princípio de que as linguagens nativas possuem um ferramental de apoio mais estável e mais sólido. Porém, os resultados deste estudo não comprovaram que a linguagem utilizada no processo de desenvolvimento tem relação direta com o processo de automação dos testes de regressão. Conforme apresentado na Seção 5.3, o número de participantes que acreditam que existe relação entre linguagem e automação e o de participantes que não fazem essa correlação é proporcional.

Enquanto as linguagens nativas, como Java e Kotlin, contam com ferramentas mais estáveis e com mais tempo de mercado, as linguagens híbridas proporcionam para a equipe agilidade para desenvolver com um único código, um aplicativo que seja multi-plataforma, ou seja funciona tanto em Android quanto em IOS, por isso, o mercado vem desenvolvendo ferramentas que testem os aplicativos gerados por essas linguagens.

Observou-se que automatizar ou não automatizar o teste de regressão está mais relacionado ao nível de maturidade da empresa e o domínio na qual ela está inserida.

5.4.3 Ferramentas para Automação

Este estudo apontou as principais ferramentas utilizadas pelos participantes no processo de automação dos testes. Observou-se que os participantes que trabalham como desenvolvedores utilizam mais as ferramentas voltadas para testes de unidade, como o JUnit, enquanto os participantes que trabalham com teste e qualidade, utilizam ferramentas voltadas para teste de interface gráfica, performance, estresse e demais tipos de teste.

Embora a literatura tenha desenvolvido ao longo do tempo ferramentas específicas para a execução de teste de regressão (AMALFITANO et al., 2011; SZABÓ et al., 2012; BAUERSFELD, 2013; HU; NEAMTIU, 2016; LI et al., 2017; CHANG et al., 2018; KONG et al., 2019), este estudo demonstrou que os participantes não utilizam nenhuma ferramenta específica para automatizar o teste de regressão. Neste contexto, os participantes apresentam como diferencial a utilização de ferramentas gerenciais que listam ou organizam quais casos e cenários de teste precisam ser reexecutados, tais como o JIRA⁸.

Este estudo identificou uma lista de requisitos funcionais e não-funcionais esperados pelos participantes nas ferramentas para automação de teste de regressão. Pode-se ressaltar: (i) algumas das funcionalidades elencadas já existem, porém em ferramentas não específicas para teste de regressão, (ii) o mercado atualmente disponibiliza diversas ferramentas que possuem recursos diversos, porém não foi encontrada uma ferramenta que agregue todos esses recursos necessários, (iii) outro desafio é conseguir integrar em uma mesma ferramenta de teste de regressão, os testes de unidade e os testes de *Graphical User Interface* (GUI).

A sugestão proposta como resultado deste estudo é o desenvolvimento de uma ferramenta com requisitos listados pelos participantes, assim como, trabalhar com a conscientização dos profissionais da importância da automação de testes.

5.4.4 Teste de Regressão

A literatura ao longo do tempo apresenta pesquisas sobre teste de regressão (LEUNG; WHITE, 1989; LEUNG; WHITE, 1991; ROTHERMEL; HARROLD, 1996; GRAVES et al., 2001; AMMANN; OFFUTT, 2008; ENGSTRÖM et al., 2010; YOO; HARMAN, 2012; KAZMI et al., 2017) tanto para compreender este tipo de teste, quanto no estudo e desenvolvimento de técnicas de teste de regressão, tais como técnicas de seleção, minimização e priorização (YOO; HARMAN, 2012), que tem como objetivo reduzir o número de casos de teste a serem reexecutados. Alguns destes estudos (DO et al., 2016; CHANG et al., 2018; JIANG et al., 2018) apresentam ferramentas que foram implementadas baseadas nestas técnicas.

Este estudo revelou que 56% dos participantes aplicam os conceitos propostos pelas técnicas de seleção, minimização ou priorização, porém, sem um conhecimento formal sobre elas. Ainda, observou-se que o processo de escolha dos casos de teste a serem

⁸<https://www.atlassian.com/br/software/jira>

reexecutados é feito de forma manual.

Observou-se que quanto mais automatizado é o processo de teste de regressão, mais é realizada a reexecução de todos os casos de teste, em virtude do participante acreditar que desta forma tem uma relação melhor de custo/benefício, visto que as escolhas de casos de teste são feitas de forma manual. De modo similar, os desenvolvedores costumam optar por reexecutar todos os casos de teste por receio de deixar de cobrir alguma falha no aplicativo. Em contrapartida, quando o processo de teste é menos automatizado, visto a quantidade de casos de teste existentes e o tamanho da equipe de execução de teste é pequeno, é feito um processo de seleção de casos de teste, de forma manual e de acordo com a experiência e percepção do profissional responsável.

Este estudo revelou ainda que a importância concedida a atividade de teste, em especial, teste de regressão, dentro das empresas, tem correlação com o domínio de atuação. Empresas que atuam no mercado financeiro, por exemplo, tendem a priorizar as atividades de teste, desde a etapa de desenvolvimento, visto o impacto que um erro pode proporcionar no aplicativo e por consequência, o prejuízo para a empresa de desenvolvimento e para o cliente.

Outro ponto importante, revelado neste estudo é que embora 100% dos participantes acreditam ser importante sistematizar o processo de teste de regressão, apenas 50% dos participantes indicaram ter um processo, fluxograma, ou algum passo-a-passo que define como eles devem testar as novas *releases*. Embora ao decorrer das entrevistas tenhamos solicitado acesso aos modelos de documentos utilizados pelos profissionais, os documentos não foram compartilhados em virtude de não ser autorizado por parte das empresas.

Por fim, este estudo revela que na percepção dos participantes o teste de regressão auxilia na garantia da qualidade da nova *release* do aplicativo, porém, somente executar teste de regressão não garante a qualidade da *release*.

5.5 AMEAÇAS À VALIDADE

Validade Interna. Uma importante ameaça à validade interna diz respeito ao número reduzido de participantes do estudo. Embora tenha sido realizado contato com vários profissionais por e-mail, grupos de WhatsApp, LinkedIn - Mala Direta, Grupos de Comunidades da área e contatos com profissionais indicados pelos participantes, apenas 16 responderam, afirmando que poderiam participar na entrevista. Para mitigar este problema, foram entrevistados profissionais que trabalham em projetos estruturados, em que o aplicativo é um produto essencial. Outra ameaça à validade interna deste estudo refere-se ao processo de codificação das entrevistas ter sido realizado de forma individual.

Validade externa. Embora este estudo seja baseado em uma amostra que considerou profissionais que atuam em projetos de desenvolvimento de software para a plataforma Android, os resultados não podem ser generalizados para outros cenários, dado o pequeno quantitativo de participantes. No entanto, este estudo é um passo inicial para fornecer mais evidências sobre a consciência e percepção dos profissionais sobre o uso de teste de regressão na manutenção de aplicativos Android.

Validade de construto. Algumas perguntas feitas na entrevista podem não ter sido claras para alguns participantes, o que pode causar interpretações erradas durante

as respostas e conseqüentemente ao analisá-las também. Para mitigar esta situação foi realizado um estudo piloto antes de inicializar a entrevista.

5.6 SÍNTESE DO CAPÍTULO

O presente capítulo apresentou as entrevistas utilizadas como instrumento de pesquisa para compreender como os profissionais lidam com teste de regressão nos projetos de aplicativos para Android.

No próximo capítulo será apresentada uma análise temática sobre o uso de técnicas de teste de regressão em projetos de aplicativos para Android.

6

SÍNTESE TEMÁTICA

O presente capítulo apresenta uma síntese temática como resultado de um estudo multi-método (literatura, *survey* e entrevistas) que trata sobre a adoção de técnicas de teste de regressão em projetos de software para a plataforma Android.

6.1 APRESENTAÇÃO

Síntese temática é uma das metodologias utilizadas em pesquisas qualitativas com os objetivos de resumir, integrar, combinar e comparar as descobertas referentes a estudos primários sobre um tópico específico ou questão de pesquisa (CRUZES; DYBÅ, 2010; CRUZES; DYBÅ, 2011b). Foi adotada nesta síntese temática a metodologia definida por Cruzes e Dybå (2011a) e refinada por Prates (2015).

A próxima seção apresentará as etapas metodológicas utilizadas para realização dessa síntese temática.

6.2 METODOLOGIA DA PESQUISA

6.2.1 Extração de Dados

Para a realização desta síntese temática foram utilizadas três fontes de dados: literatura, *survey* e as entrevistas.

1. **Literatura:** A primeira fonte de dados foram estudos primários encontrados na literatura. A busca na literatura foi realizada com base na metodologia proposta por Arksey e O'Malley (2005).
 - **Crítérios PICOC:** A Tabela 6.1 apresenta os critérios utilizados: População, Intervenção, Comparação, Saídas e Contexto.

Tabela 6.1 Síntese Temática: Critérios PICOC

População	Pesquisadores, desenvolvedores e testadores de software
Intervenção	Técnicas de Teste de Regressão
Comparação	<i>Não aplicável</i>
Saídas (<i>Outcomes</i>)	Técnicas de Teste de Regressão para Android
Contexto	Avaliadas Empiricamente

- **Estratégias de Busca:** A estratégia de busca foi dividida em três etapas:
 - (a) **Busca Automática:** Busca automática realizada nas bases de dados definidas na Tabela 6.2, utilizando-se da string de busca principal apresentada na Tabela 6.3 .

Tabela 6.2 Síntese Temática: Bases de Dados

Biblioteca Digital	URL
<i>ACM Digital Library</i>	https://dl.acm.org/
<i>DBLP</i>	https://dblp.uni-trier.de/
<i>IEEE Xplore</i>	https://ieeexplore.ieee.org/Xplore/home.jsp
<i>Scopus</i>	https://www.scopus.com/home.uri
<i>Science Direct</i>	https://www.sciencedirect.com/

Tabela 6.3 Síntese Temática: String de Busca

(“*regression test*” **OR** “*regression testing*”)
 AND
 (“Android” **OR** “Android *applications*” **OR** “Android apps” **OR** “APPS”)

- (b) **Busca Manual:** Busca manual realizada nas principais conferências das áreas de Engenharia de Software e de Teste.
 - (c) ***Snowballing*:** Análise das referências dos artigos identificados na busca automática e na busca manual.
- **Critérios para Seleção de Estudos Primários:** Para inclusão ou exclusão dos estudos foram utilizados os seguintes critérios:
 - **Critérios de Inclusão:**
 - * Estudos na área de Computação e Engenharia de Software
 - * Estudos que tratem sobre Técnicas de Teste de Regressão para Android
 - * Estudos que possuam avaliação empírica
 - * Estudos escritos na língua inglesa
 - **Critérios de Exclusão:**

- * Estudos duplicados
- * Estudos sem comprovação empírica
- * Estudos não escritos em inglês
- * Versão reduzida de outros estudos

Para que um estudo seja aceito, precisa contemplar todos os critérios de inclusão. Um estudo será excluído se atender pelo menos um critério de exclusão. O processo de seleção dos artigos será feita através do processo a seguir:

- Excluir estudos candidatos por título ou resumo
 - Excluir estudos candidatos depois de ler alguma seção
 - Excluir estudos candidatos depois de ler todo o artigo
- **Resultados encontrados:** A Tabela 6.4 apresenta os artigos encontrados após aplicação dos critérios de inclusão e exclusão, e os macro temas relacionados.

Tabela 6.4 Síntese Temática: Artigos encontrados na literatura

#	Título do Artigo	Referência
S1	<i>A framework for testing Android apps by reusing test cases</i>	(JHA et al., 2019)
S2	<i>A GUI Crawling-based technique for Android Mobile Application Testing</i>	(AMALFITANO et al., 2011)
S3	<i>ATOM: Automatic Maintenance of GUI Test Scripts for Evolving Mobile Applications</i>	(LI et al., 2017)
S4	<i>Automated Testing of Android Apps: A Systematic Literature Review</i>	(KONG et al., 2019)
S5	<i>Change-Based Test Script Maintenance for Android Apps</i>	(CHANG et al., 2018)
S6	<i>Database Refactoring and Regression Testing of Android Mobile Applications</i>	(SZABÓ et al., 2012)
S7	<i>DetReduce: Minimizing Android GUI Test Suites for Regression Testing</i>	(CHOI et al., 2018)
S8	<i>GUIDiff – A Regression Testing Tool for Graphical User Interfaces</i>	(BAUERSFELD, 2013)
S9	<i>Meeting Quality Standards for Mobile Application Development in Businesses: A Framework for Cross-Platform Testing</i>	(GRØNLI; GHINEA, 2016)
S10	<i>Mining Test Repositories for Automatic Detection of UI Performance Regressions in Android Apps</i>	(GÓMEZ et al., 2016)
S11	<i>Redroid: A regression test selection approach for android applications</i>	(DO et al., 2016)
S12	<i>ReTestDroid: Towards Safer Regression Test Selection for Android Application*</i>	(JIANG et al., 2018)
S13	<i>VALERA: An Effective and Efficient Record-and-replay Tool for Android</i>	(HU; NEAMTIU, 2016)

2. **Survey:** A segunda fonte de dados utilizada nesta síntese temática foram os resultados obtidos com a aplicação do *Survey* para 100 profissionais/pesquisadores que atuavam em projetos de aplicativos para Android, descritos na Seção 4.3.
3. **Entrevistas:** A terceira fonte de dados utilizada nesta síntese temática foram os resultados obtidos com a realização de entrevista com 16 profissionais que atuavam em projetos de aplicativos para Android, descritos na Seção 5.3.

6.2.2 Codificação

O tipo realizado de codificação foi aberta (HODA, 2021), sem uso de ferramentas, identificando-se as palavras-chave nos resultados dos estudos primários citados anteriormente. Os códigos obtidos serão apresentados na Tabela 6.5:

Tabela 6.5 Síntese Temática: Códigos x Fonte de Dados

	Código	Fonte de Dados
1	Abordagens / <i>Framework</i>	Literatura
2	Android	Literatura / <i>Survey</i> / Entrevistas
3	Aplicativos	Literatura / <i>Survey</i> / Entrevistas
4	Automação de teste	<i>Survey</i> / Entrevistas
5	Casos de teste	<i>Survey</i> / Entrevistas
6	Experiência profissional	<i>Survey</i> / Entrevistas
7	Ferramentas	Literatura / <i>Survey</i> / Entrevistas
8	Fontes de aprendizado	Entrevistas
9	Funcionalidades / <i>Survey</i> / Entrevistas	
10	Linguagens de programação	Entrevistas
11	Linguagens (nativas e não-nativas)	Entrevistas
12	Manutenção	Literatura / <i>Survey</i>
13	Nível de Conhecimento	<i>Survey</i>
14	Profissional	<i>Survey</i> / Entrevistas
15	Requisitos / <i>Survey</i> / Entrevistas	
16	Técnicas	Literatura
17	Teste	Literatura / <i>Survey</i> / Entrevistas
18	Teste de Regressão	Literatura / <i>Survey</i> / Entrevistas
19	Teste em cenário de evolução	Entrevistas
20	Tipos de Teste	<i>Survey</i>

6.2.3 Tradução de códigos em temas

Com os códigos gerados a partir do processo de extração de dados, foram gerados quatro temas: teste de regressão, técnicas, aplicativos e profissional.

6.2.3.1 Teste de regressão Este tema está relacionado ao foco deste estudo, como apresenta a Figura 6.1. O foco deste estudo é teste de software. Ao longo do desenvolvimento de um software os testes são planejados, especificados, implementados, executados e os resultados são apresentados. O teste de regressão é utilizado para garantir que as

manutenções realizadas durante o processo de evolução do software não afetaram seu comportamento funcional de forma não-intencional.

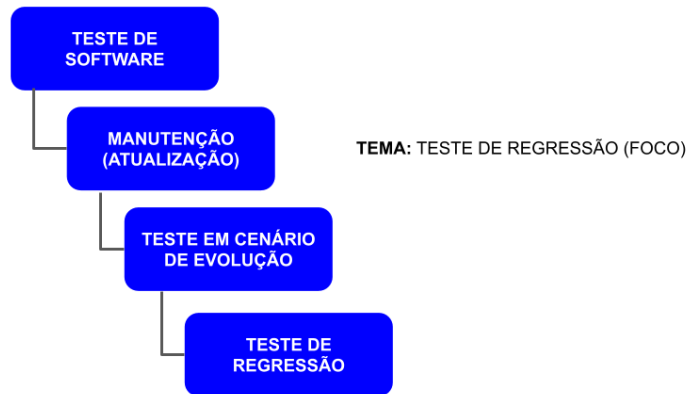
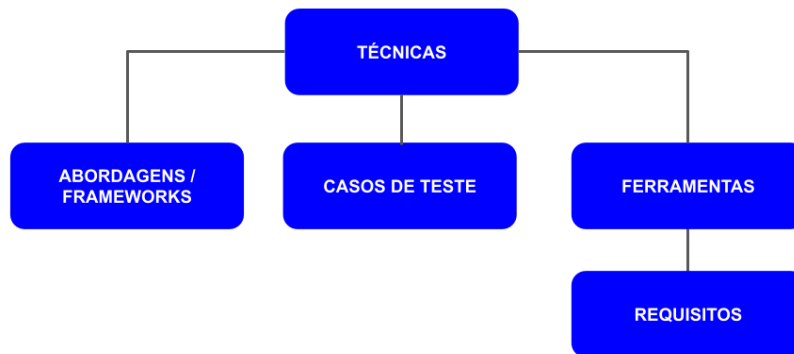


Figura 6.1 Síntese Temática: Tema Teste de Regressão

6.2.3.2 Técnicas Este tema está relacionado a aplicação deste estudo, como apresenta a Figura 6.2. Este estudo avalia o uso das três técnicas de teste de regressão mais conhecidas, que são elas: técnicas de seleção, (ii) técnicas de minimização e (iii) técnicas de priorização. A literatura ao longo dos anos apresenta estudos criando abordagens e frameworks baseados nessas técnicas. Essas técnicas propostas pela literatura visam a redução dos casos de teste a serem reexecutados no processo de teste de novas versões. Por fim, a indústria produz ferramentas para automatizar o processo de teste de regressão, usualmente, baseadas na reexecução de todos os casos de teste, sendo a forma primária de execução de teste de regressão. Este estudo identifica quais são as ferramentas já utilizadas pelos profissionais e quais são os requisitos esperados nas ferramentas para automação de teste de regressão.



TEMA: TÉCNICAS (APLICAÇÃO)

Figura 6.2 Síntese Temática: Tema Técnicas

6.2.3.3 Aplicativos Este tema está relacionado a área de domínio a qual este estudo está inserido, como apresenta a Figura 6.3. O presente estudo foca no uso das técnicas de teste de regressão em aplicativos, especificamente, para Android. Também busca identificar quais são as linguagens mais utilizadas no desenvolvimento de aplicativos para Android, categorizando-as como linguagens nativas e não-nativas e correlacionando a escolha da linguagem de programação com o processo de automação dos testes.

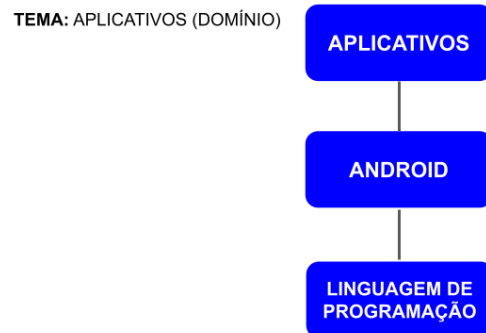


Figura 6.3 Síntese Temática: Tema Aplicativos

6.2.3.4 Profissional Este tema está relacionado ao público alvo deste estudo, como apresenta a Figura 6.4. Este estudo busca compreender o uso das técnicas de teste de regressão por profissionais, desenvolvedores ou testers, que atuam em projetos de aplicativos para Android, identificando o nível de conhecimento, a experiência profissional e as fontes de aprendizado por eles.

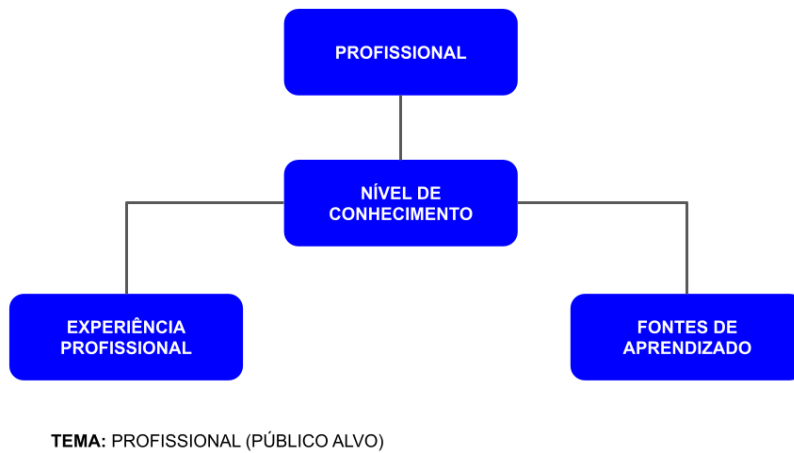


Figura 6.4 Síntese Temática: Tema Profissional

6.2.4 Criação de um modelo temático

Baseado nos temas TESTE DE REGRESSÃO, TÉCNICAS, APLICATIVOS e PROFISIONAL foi criado um modelo temático apresentado na Figura 6.5. O modelo temático permite representar o propósito deste estudo que refere-se ao o uso das técnicas de teste de regressão por profissionais que atuam em projetos de aplicativos para Android.

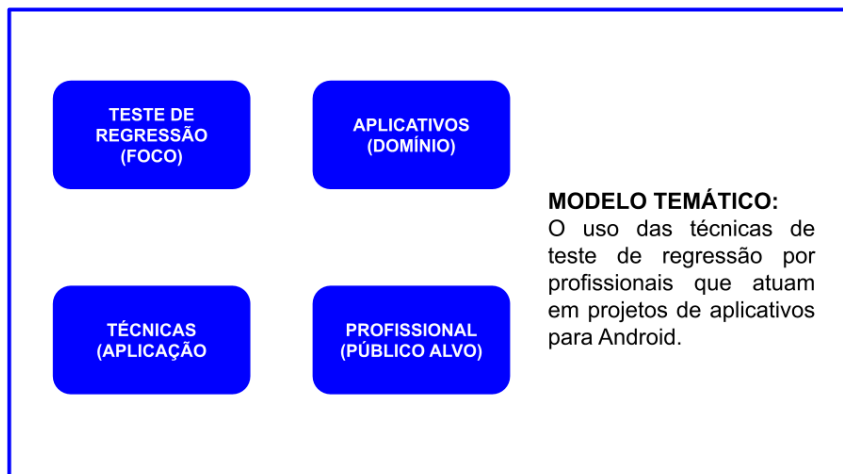


Figura 6.5 Síntese Temática: Modelo Temático

6.3 DISCUSSÃO

A seguir serão apresentadas as discussões acerca do modelo temático proposto.

6.3.1 Teste de Regressão

Ao longo dos anos com o aumento da utilização e importância dos softwares, assim como, na busca de estratégias que garantam a qualidade destes produtos que são desenvolvidos, a atividade de teste de software vem sendo estudada pelos pesquisadores em suas diversas vertentes, tais como: estudos sobre tipos de teste (testes funcionais x testes estruturais), categorias de teste (teste de unidade, teste de integração, teste de validação, teste de sistema), automação e assuntos correlatos.

O processo constante de evolução de software também proporcionou uma vasta área de estudos, assim como, estudos relacionados a teste de regressão, visto na literatura, como o teste utilizado para verificar se o software que passou por manutenção, não foi afetado de maneira não intencional e continua atendendo aos requisitos anteriormente já validados.

Embora a literatura já tenha provido diversos estudos com validação empírica sobre o conceito e a importância da execução do teste de regressão ao longo do processo de desenvolvimento de um software, na prática, observamos que os profissionais que atuam no mercado de projetos de aplicativos para Android nem sempre possuem este conhecimento, e também, nem sempre executam teste de regressão.

Os estudos realizados apontam alguns fatores que contribuem para esta realidade, tais como: (i) equipes de teste com poucos colaboradores, (ii) tempo de entrega reduzido para o lançamento das novas *releases*, (iii) processos não estabelecidos, (iv) necessidade de alto esforço da equipe de teste, (v) baixo nível de automação dos testes, (vi) cultura organizacional que não valoriza a importância da atividade de teste e (vii) ausência ou baixo conhecimento teórico sobre teste de regressão por parte dos profissionais.

Os resultados obtidos com as entrevistas apontaram ainda que a valorização da atividade de teste por parte das empresas tem correlação com o domínio em que elas atuam. Assim, empresas que atuam com aplicativos voltados para a área financeira demonstram uma maior preocupação com a qualidade e processos de testes, do que empresas que atuam em áreas onde um erro no aplicativo não tem tanto impacto financeiro.

6.3.2 Técnicas

Na realização do teste de regressão tende-se a reexecutar todos os casos de teste da versão original do software para testar a versão modificada, porém, conforme apresentado na literatura, com o crescimento do software e ao decorrer do seu processo evolutivo, consequentemente com o aumento do número dos casos de teste, esta estratégia torna-se menos viável. Diante deste cenário, a literatura propõe as técnicas de teste de regressão, sendo as mais conhecidas, as técnicas de seleção, de minimização e de priorização. Todas estas técnicas buscam reduzir o conjunto de casos de teste inicial criando um subconjunto representativo para ser reexecutado no processo de teste da nova versão.

A literatura apresenta diversos estudos empíricos sobre a aplicação destas técnicas em domínios diferentes, dentre eles, o domínio de aplicativos para Android, onde pesquisadores propõem abordagens, *frameworks* e ferramentas que implementam essas técnicas, reduzindo o esforço e o custo do processo de teste de regressão.

Os estudos realizados junto aos profissionais que atuam em projetos de desenvolvi-

mento de aplicativos para Android e apresentados anteriormente (*survey* e as entrevistas) demonstraram que alguns profissionais aplicam o conceito dessas técnicas que a literatura propõe (minimização, seleção e priorização), porém, sem o conhecimento formal sobre elas. Observou-se ainda que quanto mais o processo de teste é automatizado, a estratégia de reexecutar todos os casos de teste é a mais utilizada. Em contrapartida, quanto menos o processo de teste é automatizado, mais utiliza-se de alguma estratégia de redução dos casos de teste a serem reexecutados para testar a nova versão, com o objetivo de reduzir o tempo e o esforço na realização do teste de regressão.

Ainda, observa-se que essas soluções que a literatura implementa (abordagens, *frameworks* e ferramentas) não são de conhecimento dos profissionais e nem sempre estão disponíveis em forma de código-fonte para que sejam testadas. Algumas dessas soluções apresentam-se como modelos formais, e não como ferramental de apoio.

Por outro lado, as ferramentas disponíveis no mercado para automação de teste em aplicativos para Android ainda deixam lacunas que geram resistência aos profissionais no que refere-se ao processo de automação dos testes. Quando trata-se de automação do teste de regressão, esta lacuna é ainda maior. As ferramentas disponíveis no mercado mais utilizadas trabalham utilizando a técnica de reexecução de todos os casos de testes automatizados, e na maioria das vezes, não integram testes unitários com testes instrumentados.

No processo de realização das entrevistas semi-estruturadas, os participantes elencaram diversos requisitos, funcionais e não-funcionais, que consideravam ser relevantes para uma ferramenta de automação de teste de regressão para Android. Dentre os requisitos funcionais elencados, observa-se que a maior parte deles já são contemplados nas ferramentas existentes, seja ferramenta para automação de teste de regressão ou ferramenta de CI/CD. Porém, a proposta com que esse questionamento era identificar esses requisitos e contemplá-los em uma única ferramenta.

Observa-se então a necessidade de implementar uma ferramenta baseada nas técnicas propostas pela literatura e que possam ser disponibilizadas e assim utilizadas pelos profissionais que atuam neste cenário.

6.3.3 Aplicativos

Pesquisas recentes apontam o crescimento do mercado de dispositivos móveis, e consequentemente, da demanda de aplicativos para diversos domínios, como entretenimento, finanças, compras, utilitários e entre outros. As pesquisas apontam que o Android é o sistema para dispositivos móveis mais utilizado. Este cenário indica a necessidade de estudos voltados para este domínio específico de software, aplicativos mobile.

O desenvolvimento de aplicativos para Android é desafiador, quer seja pela constante evolução deste sistema operacional, ou pela necessidade de desenvolver um código-fonte que rode em diferentes dispositivos, com diferentes configurações de hardware e de software.

Com o crescimento do mercado mobile, outro fator desafiador é definir a linguagem de programação a ser utilizada, ou seja, optar uma linguagem nativa, definida pela empresa que desenvolve o Android, ou escolher uma linguagem não-nativa, também conhecida

como linguagem híbrida, que permite que com um único código-fonte construa-se um aplicativo que rode em múltiplas plataformas. Assim, este torna-se outro campo de pesquisa em expansão.

6.3.4 Profissional

Os estudos realizados com os profissionais trazem dados relevantes referente ao perfil de quem atua nos projetos de aplicativos para Android. Os resultados demonstram que a área de teste de aplicativos para Android no Brasil, é uma área em expansão que tem captado profissionais de outras áreas, correlatas ou não, que estejam dispostos a passar por um processo de migração de área. Quando trata-se de experiência na área de testes, os profissionais já possuem uma trajetória estável, porém, ao tratar-se da área específica de teste de aplicativos para Android, esta experiência é menor, às vezes, até mesmo, em estágio inicial.

Embora a maior porcentagem dos entrevistados sejam da área de informática, computação ou áreas afins, o mercado também abarca profissionais de outras áreas, nem sempre correlatas, mas que apresentam experiência prática.

Outro ponto relevante refere-se as fontes de aprendizagem. Os resultados demonstram que os profissionais aprendem mais sobre a atividade que executam como desenvolvedores e testadores, com a prática profissional e a busca de conteúdo na literatura cinza (blogs, sites, artigos não acadêmicos, fóruns, vídeos e entre outros). Os participantes apontam ainda não terem recebido conhecimento sobre teste na faculdade de forma satisfatória e que atenda de fato, as necessidades encontradas por eles na atuação profissional no mercado.

6.3.5 O uso das técnicas de teste de regressão por profissionais que atuam em projetos de aplicativos para Android

A literatura apresenta diversos estudos com validação empírica sobre teste de regressão e técnicas de teste de regressão, porém, observa-se que este conhecimento está distante dos profissionais que atuam na prática do projeto de desenvolvimento de aplicativos para Android.

Observa-se uma lacuna entre o que a academia produz e o que de fato os profissionais necessitam para executarem teste de regressão com qualidade nos projetos em que atuam.

Os estudos realizados apontam a necessidade da implementação de novas ferramentas que contemplem os requisitos necessários para atenderem os profissionais que atuam neste cenário, assim como, que essas ferramentas sejam disponibilizadas em repositórios para que a comunidade profissional possa utilizar e sugerir melhorias, ou até mesmo contribuir com a implementação.

Esta lacuna observa-se também quando trata-se da abrangência do estudo de teste de software na academia em relação ao que o mercado espera dos profissionais, em especial, no que refere-se ao domínio de aplicativos mobile.

6.4 SÍNTESE DO CAPÍTULO

Neste capítulo foi apresentado a síntese temática dos achados referente ao tema desta dissertação, oriundos da literatura, do *Survey* e das entrevistas. Foram as apresentadas as etapas metodológicas realizadas, assim como, a proposição do modelo temático e as referidas discussões.

No próximo capítulo serão apresentadas as considerações finais desta dissertação, considerando os resultados, contribuições e trabalhos futuros.

7

CONSIDERAÇÕES FINAIS

Este capítulo destina-se a apresentar uma síntese dos resultados obtidos, das contribuições e trabalhos futuros.

7.1 RESULTADOS

A presente dissertação teve como objetivo investigar a adoção das técnicas de teste de regressão por profissionais que atuam em projetos de software para a plataforma Android, tendo como recorte profissionais que atuam no cenário brasileiro. Para alcançar esse objetivo, foram realizados dois estudos exploratórios, sendo o primeiro um *survey* aplicado a 100 profissionais, e o segundo uma entrevista realizada com 16 profissionais. Esses estudos foram analisados de forma sintetizada e comparativa, junto aos achados na literatura e apresentados na síntese temática apresentada.

Com a conclusão dos estudos realizados foi possível obter os seguintes resultados da pesquisa:

1. Apresentação de evidências empíricas que demonstram como os profissionais que atuam em projetos de aplicativos para Android realizam teste de regressão.
2. Identificação de como os profissionais utilizam as técnicas de teste de regressão apresentadas na literatura.
3. Apresentação do conjunto de requisitos necessários para atender a demanda de execução de teste de regressão indicadas pelos profissionais que participaram dos estudos.

7.2 CONTRIBUIÇÕES

Os estudos presentes nesta dissertação apresentam contribuições para a área de Engenharia de Software, especialmente para pesquisas relacionadas à teste de regressão tendo como público alvo profissionais que atuam em projetos de aplicativos para Android. As principais contribuições desta dissertação são:

1. Apresentação do nível de conhecimento dos profissionais que atuam em projetos de aplicativos para Android sobre teste de regressão.
2. Apresentação de como profissionais que atuam em projetos de aplicativos para Android lidam e executam teste de regressão.
3. Apresentação da síntese temática sobre o uso das técnicas de teste de regressão em projetos de aplicativos para Android.
4. Apresentação dos requisitos necessários para execução de teste de regressão apresentadas pelos profissionais que atuam neste cenário.
5. Apresentação das metodologias utilizadas em todos os estudos para que possam ser replicados ou estendidos.
6. Apresentação do artigo científico intitulado “*Unveiling Practitioners Awareness of Android Apps Regression Testing through an Expert Survey*”(LIMA et al., 2020).

7.3 TRABALHOS FUTUROS

A presente dissertação apresenta um ponto de partida no que refere-se ao estudo de como profissionais que atuam em projetos de aplicativos para Android lidam com teste de regressão no seu cotidiano. Assim, como trabalhos futuros a serem realizados, pretende-se:

1. Replicar o estudo referente as entrevistas com um número maior de profissionais.
2. Propor um modelo formal de ferramenta que contemple os requisitos elencados pelos participantes como necessárias para execução de teste de regressão.
3. Validar o modelo temático proposto através de pesquisa ativa junto aos profissionais que atuam no cenário desta dissertação.
4. Investigar como a academia pode contribuir na formação no que refere-se a teste de aplicativos de acordo com a necessidade do mercado de trabalho.

REFERÊNCIAS

ALI, N. B.; ENGSTRÖM, E.; TAROMIRAD, M.; MOUSAVI, M. R.; MINHAS, N. M.; HELGESSON, D.; KUNZE, S.; VARSHOSAZ, M. On the search for industry-relevant regression testing research. *Empir. Softw. Eng.*, v. 24, n. 4, p. 2020–2055, 2019. Disponível em: <https://doi.org/10.1007/s10664-018-9670-1>.

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P. A GUI crawling-based technique for android mobile application testing. In: *Fourth IEEE International Conference on Software Testing, Verification and Validation, ICST 2012, Berlin, Germany, 21-25 March, 2011, Workshop Proceedings*. IEEE Computer Society, 2011. p. 252–261. Disponível em: <https://doi.org/10.1109/ICSTW.2011.77>.

AMMANN, P.; OFFUTT, J. *Introduction to Software Testing*. Cambridge University Press, 2008. ISBN 978-0-521-88038-1. Disponível em: <https://doi.org/10.1017/CBO9780511809163>.

ARKSEY, H.; O'MALLEY, L. Scoping studies: towards a methodological framework. *International Journal of Social Research Methodology*, Routledge, v. 8, n. 1, p. 19–32, 2005. Disponível em: <https://doi.org/10.1080/1364557032000119616>.

BAUERSFELD, S. Guidiff - A regression testing tool for graphical user interfaces. In: *Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013, Luxembourg, Luxembourg, March 18-22, 2013*. IEEE Computer Society, 2013. p. 499–500. Disponível em: <https://doi.org/10.1109/ICST.2013.84>.

CHANG, N.; WANG, L.; PEI, Y.; MONDAL, S. K.; LI, X. Change-based test script maintenance for android apps. In: *2018 IEEE International Conference on Software Quality, Reliability and Security, QRS 2018, Lisbon, Portugal, July 16-20, 2018*. IEEE, 2018. p. 215–225. Disponível em: <https://doi.org/10.1109/QRS.2018.00035>.

CHAPIN, N.; HALE, J. E.; KHAN, K. M.; RAMIL, J. F.; TAN, W. Types of software evolution and software maintenance. *J. Softw. Maintenance Res. Pract.*, v. 13, n. 1, p. 3–30, 2001. Disponível em: <https://doi.org/10.1002/smr.220>.

CHOI, W.; SEN, K.; NECULA, G. C.; WANG, W. Detreduce: minimizing android GUI test suites for regression testing. In: CHAUDRON, M.; CRNKOVIC, I.; CHECHIK, M.; HARMAN, M. (Ed.). *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. ACM, 2018. p. 445–455. Disponível em: <https://doi.org/10.1145/3180155.3180173>.

CHOUDHARY, S. R.; GORLA, A.; ORSO, A. Automated test input generation for android: Are we there yet? (E). In: COHEN, M. B.; GRUNSKÉ, L.; WHALEN, M. (Ed.). *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*. IEEE Computer Society, 2015. p. 429–440. Disponível em: <https://doi.org/10.1109/ASE.2015.89>.

CROSBY, P. *Quality is Free: The Art of Making Quality Certain*. McGraw-Hill, 1979. (Mentor book). ISBN 9780070145122. Disponível em: <https://books.google.com.br/books?id=n4IubCcpm0EC>.

CRUZES, D. S.; DYBÅ, T. Synthesizing evidence in software engineering research. In: SUCCI, G.; MORISIO, M.; NAGAPPAN, N. (Ed.). *Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ESEM 2010, 16-17 September 2010, Bolzano/Bozen, Italy*. ACM, 2010. Disponível em: <https://doi.org/10.1145/1852786.1852788>.

CRUZES, D. S.; DYBÅ, T. Recommended steps for thematic synthesis in software engineering. In: *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*. IEEE Computer Society, 2011. p. 275–284. Disponível em: <https://doi.org/10.1109/ESEM.2011.36>.

CRUZES, D. S.; DYBÅ, T. Research synthesis in software engineering: A tertiary study. *Inf. Softw. Technol.*, v. 53, n. 5, p. 440–455, 2011. Disponível em: <https://doi.org/10.1016/j.infsof.2011.01.004>.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. *Introdução ao teste de software*. 4ª. ed. Rio de Janeiro, RJ, Brasil: Elsevier, 2007. ISBN 978-85-352-2634-8.

DO, Q. C. D.; YANG, G.; CHE, M.; HUI, D.; RIDGEWAY, J. Redroid: A regression test selection approach for android applications. In: GOU, J. (Ed.). *The 28th International Conference on Software Engineering and Knowledge Engineering, SEKE 2016, Redwood City, San Francisco Bay, USA, July 1-3, 2016*. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2016. p. 486–491. Disponível em: <https://doi.org/10.18293/SEKE2016-223>.

ENGSTRÖM, E.; RUNESON, P.; SKOGLUND, M. A systematic review on regression test selection techniques. *Information and Software Technology*, v. 52, n. 1, p. 14 – 30, 2010. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584909001219>.

GLAUBER, N. *Dominando o Android com Kotlin*. NOVATEC, 2019. ISBN 9788575227268. Disponível em: <https://books.google.com.br/books?id=JEeMDwAAQBAJ>.

GÓMEZ, M.; ROUYVOY, R.; ADAMS, B.; SEINTURIER, L. Mining test repositories for automatic detection of UI performance regressions in android apps. In: KIM, M.; ROBES, R.; BIRD, C. (Ed.). *Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016, Austin, TX, USA, May 14-22, 2016*. ACM, 2016. p. 13–24. Disponível em: <https://doi.org/10.1145/2901739.2901747>.

GRAVES, T. L.; HARROLD, M. J.; KIM, J.-M.; PORTER, A.; ROTHERMEL, G. An empirical study of regression test selection techniques. *ACM Trans. Softw. Eng. Methodol.*, ACM, New York, NY, USA, v. 10, n. 2, p. 184–208, abr. 2001. ISSN 1049-331X. Disponível em: <http://doi.acm.org/10.1145/367008.367020>.

GRØNLI, T.; GHINEA, G. Meeting quality standards for mobile application development in businesses: A framework for cross-platform testing. In: BUI, T. X.; JR., R. H. S. (Ed.). *49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016*. IEEE Computer Society, 2016. p. 5711–5720. Disponível em: <https://doi.org/10.1109/HICSS.2016.706>.

GS.STATCOUNTER. *Distribuição mundial do uso de sistemas operacionais para aplicativos*. 2021. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/worldwide>.

HIRAMA, K. *Engenharia de Software: Qualidade e Produtividade com Tecnologia*. 1st. ed. Rio de Janeiro, RJ, Brasil: Elsevier Editora Ltda., 2011. ISBN 978-85-352-4882-1.

HODA, R. Socio-technical grounded theory for software engineering. *CoRR*, abs/2103.14235, 2021. Disponível em: <https://arxiv.org/abs/2103.14235>.

HOVE, S. E.; ANDA, B. Experiences from conducting semi-structured interviews in empirical software engineering research. In: *11th IEEE International Symposium on Software Metrics (METRICS 2005), 19-22 September 2005, Como Italy*. IEEE Computer Society, 2005. p. 23. Disponível em: <https://doi.org/10.1109/METRICS.2005.24>.

HU, Y.; NEAMTIU, I. VALERA: an effective and efficient record-and-replay tool for android. In: *Proceedings of the International Conference on Mobile Software Engineering and Systems, MOBILESoft '16, Austin, Texas, USA, May 14-22, 2016*. ACM, 2016. p. 285–286. Disponível em: <https://doi.org/10.1145/2897073.2897712>.

IEEE. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, p. 1–84, Dec 1990.

IEEE. IEEE Standard for Software Maintenance. *IEEE Std 1219-1998*, p. 1–56, 1998.

JHA, A. K.; KIM, D. Y.; LEE, W. J. A framework for testing android apps by reusing test cases. In: TILEVICH, E. (Ed.). *Proceedings of the 6th International Conference on Mobile Software Engineering and Systems, MOBILESoft@ICSE 2019, Montreal, QC, Canada, May 25, 2019*. IEEE / ACM, 2019. p. 20–24. Disponível em: <https://doi.org/10.1109/MOBILESoft.2019.00012>.

JIANG, B.; WU, Y.; ZHANG, Y.; ZHANG, Z.; CHAN, W. Retestdroid: Towards safer regression test selection for android application. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. New York, USA: IEEE, 2018. v. 01, p. 235–244.

JURAN, J.; GRZYNA, F. *Quality Planning and Analysis: From Product Development Through Usage*. McGraw-Hill, 1970. ISBN 9780070331716. Disponível em: <https://books.google.com.br/books?id=xgqTnQAACAAJ>.

KASUNIC, M. *Designing an Effective Survey*. Pittsburgh, PA, USA, 2005. CMU/SEI-2005-HB-004.

KAZMI, R.; JAWAWI, D.; MOHAMAD, R.; GHANI, I. Effective regression test case selection: A systematic literature review. *ACM Computing Surveys*, v. 50, p. 1–32, 05 2017.

KITCHENHAM, B. A.; PFLEEGER, S. L. Principles of survey research part 2: designing a survey. *ACM SIGSOFT Softw. Eng. Notes*, v. 27, n. 1, p. 18–20, 2002. Disponível em: <https://doi.org/10.1145/566493.566495>.

KOCHHAR, P. S.; THUNG, F.; NAGAPPAN, N.; ZIMMERMANN, T.; LO, D. Understanding the test automation culture of app developers. In: *8th IEEE International Conference on Software Testing, Verification and Validation, ICST 2015, Graz, Austria, April 13-17, 2015*. IEEE Computer Society, 2015. p. 1–10. Disponível em: <https://doi.org/10.1109/ICST.2015.7102609>.

KONG, P.; LI, L.; GAO, J.; LIU, K.; BISSYANDÉ, T. F.; KLEIN, J. Automated testing of android apps: A systematic literature review. *IEEE Trans. Reliab.*, v. 68, n. 1, p. 45–66, 2019. Disponível em: <https://doi.org/10.1109/TR.2018.2865733>.

LEUNG, H. K. N.; WHITE, L. J. Insights into regression testing [software testing]. In: *Proceedings of the Conference on Software Maintenance, ICSM 1989, Miami, FL, USA, 16-19 October, 1989*. IEEE, 1989. p. 60–69. Disponível em: <https://doi.org/10.1109/ICSM.1989.65194>.

LEUNG, H. K. N.; WHITE, L. J. A cost model to compare regression test strategies. In: *Proceedings of the Conference on Software Maintenance, ICSM 1991, Sorrento, Italy, 15-17 October 1991*. IEEE, 1991. p. 201–208. Disponível em: <https://doi.org/10.1109/ICSM.1991.160330>.

LI, X.; CHANG, N.; WANG, Y.; HUANG, H.; PEI, Y.; WANG, L.; LI, X. ATOM: automatic maintenance of GUI test scripts for evolving mobile applications. In: *2017 IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, Tokyo, Japan, March 13-17, 2017*. IEEE Computer Society, 2017. p. 161–171. Disponível em: <https://doi.org/10.1109/ICST.2017.22>.

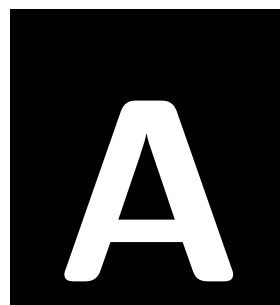
- LIMA, S. M. O.; CAMPOS, D.; SOARES, L. R.; MACHADO, I. Unveiling practitioners awareness of android apps regression testing through an expert survey. In: CAVALCANTE, E.; DANTAS, F.; BATISTA, T. (Ed.). *SBES '20: 34th Brazilian Symposium on Software Engineering, Natal, Brazil, October 19-23, 2020*. ACM, 2020. p. 303–308. Disponível em: <https://doi.org/10.1145/3422392.3422470>.
- MENS, T.; DEMEYER, S. (Ed.). *Software Evolution*. Springer, 2008. ISBN 978-3-540-76439-7. Disponível em: <https://doi.org/10.1007/978-3-540-76440-3>.
- MILLAR, D.; MCNAUGHTON, D.; LIGHT, J. A comparison of accuracy and rate of transcription by adults with learning disabilities using a continuous speech recognition system and a traditional computer keyboard. *Journal of Postsecondary Education and Disability*, 01 2005.
- PFLEEGER, S. L.; KITCHENHAM, B. A. Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Softw. Eng. Notes*, v. 26, n. 6, p. 16–18, 2001. Disponível em: <https://doi.org/10.1145/505532.505535>.
- PRATES, L. C. L. *Aplicando Síntese Temática em Engenharia de Software*. Dissertação (Mestrado) — Universidade Federal da Bahia, Salvador, 2015.
- ROMANO, S.; SCANNIELLO, G.; ANTONIOL, G.; MARCHETTO, A. Spiritus: a simple information retrieval regression test selection approach. *Information and Software Technology*, v. 99, p. 62 – 80, 2018. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584918300405>.
- ROTHERMEL, G.; HARROLD, M. J. Analyzing regression test selection techniques. *IEEE Trans. Software Eng.*, v. 22, n. 8, p. 529–551, 1996. Disponível em: <https://doi.org/10.1109/32.536955>.
- ROTHERMEL, G.; UNTCH, R. H.; CHU, C.; HARROLD, M. J. Prioritizing test cases for regression testing. *IEEE Trans. Software Eng.*, v. 27, n. 10, p. 929–948, 2001. Disponível em: <https://doi.org/10.1109/32.962562>.
- SACHTLEBEN, R.; PELESKA, J. Effective grey-box testing with partial FSM models. *CoRR*, abs/2106.14284, 2021. Disponível em: <https://arxiv.org/abs/2106.14284>.
- SCHMAUCH, C. H. *ISO 9000 for Software Developers*. 2nd. ed. USA: ASQ Quality Press, 1995. ISBN 0873893484.
- SILVEIRA-NETO, P. A. d. M.; MACHADO, I. C.; CAVALCANTI, Y. C.; ALMEIDA, E. S.; GARCIA, V. C.; MEIRA, S. R. L. A regression testing approach for software product lines architectures. In: *2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse*. IEEE Computer Society, 2010. p. 41–50. Disponível em: <https://doi.org/10.1109/SBCARS.2010.14>.
- SOMMERVILLE, I. *Software Engineering*. 10th. ed. England: Pearson Education, 2016.

SZABÓ, C.; SAMUELIS, L.; IVANOVIC, M.; FESIC, T. Database refactoring and regression testing of android mobile applications. In: *10th IEEE Jubilee International Symposium on Intelligent Systems and Informatics, SISY 2012, Subotica, Serbia, September 20-22, 2012*. IEEE, 2012. p. 135–139. Disponível em: <https://doi.org/10.1109/SISY.2012.6339502>.

VÁSQUEZ, M. L.; BERNAL-CÁRDENAS, C.; MORAN, K.; POSHYVANYK, D. How do developers test android applications? In: *2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, Shanghai, China, September 17-22, 2017*. IEEE Computer Society, 2017. p. 613–622. Disponível em: <https://doi.org/10.1109/ICSME.2017.47>.

WONG, W. E.; HORGAN, J. R.; LONDON, S.; MATHUR, A. P. Effect of test set minimization on fault detection effectiveness. In: PERRY, D. E.; JEFFERY, R.; NOTKIN, D. (Ed.). *17th International Conference on Software Engineering, Seattle, Washington, USA, April 23-30, 1995, Proceedings*. ACM, 1995. p. 41–50. Disponível em: <https://doi.org/10.1145/225014.225018>.

YOO, S.; HARMAN, M. Regression testing minimization, selection and prioritization: a survey. *Softw. Test. Verification Reliab.*, v. 22, n. 2, p. 67–120, 2012. Disponível em: <https://doi.org/10.1002/stv.430>.



QUESTIONÁRIO SURVEY

Prezado colaborador, este questionário destina-se a coletar dados para um trabalho acadêmico de mestrado em Ciência da Computação da Universidade Federal da Bahia (UFBA), com objetivo de investigar a aplicação de teste de regressão no desenvolvimento de Aplicativos para dispositivos móveis (APPS) Android. A sua participação é de fundamental importância.

A1 TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

- 1 O/A senhor(a) está sendo convidada(o) a participar voluntariamente da pesquisa sobre processo de teste de APPS Android.
- 2 Sua participação não é obrigatória.
- 3 A qualquer momento você pode desistir de participar e retirar seu consentimento.
- 4 Sua recusa não trará nenhum prejuízo em sua relação com a pesquisadora ou com a instituição.
- 5 Este formulário tem por objetivo investigar o processo de teste de APPS Android.
- 6 Sua participação neste formulário consistirá em responder questões objetivas e subjetivas.
- 7 Sua identificação é opcional, ou seja, você não precisa informar nome ou e-mail caso assim deseje.
- 8 A aplicação do formulário está sendo realizada por Sara Mendes Oliveira Lima, estudante da pós graduação em Ciência da Computação na Universidade Federal da Bahia, sob a supervisão do Prof. Dr. Ivan C. Machado.
- 9 Os benefícios relacionados à sua participação estão apenas em contribuir com a pesquisa científica. Será permitido acesso aos resultados desta pesquisa por meio da dissertação ou publicações científicas realizadas a partir desse estudo.

- 10 As informações pessoais obtidas através desta pesquisa serão confidenciais e não serão distribuídas ou divulgadas pela pesquisadora.
- 11 Os dados coletados neste formulário não serão divulgados de forma a possibilitar sua identificação.
- 12 Ao continuar respondendo este questionário, o/a senhor(a) concorda com as informações aqui descritas, porém a qualquer momento o/a senhor(a) poderá interromper a pesquisa sem ônus algum.
- 13 Este questionário utiliza o pacote de aplicativo Google Docs, portanto a coleta e o uso de informações do Google estão sujeitos à Política de privacidade do Google (<https://www.google.co.uk/policies/privacy/>).
- 14 Abaixo seguem os dados de contato dos responsáveis por esta pesquisa, com os quais você pode tirar suas dúvidas sobre sua participação.

Pesquisadores responsáveis:

Sara Mendes Oliveira Lima - lima.sara@ufba.br

Ivan C. Machado, Ph.D. - ivan.machado@ufba.br (supervisor).

Universidade Federal da Bahia (UFBA)

Instituto de Matemática e Estatística

Departamento de Ciência da Computação

Av. Adhemar de Barros, s/n, sala 280, Ondina, 40170-110, Salvador – BA

Salvador - Ba, Janeiro de 2020.

Declaro que entendi os objetivos, riscos e benefícios de minha participação na pesquisa.*

- **Concordo**

A2 PERFIL DO PARTICIPANTE

Essa sessão tem como objetivo identificar o perfil do respondente, tipo de trabalho realizado e aptidões técnicas.

1 Gênero: *

(múltipla escolha)

- Homem
- Mulher
- Outros

2 Idade: *

(questão subjetiva)

3 Nível de Escolaridade: *

(múltipla escolha)

- Nível médio completo
 - Curso Técnico completo
 - Graduação completa
 - Especialização completa
 - Mestrado completo
 - Doutorado completo
 - Pós-Doutorado
 - Outros
- 4 **Área de Formação Acadêmica:** *
(múltipla escolha)
- Informática, Computação ou áreas afins
 - Engenharia Elétrica
 - Engenharia de Produção
 - Outras Engenharias
 - Outros
- 5 **Você já fez algum curso / certificação na área de testes?** *
(múltipla escolha)
- Sim
 - Não
- 6 **Caso a pergunta acima tenha sido "Sim", quais foram esses cursos / certificações?**
(questão subjetiva)
- 7 **Em relação a projetos de APPS Android, você:** *
(caixa de seleção)
- Trabalha de forma autônoma
 - Trabalha / pesquisa em uma empresa
 - Estudante da área
 - Outros
- 8 **Estado onde trabalha / estuda:** *
(múltipla escolha)
- Acre
 - Alagoas
 - Amapá
 - Amazonas
 - Bahia
 - Ceará
 - Distrito Federal
 - Espírito Santo

- Goiás
- Maranhão
- Mato Grosso
- Mato Grosso do Sul
- Minas Gerais
- Pará
- Paraíba
- Paraná
- Pernambuco
- Piauí
- Rio de Janeiro
- Rio Grande do Norte
- Rio Grande do Sul
- Rondônia
- Roraima
- Santa Catarina
- São Paulo
- Sergipe
- Tocantins
- Outros

9 **Experiência profissional na área de TESTES DE SOFTWARE:**

*

(múltipla escolha)

- Até 1 ano
- De 1 a 3 anos
- De 3 a 5 anos
- De 5 a 10 anos
- Acima de 10 anos

10 **Experiência profissional na área específica de TESTES DE APPS ANDROID: ***

(múltipla escolha)

- Até 1 ano
- De 1 a 3 anos
- De 3 a 5 anos
- De 5 a 10 anos
- Acima de 10 anos

11 **Quando comparado a outros profissionais, como você considera o seu nível de conhecimento em TESTES DE APPS ANDROID?**

*

(múltipla escolha)

- Muito baixo
- Baixo
- Regular
- Bom
- Excelente

12 **Caso seja funcionário de uma empresa, qual a sua principal função no projeto atual?**

(questão subjetiva)

13 **Qual(is) atividade(s) realiza no processo de teste de software: ***

(caixa de seleção)

- Trabalha com criação / design de casos de teste
- Trabalha com execução de casos de teste
- Outro

A3 PROCESSO DE TESTES DE APPS ANDROID:

As perguntas a seguir referem-se ao processo de teste de APPS Android que você realiza. Utilize como base o comportamento encontrado no seu dia-a-dia de trabalho.

1 **Os testes realizados se enquadram em qual(is) categorias: ***

(caixa de seleção)

- Funcionais, também conhecido como black-box, quando não há acesso ao código fonte.
- Não-funcionais, também conhecido como white-box, quando há acesso ao código fonte.
- Grey-box, quando há acesso parcial ao código fonte.
- Outros

2 **Qual(is) tipos de testes a empresa realiza durante o processo de desenvolvimento APPS Android?**

(caixa de seleção)

- Teste de GUI
- Teste de Unidade
- Teste de Integração
- Teste de Validação
- Teste de Sistema
- Teste de Recuperação
- Teste de Proteção

- Teste de Estresse
- Teste de Desempenho
- Teste de Conectividade
- Teste de Segurança
- Teste em Condições Naturais
- Teste de Certificação
- Teste de Regressão
- Outros

3 Os testes são realizados de forma: *

(múltipla escolha)

- Manual
- Automatizada
- Ambas as formas

4 Quais ferramentas você costuma utilizar em seus projetos de APPS ANDROID? *

(caixa de seleção)

- APPIUM
- CALABASH
- ESPRESSO
- KATALON
- KMAX
- KOBITON
- MONKEY
- ROBOELETRIC
- ROBOTIUM
- SEE TEST
- TELERIK
- TESTINGBOT
- UI AUTOMATOR
- Não utilizo ferramentas
- Outros

5 Quando é realizada uma MANUTENÇÃO (ATUALIZAÇÃO) no app Android, quer seja perfectiva, corretiva, adaptativa ou preventiva, é realizado algum tipo de teste com o objetivo de garantir que as manutenções realizadas não alteraram o comportamento funcional do app? *

(múltipla escolha)

- Nunca

- Raramente
 - Às vezes
 - Muitas vezes
 - Sempre
- 6 **Caso, a resposta anterior seja positiva, o processo de teste durante a MANUTENÇÃO (ATUALIZAÇÃO) do app é feita de que forma? ***
(múltipla escolha)
- Manual, sem uso de ferramentas
 - Automatizada, com uso de ferramentas
 - Ambas
 - Não testo o aplicativo após MANUTENÇÃO (ATUALIZAÇÃO)
- 7 **Quais processos você realiza para testar a versão atualizada do app: ***
(caixa de seleção)
- Reexecuta todos os casos de teste da versão original para testar a versão atualizada do app
 - Seleciona os casos de teste referente as mudanças entre a versão original e a versão atualizada do app
 - Avalia se há a necessidade de criar novos casos de teste para a versão atualizada do app
 - Exclui casos de teste que não são relevadores de falhas para a versão atualizada do app
 - Reutiliza alguns casos de teste da versão original do app
 - Não testo o aplicativo após MANUTENÇÃO (ATUALIZAÇÃO)
 - Outros
- 8 **Quando realiza alguma MANUTENÇÃO (ATUALIZAÇÃO) utiliza alguma ferramenta para automatizar o teste da nova versão do APP ANDROID? ***
(múltipla escolha)
- Sim
 - Não
- 9 **Caso utilize, qual(is) são a(s) ferramenta(s)? ***
(caixa de seleção)
- ATOM
 - CHATEM
 - DETREDUCE
 - GUIDIFF
 - MONKEYRUNNER

- RANOREX
 - REDROID
 - RETESTDROID
 - SQUISH
 - TEMA
 - TESTCOMPLETE
 - UFT
 - Não utilizo ferramentas
 - Não testo o aplicativo após MANUTENÇÃO (ATUALIZAÇÃO)
 - Outros
- 10 **A(s) ferramenta(s) utilizada(s) atendem as necessidades para testar o app após ATUALIZAÇÃO (MANUTENÇÃO)? ***
(múltipla escolha)
- Nunca
 - Raramente
 - Às vezes
 - Muitas vezes
 - Sempre
 - Não testo o aplicativo após MANUTENÇÃO (ATUALIZAÇÃO)
- 11 **Justifique a resposta anterior: ***
(questão subjetiva)
- 12 **Considera relevante testar apps ao realizar algum tipo de MANUTENÇÃO (ATUALIZAÇÃO)? ***
(múltipla escolha)
- Sim
 - Não
- 13 **Qual(is) fator(es) você considera que fazem com que não sejam feitos testes em aplicativos após realização de MANUTENÇÃO (ATUALIZAÇÃO): ***
(caixa de seleção)
- Alto esforço da equipe de teste
 - Baixa importância na revelação de erros da nova versão
 - Falta de ferramentas eficientes para automatização desse processo
 - Tempo Reduzido para entrega da versão atualizada
 - Outros
- 14 **Na sua opinião, quais são as features / funcionalidades que as ferramentas de teste precisam implementar para melhorar / automatizar o processo de testes durante a manutenção de apps Android? Justifique sua resposta. *** (questão subjetiva)

15 **Você está satisfeito(a) com as features / funcionalidades oferecidas pelas ferramentas de testes de app Android existentes? ***
(múltipla escolha)

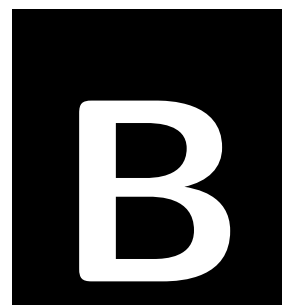
- Sim
- Não

16 **Justifique a sua resposta. ***
(questão subjetiva)

A4 CONSIDERAÇÕES FINAIS:

Agradecemos sua participação nessa pesquisa acadêmica. Ela é de fundamental importância para o estudo da área de testes de apps Android.

Caso queira receber os resultados desse survey, fineza informar o seu e-mail:
(questão subjetiva)



SURVEY: PERFIL DOS PARTICIPANTES

Tabela B.1 *Survey*: Perfil dos Participantes

P	Gên.	Idade	Formação*	Educação	UF	Exp.**	Função na Empresa
1	M	25	Informática	Graduação	SP	3–5	Desenvolvimento mobile com Android
2	M	23	Informática	Graduação	SP	1–3	Desenvolvedor
3	M	40	Informática	ESP	BA	<=1	—
4	F	20	Informática	Nível médio	PB	<=1	Arquit./escrita de testes automatizados
5	F	26	Informática	Graduação	BA	<=1	Consultora técnica de qualidade de software
6	M	48	Informática	Mestrado	BA	<=1	Programação
7	M	23	Informática	Graduação	BA	<=1	—
8	M	35	Informática	Mestrado	BA	<=1	—
9	M	31	Informática	Graduando	BA	5–10	QA
10	F	28	Informática	Graduação	BA	<=1	Analista de teste
11	M	28	Informática	Graduando	SE	<=1	Analista de sistema
12	F	28	Informática	Graduação	BA	<=1	Elaborar teste funcional automatizado
13	F	26	Informática	Graduação	PR	1–3	Gerente de Projetos
14	M	29	Informática	Doutorado	PB	<=1	—
15	M	36	Informática	Mestrado	BA	<=1	Analista de Teste
16	M	27	Informática	Graduação	PB	<=1	Analista de Teste
17	M	31	GE	Graduação	SE	<=1	Líder Arquitetura
18	F	40	Informática	Mestrado	PE	3–5	Engenheira de Teste Pleno
19	M	29	Informática	ESP	PE	3–5	Engenheiro de Testes
20	M	31	Informática	Graduando	BA	5–10	QA
21	M	25	Informática	Graduando	BA	3–5	—
22	M	22	Informática	Curso técnico	BA	<=1	—
23	F	30	Informática	Graduação	SP	<=1	Analista de Sistema Pleno
24	M	21	Informática	Nível médio	BA	1–3	Analista de Qualidade
25	M	23	Comunicação*	Nível médio	BA	1–3	—

Legenda: [P] Participante, (*) Área de Formação, (**) Experiência em testes de aplicativos para Android (em anos), [F] Feminino, [M] Masculino, [Comunicação*] Comunicação Social, [Eng. Computação] Engenharia da Computação, [Eng. Elétrica] Engenharia Elétrica, [Eng. Produção] Engenharia de Produção, [*Engenharias] Outras engenharias, [GE] Gestão Empresarial, [Informática] Informática, Ciência da Computação ou áreas afins, [ESP] Pós Graduação (Especialização), [Mestrado Inc*] Mestrado Incompleto

Tabela B.2 *Survey*: Perfil dos Participantes

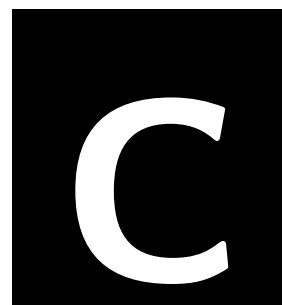
P	Gên.	Idade	Formação*	Educação	UF	Exp.**	Função na Empresa
26	M	23	Informática	Graduando	BA	3–5	Liderança / Execução de <i>checklist</i>
27	M	25	Informática	Nível médio	BA	1–3	Testes web
28	M	34	*Engenharias	Graduação	AM	1–3	<i>Homologation Tests</i>
29	M	28	Informática	Graduação	BA	1–3	Desenvolvedor <i>back-end</i>
30	M	39	Informática	Graduação	PA	<=1	Desenvolvedor
31	M	31	Informática	Graduação	SP	3–5	QA
32	F	29	Informática	ESP	BA	1–3	Analista de Teste
33	M	27	Informática	Graduação	BA	<= 1	—
34	M	25	*Engenharias	Graduando	ES	<=1	Desenvolvedor Júnior
35	M	33	Informática	ESP	BA	<=1	Analista de teste pleno
36	F	27	Informática	Graduação	BA	<=1	Analista de Qualidade Pleno
37	M	38	Informática	Graduação	DF	<=1	Tester funcional
38	M	26	Informática	Graduação	SP	<=1	Arquitetura de aplicações.
39	M	23	Informática	Curso técnico	SP	5–10	Desenvolvedor Android
40	M	29	Informática	Graduação	PE	3–5	Desenvolvedor Sr.
41	M	33	Informática	Graduação	33	MG	Desenvolvedor Android
42	M	32	Informática	Graduação	SP	<=1	Engenheiro Android
43	M	25	Informática	Graduação	SP	<=1	Desenvolvedor Android
44	F	25	Informática	Graduação	SP	1–3	Desenvolvedora Android
45	M	31	Informática	ESP	SP	<=1	—
46	M	27	Informática	Graduação	27	SP	1–3
47	M	24	Eng. Computação	Graduação	SP	1–3	Desenvolvedor Android Pleno
48	M	22	Informática	Graduação	SP	<=1	Desenvolvedor Android
49	F	27	Informática	ESP	MG	3–5	QA
50	M	26	Informática	Graduando	PA	<=1	—
51	M	35	Informática	Graduação	TO	<=1	Desenvolvedor
52	F	22	Informática	ESP	BA	1–3	Modelagem de testes
53	M	27	Informática	Graduação	BA	3–5	Analista de testes automatizados
54	F	28	Informática	Graduação	BA	<=1	QA, Testes web
55	M	28	Informática	Graduação	BA	<=1	Líder de qualidade
56	M	24	Eng. Elétrica	Graduação	BA	1–3	Testes manuais
57	M	26	Informática	Graduação	PE	3–5	Engenheiro de testes
58	M	33	Informática	Graduação	SP	5–10	Gerente Técnico
59	M	39	Informática	ESP	DF	1–3	QA
60	M	33	Informática	Graduação	MG	1–3	—
61	M	30	Informática	Graduação	MG	3–5	Automatizador
62	M	30	Eng. Elétrica	Graduação	BA	1–3	Líder de testes
63	M	39	Eng. Elétrica	ESP	PE	5–10	QA
64	M	28	Informática	Graduação	SP	<=1	Desenvolvedor
65	M	23	Eng. Produção	Nível médio	BA	<=1	Analista de Controle de Qualidade
66	F	33	Informática	Graduação	BA	<=1	Analista de Qualidade
67	M	28	Informática	Graduação	BA	<=1	Líder de qualidade
68	F	28	Eng. Produção	Graduação	BA	<=1	Analista de Qualidade
69	M	42	Informática	Mestrado	TO	1–3	—
70	M	28	Informática	Graduação	SP	<=1	QA <i>Senior</i>

Legenda: [P] Participante, (*) Área de Formação, (**) Experiência em testes de aplicativos para Android (em anos), [F] Feminino, [M] Masculino, [Comunicação*] Comunicação Social, [Eng. Computação] Engenharia da Computação, [Eng. Elétrica] Engenharia Elétrica, [Eng. Produção] Engenharia de Produção, [*Engenharias] Outras engenharias, [GE] Gestão Empresarial, [Informática] Informática, Ciência da Computação ou áreas afins, [ESP] Pós Graduação (Especialização), [Mestrado Inc*] Mestrado Incompleto

Tabela B.3 *Survey*: Perfil dos Participantes

P	Gên.	Idade	Formação*	Educação	UF	Exp.**	Função na Empresa
71	M	23	Informática	Nível médio	BA	3–5	Analista de Qualidade
72	M	28	Informática	Curso técnico	BA	1–3	—
73	M	25	*Engenharias	Nível médio	BA	3–5	—
74	M	24	Informática	Nível médio	BA	3–5	Analisar e definir os critérios de aceite
75	M	30	Informática	Graduação	SP	3–5	Analista de Testes Sr.
76	M	24	Informática	Graduação	SP	1–3	QA
77	M	28	Informática	Graduação	SP	3–5	<i>Embedded Quality Analyst</i>
78	M	26	Informática	ESP	SP	<=1	QA
79	M	27	Informática	Graduação	BA	3–5	Automatizador de testes
80	F	26	Informática	Graduação	SP	1–3	QA
81	M	27	Informática	Graduação	PR	<=1	Analista de teste/QA
82	M	23	Informática	Graduação	SP	1–3	Analista de qualidade
83	F	34	Informática	Mestrado Inc*	SP	1–3	QA
84	M	25	Informática	Nível médio	SP	<=1	—
85	M	26	Informática	Graduação	SP	1–3	Automatizador de testes Web e Mobile
86	M	35	Informática	Graduação	BA	<=1	Analista de teste
87	F	36	Informática	ESP	MG	<=1	—
88	M	31	Informática	Mestrado	Canadá	5–10	Desenvolvedor
89	M	34	Informática	Graduação	SP	5–10	Analista de qualidade Sr.
90	M	25	Informática	ESP	SP	<=1	—
91	F	35	*Engenharias	Graduação	MG	<=1	Validar requisitos/criar cenários de testes de API
92	M	25	Informática	Graduação	SP	5–10	Analista de Qualidade Sr
93	M	28	Informática	Curso técnico	MG	<=1	<i>Bug fixer, Refactoring, Analysis</i>
94	M	38	Informática	ESP	PB	3–5	QA Sr.
95	M	33	Informática	Graduação	BA	<=1	Analista Desenvolvedor Web
96	M	24	Informática	Graduação	ES	<=1	Manutenção em um aplicativo em produção.
97	M	36	Informática	Mestrado	CE	1–3	Automação de testes
98	M	26	Informática	Mestrado	SP	<=1	—
99	M	22	Informática	Curso técnico	ES	1–3	—
100	M	32	Informática	ESP	BA	5–10	Analista Sr.

Legenda: [P] Participante, (*) Área de Formação, (**) Experiência em testes de aplicativos para Android (em anos), [F] Feminino, [M] Masculino, [Comunicação*] Comunicação Social, [Eng. Computação] Engenharia da Computação, [Eng. Elétrica] Engenharia Elétrica, [Eng. Produção] Engenharia de Produção, [*Engenharias] Outras engenharias, [GE] Gestão Empresarial, [Informática] Informática, Ciência da Computação ou áreas afins, [ESP] Pós Graduação (Especialização), [Mestrado Inc*] Mestrado Incompleto



PROTOCOLO DA ENTREVISTA

O presente protocolo deverá ser apresentado no momento da entrevista através do “roteiro da entrevista”.

1. ORIENTAÇÕES PARA A ENTREVISTA:

- As perguntas das entrevistas deverão ser perguntas subjetivas;
- A entrevistadora deverá falar o mínimo possível;
- A entrevistadora não deverá emitir sua opinião;
- Deverá ser evitado ou minimizado viés;
- A entrevista poderá ser ajustada de acordo com o perfil do(a) entrevistado(a);
- Evitar realizar entrevista em locais barulhentos.

2. PRELIMINARES:

- Todos os possíveis entrevistados serão contatados por e-mail para formalizar o contato e agendar a entrevista;
- Caso o contato não possa ser feito por e-mail previamente, o entrevistador deverá coletar o e-mail do entrevistado no momento da entrevista;
- O protocolo de entrevista deverá ser apresentado no momento da entrevista através do “roteiro da entrevista”.

3. AQUECIMENTO DA ENTREVISTA:

- Explicar o objetivo geral da pesquisa em termos gerais;
- Explique o protocolo padrão e perguntar se está tudo bem (caso contrário, adapte-se às necessidades);
- Explicar os motivos de gravação e confidencialidade;

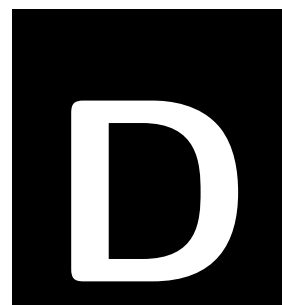
- Todos os resultados serão publicados de forma anônima.

4. DURANTE A ENTREVISTA:

- A entrevistadora poderá fazer algumas anotações em papel ou computador;
- O áudio completo será gravado;
- Caso seja possível, a entrevistadora poderá solicitar artefatos (documentos, códigos etc) para adicionar a análise da pesquisa.

5. APÓS A ENTREVISTA:

- A entrevistadora produzirá iterativamente um documento de trabalho (notas da pergunta de pesquisa) estruturado em torno das perguntas iniciais da pesquisa. Para cada questão de pesquisa, ela escreverá observações feitas pelos entrevistados sobre tópicos relacionados;
- A entrevistadora deverá pedir sugestões de empresas ou pessoas a serem entrevistadas;
- A entrevistadora deverá coletar informações pessoais e profissionais (tratadas com confidencialidade), através de formulário online, como: (i) Pessoais: nome, idade, nível de escolaridade mais alto completo e área; e, (ii) Profissionais: cargo atual, tempo de empresa, nome da empresa, URL, número de funcionários da empresa, número de funcionários do projeto (se necessário), há quanto tempo a empresa foi fundada (idade / meses), porte.



ROTEIRO DA ENTREVISTA

- **Apresentação:**

Bom Dia / Boa Tarde / Boa Noite. Me chamo Sara Lima. Obrigada por participar desta entrevista. Ela tem como objetivo compreender como profissionais que trabalham com testes de aplicativos para Android executam testes em cenário de evolução nos projetos em que atuam. Não existem respostas certas ou erradas, nem desejáveis ou indesejáveis. Sinta-se à vontade para dizer o que pensa.

- **Instruções para gravação:**

Reforçando o que foi definido no termo de consentimento, nossa conversa será gravada. O objetivo é para que eu possa obter todos os detalhes, e ao mesmo tempo ser capaz de manter uma conversa atenta com você. Seus comentários permanecerão confidenciais. O relatório final conterà os comentários dos participantes sem nenhuma referência aos indivíduos.

Perguntas da Entrevista:

1. **Como você começou a trabalhar com teste de aplicativos para Android?**
(Criar um vínculo inicial com o participante, assim como, descobrir o nível de experiência em teste de aplicativos para Android)
2. **Quais foram suas fontes de aprendizado sobre teste de aplicativos para Android?** (Empresa atual ou anteriores, Faculdade / Universidade, Sites, Cursos, Certificações)
3. **Qual é a linguagem (Quais são as linguagens) que você utiliza (ou a empresa utiliza) para desenvolver os aplicativos para Android no projeto atual?**
 - (a) Existe algum critério de escolha da linguagem? É a mesma linguagem para todos os projetos?

- (b) Linguagem nativa?
 - (c) Existem ferramentas para automação de testes para essa(s) linguagem(ens)?
4. **Quando você precisa entregar uma *release* de um aplicativo, qual o passo a passo para testar esta nova versão?** (Desejamos entender o passo a passo.)
- (a) Existe algum fluxograma, *template*, documento, tutorial que padronize esse processo? (Se existe, é possível ter acesso a uma cópia)
 - (b) Existem ferramentas específicas para automatização?
5. **Em relação a escolha dos casos de teste a serem reexecutados para testar uma nova release do aplicativo, é utilizado algum critério de seleção?**
- (a) Se sim, qual(is)?
 - (b) Esse processo é manual ou automatizado?
 - (c) Se não, reexecuta todos os casos de teste?
6. **O que você entende como teste de regressão?**
- (a) Caso o participante não saiba o que é, apresentar o seguinte conceito “O teste de regressão é o processo de testar novamente o software que foi modificado, para verificar se as modificações não causaram efeitos não-intencionais e se o software ainda está em conformidade com o requisito especificado”.
 - (b) Caso conheça o assunto:
 - i. Como você conheceu este assunto?
 - ii. Você possui alguma dúvida sobre o tema?
7. **O que uma ferramenta precisa ter para atender satisfatoriamente a demanda de executar com qualidade teste de regressão?** Usabilidade? Performance? Citar funcionalidades.
8. **Na sua opinião, automatizar (ou não) o processo de teste de regressão tem relação com a linguagem utilizada no desenvolvimento?**
9. **Qual a importância de sistematizar o processo de execução do teste de regressão?**
10. **Qual a relação entre a execução de teste de regressão e a qualidade da release de um aplicativo?** (Ou seja, executar teste de regressão auxilia no processo de garantia da qualidade da nova versão do aplicativo?)
11. **Você tem alguma pergunta para mim, ou gostaria de acrescentar alguma informação, ou, ainda tem alguma sugestão para melhorar nossa entrevista?** (OPCIONAL)



PESQUISA SOBRE TESTE EM CENÁRIO DE EVOLUÇÃO DE APLICATIVOS PARA ANDROID/FORMULÁRIO ONLINE

Prezado(a) colaborador(a), este questionário destina-se a colher dados para um trabalho acadêmico do mestrado em Ciência da Computação da Universidade Federal da Bahia (UFBA), com objetivo de compreender como profissionais que trabalham com testes de aplicativos para Android executam testes em cenário de evolução nos projetos em que atuam. A sua participação é de fundamental importância.

A1 TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

- 1 O/A senhor(a) está sendo convidado(a) a participar voluntariamente da pesquisa sobre processo de qualidade de código de teste de software.
- 2 Sua participação não é obrigatória.
- 3 A qualquer momento você pode desistir de participar e retirar seu consentimento.
- 4 Sua recusa não trará nenhum prejuízo em sua relação com o pesquisador ou com a instituição.
- 5 Esta entrevista tem por objetivo compreender como profissionais que trabalham com testes de aplicativos para Android executam testes em cenário de evolução nos projetos em que atuam, e sua participação consistirá em responder as perguntas da entrevista e um questionário com questões objetivas e de texto curto.
- 6 A sua participação nesta pesquisa pode envolver algum desconforto, ainda que pequeno, relacionado ao tempo despendido ao responder as perguntas da entrevista e preenchimento do questionário. Faremos o possível para minimizar possíveis desconfortos e não ocupar seu tempo desnecessariamente.

- 7 A entrevista será gravada. O objetivo disso é para que a pesquisadora possa obter todos os detalhes, mas ao mesmo tempo ser capaz de manter uma conversa atenta com você. Garantimos que todos os seus comentários permanecerão confidenciais. O relatório final conterá os comentários dos participantes sem nenhuma referência aos indivíduos.
- 8 Os benefícios relacionados à sua participação estão apenas em contribuir com a pesquisa científica. Será permitido acesso aos resultados desta pesquisa por meio da dissertação ou publicações científicas realizadas a partir desse estudo.
- 9 As informações pessoais obtidas através desta pesquisa serão confidenciais e não serão distribuídas ou divulgadas pelo pesquisador.
- 10 Os dados coletados neste formulário não serão divulgados de forma a possibilitar sua identificação.
- 11 Ao continuar respondendo este questionário, o/a senhor(a) concorda com as informações aqui descritas, porém a qualquer momento o/a senhor(a) poderá interromper a pesquisa sem ônus algum.
- 12 Este questionário utiliza o pacote de aplicativo Google Docs, portanto a coleta e o uso de informações do Google estão sujeitos à Política de privacidade do Google (<https://www.google.co.uk/policies/privacy/>).
- 13 Abaixo seguem os dados de contato dos responsáveis por esta pesquisa, com os quais você pode tirar suas dúvidas sobre sua participação:

Sara Mendes Oliveira Lima - lima.sara@ufba.br,
Larissa Rocha - larissars@dcc.ufba.br,
Ivan C. Machado, Ph.D. - ivan.machado@ufba.br (supervisor)
Universidade Federal da Bahia (UFBA)
Instituto de Matemática e Estatística - Departamento de Ciência da Computação
Av. Adhemar de Barros, s/n, sala 280, Ondina, 40170-110, Salvador – BA
Salvador, 22 de Janeiro de 2021.

Declaro que entendi os objetivos, riscos e benefícios de minha participação na pesquisa.*

- **Concordo**

(a) **PERFIL DO PARTICIPANTE**

Essa sessão tem como objetivo identificar o perfil do respondente, tipo de trabalho realizado e aptidões técnicas.

- 1 **E-mail:** *
(questão subjetiva)

2 Idade: *

(questão subjetiva)

3 Área de Formação Acadêmica *

(questão múltipla escolha)

- Informática, Computação ou áreas afins
- Engenharias (Elétrica, de Produção)
- Outros

4 Titulação Máxima: *

(questão múltipla escolha)

- Nível Médio
- Curso Técnico
- Graduação
- Pós Graduação - Especialização
- Mestrado
- Doutorado
- Pós-Doutorado

5 Tem curso/certificação na área de teste? *

(múltipla escolha)

- Sim
- Não

6 Experiência em teste de aplicativos para Android (em anos):

*

(questão subjetiva)

7 Estado em que trabalha: *

(múltipla escolha)

- Acre
- Alagoas
- Amapá
- Amazonas
- Bahia
- Ceará
- Distrito Federal
- Espírito Santo
- Goiás
- Maranhão

- Mato Grosso
- Mato Grosso do Sul
- Minas Gerais
- Pará
- Paraíba
- Paraná
- Pernambuco
- Piauí
- Rio de Janeiro
- Rio Grande do Norte
- Rio Grande do Sul
- Rondônia
- Roraima
- Santa Catarina
- São Paulo
- Sergipe
- Tocantins
- Outros

8 **Nome da empresa:** *
(questão subjetiva)

9 **Site da empresa:** *
(questão subjetiva)

10 **Número de funcionários:** *
(questão múltipla escolha)

- De 1 a 25
- De 26 a 50
- De 50 a 75
- De 75 a 100
- Acima de 100
- Não sabe informar

11 **Tempo que trabalha na empresa (em anos):** *
(questão subjetiva)

12 **Cargo atual na empresa:** *
(questão subjetiva)

13 Atividades que realiza na empresa: *
(questão subjetiva)



ENTREVISTA: PERFIL DOS PARTICIPANTES

Tabela F.1 Entrevistas: Perfil dos Participantes

#	Gênero	Idade	Formação*	Educação	Exp.**	UF	Função na Empresa
1	F	28	Informática	MBA	3	DF	Analista de Teste
2	F	32	Informática	Graduação	1–2	DF	Analista de Teste
3	M	27	Informática	Graduação	5	SP	Analista pleno
4	M	31	Informática	Graduação	< 1	SP	Engenheiro Android
5	M	24	Informática	Graduação	3	SP	Analista de Qualidade-III
6	M	31	Informática	ESP	2	SP	Desenvolvedor Android
7	M	23	Informática	Graduação	1–2	SP	Analista de Qualidade-III
8	M	28	Informática	ESP	< 1	RJ	Analista de Automação Sênior
9	M	33	Informática	ESP	5	SP	Analista de Qualidade
10	M	34	Informática	ESP	1	MG	Desenvolvedor Android
11	M	28	Informática	Curso Técnico	2	AM	Analista de Teste
12	M	30	Informática	Graduação	3	MG	Engenheiro de Qualidade de Software
13	M	39	Matemática	Nível Médio	3	SP	Desenvolvedor Android
14	M	30	Informática	Graduação	1	MG	Desenvolvedor Android Sênior
15	M	40	Metereologia	Mestrado	3	SP	Analista de Qualidade
16	M	36	Informática	ESP	2	GO	Desenvolvedor Android

Legenda:

[P] - Participante / [F] - Feminino / [M] - Masculino

[Informática] - Informática, Ciência da Computação ou áreas afins / [ESP] - Pós Graduação (Especialização)

(*) Área de Formação / (**) Experiência em testes de aplicativos para Android (em anos)



TRANSCRIÇÃO DAS ENTREVISTAS

1. PARTICIPANTE 1:

Pergunta 1: Então, a empresa tinha uma equipe que fazia os aplicativos, e tinha um uma estagiária para essa equipe. Porém, a equipe não tinha um processo estabelecido e tinha vários gerentes, onde tinha os analistas, os testers, os analistas de testes alocado em cada gerência. Então, houve a necessidade de padronizar o tipo de teste e criou uma gerência só. Aí a gerência dos aplicativos mobile, eles não quiseram disponibilizar a estagiário deles, aí eles pegaram, ficaram com ela e tipo, transformou ela em analista de requisito. E por, devido a essa ausência, eu me candidatei para ficar na vaga dela. Aí foi isso. Não teve, tipo, tipo, foi uma questão de oportunidade.

Pergunta 2: Então, eu tive que procurar conhecimento por conta. Foi, fiz bastante pesquisa na internet, para descobrir ferramentas, o que é que eu precisava, como que era, fora o apoio da equipe, em si, da gerência do mobile.

Pergunta 3: São as nativas. No caso, Java e Kotlin. **Existe alguma ferramenta que vocês utilizam para automatização dos testes?** Katalon, a gente está usando Katalon.

Pergunta 4: A gente usa, a gente definiu um processo na equipe, para todas as gerências, onde o desenvolvedor, ele vai ter que abrir um chamado, uma solicitação de teste, e vai ter que disponibilizar um ambiente de homologação, onde eles não podem alterar durante os testes que a gente tá fazendo. E durante esse período eu vou reportando os defeitos e eles só podem gerar uma correção nova, numa correção, depois que eu testar toda aquela versão, corrigir aquela versão e depois de corrigida ele realiza, retorna pra gente. Depois que eu der o OK, aí eu faço a entrega do

termo de aceite de teste, depois o aceite de teste, no caso o produto pode ir para a produção direto. O processo é automatizado ou manual: Então, automatizada não é, mas a gente usa uma ferramenta que padroniza todas a maioria dos chamados, da empresa. Eu vou tentar lembrar até o final da reunião. **No caso, é uma ferramenta da empresa?** Isso.

Pergunta 5: Por questão de boas práticas, eu re-executo novamente, porque, às vezes, pode acrescentar alguma coisa durante as correções. Mas o foco principal é re-executar as que deram errado na primeira. Mas executo por questão de boas práticas mesmo, passar um pente fino. **Esse processo de reexecução, ele é manual ou é automatizado?** Os dois.

Pergunta 6: É quando eu vou, eu estou confundindo teste de regressão com teste de confirmação. O de confirmação é quando eu valido se aquele defeito foi corrigido. O de regressão é quando eu passo por todo ele. Pra ver se não ficou nenhum resíduo, nenhum defeito escondido ali. Ou foi acrescentado durante as correções dos outros defeitos. **Você tem alguma dúvida sobre esse tema de teste regressão ou pra você um conceito bem, bem tranquilo assim?** É tranquilo.

Pergunta 7: Eu acho que deveria ser questão mais de usabilidade. Algo que tipo, agora, por exemplo, eu estou de licença maternidade. Uma ferramenta que seria tão fácil pra, que alguém que possa me substituir, ela conseguisse executar as coisas que eu já fazia. E às vezes por não estar lá, algumas coisas podem não estar, não estar acontecendo.

Pergunta 8: Sim, sim, acredito que facilita.

Pergunta 9: Então, não existe o teste cem por cento, mas eu acho que quanto mais definido estiver o processo, digamos que a gente consegue cobrir um pouco mais o cenário. Os cenários possíveis e alguns fluxos de exceção. Eu acho importante por isso.

Pergunta 10: Sim, sim. Eu acredito sim. Que dá uma uma reforçada na qualidade sim.

Pergunta 11: Não não, estou só acompanhando mesmo. **Você achou as perguntas, assim, pertinentes com relação ao tema, que é teste de regressão?** Sim, pelo que eu entendi vocês estão mais a procura de mobile, Android e teste de regressão. E se está respeitando a literatura do teste. Os termos foram bem coerentes. Ficou muito bom. Parabéns.

2. PARTICIPANTE 2:

Pergunta 1: Então, eu prestei serviço para o DETRAN, de dois mil e dezoito até setembro do ano passado e houve-se a necessidade, de que tudo podia, boa parte de alguns sistemas, alguns sistemas que o pessoal que a gente, alguma, alguns temas específicos, virassem Android, inclusive para facilitar fiscalização, remoção de veículos, alguns processos de habilitação, de serviços relacionados a veículos. Aí a partir da necessidade do DETRAN, a gente entrou nesse universo de API/Mobile porque são interligados, porque são serviços que passam por sistemas distintos e são migrados e enviados de para outros órgãos. Aí por isso a necessidade de API/Mobile e com isso se eu não me engano foram seis sistemas, eu trabalhei em seis aplicativos, em um ano e seis meses. É porque é tudo tão automático e assim, era meio automático e muito, tá, vamos fazer, e muito, eu pelo menos não tinha muito conhecimento sobre o assunto, boa parte do meu time também não, mas acabou que quatro deles, são aplicativos que pelo menos o que eu ouço dizer que funcionam até hoje. Porque não é algo que se use na rua, mas um deles inclusive é específico para depósito, onde o veículo fica locado, depois que ele é removido da rua por uma série de fatores. Bem, vida de aplicativo. Vida de analista de teste de fábrica. O que vier você faz e faz pra ontem. E dá um jeito de aprender. Porque eu falo porque eu nem sabia o que era API. O Mobile você já tem noção. Porque todo mundo tem um celular, um celular cheio de coisa. O caramba a quatro, mas, a API, que é a base do serviço, para que o mobile exista, e eu só fui entender isso depois, aplicando mobile em si, assim, não existe mobile sem ter um serviço pra fazer a coisa funcionar, necessidade que tem do mobile geralmente é um serviço que precisa ser melhorado, mais atendido, expandido. É a minha visão. De quem sofreu um pouquinho pra adaptar e aprender. Meio na marra. Para falar a verdade.

Pergunta 2: Muito Google. E aí, foi. Eu não tenho, eu não me lembro exatamente o que agora, que eu já tenho o que, quase cinco meses que eu não estou trabalhando diretamente com o mobile. Aí, tanto é que, assim, quando no período de pesquisa, de estudo, de aplicação, eu tinha documentos no Word, tudo virava favorito. Sempre que eu achava alguma coisa interessante, isso virava favoritos, informação concisa que muitas das vezes eu tinha que validar com o próprio desenvolvedor para ver se estava de acordo, porque era a pessoa próxima que tinha, aquele conhecimento. E quando eu discordava, achava que o cara estava tentando me passar para trás, eu achava que podia ficar melhor, achava que tinha um, porque a gente podia está de informação, daí eu buscava de colegas que também trabalhavam na área, que tinha um pouco mais de conhecimento que eu, que estava no mesmo barco de aprender e no passado, no final de dois mil e dezenove, a BSTQB lançou um, uma certificação mobile a CTFL MAT. Com base nela, eu já consegui, já visualizar mais coisas, tipo assim, coisas que fazia no automático do tipo a qualidade, tá tipo tela, resolução bonitinho no quatro G, no dois G, paro três G. Uma coisa que eu fazia instintivamente, automaticamente, mas que eu nunca tinha me atentado. E depois

de ver o material da certificação eu consegui, acho que até ser mais crítica e mais concisa no que eu estava fazendo. Tanto é que eu virei a insuportável do meu time. Isso antes de sair, tipo, foi, a certificação saiu em dezembro. Eu descobri o material no final de janeiro e em março eu já tava aplicando aquilo que tinha nela, porque eu peguei a certificação e comecei a pesquisar no Google mesmo o que podia ser, o que podia não ser, os impactos negativos disso, o que podia fazer pra melhorar e virei a chata, a brigona, a tudo, mas, graças a Deus eu entreguei tudo, antes de sair de lá. Mas assim, basicamente: Google, colegas de time mesmo, que já tinha um pessoal que já tinham passado o mesmo que eu. Tinham quatro desenvolvedores mobile na minha equipe mais antigos e tinham dois mais juniores como eu, mas a gente se virava muito bem. Necessidade, que é o que faz a gente na TI, como você deve saber bem. E depois entrou a certificação mobile do BSTQB, que também virado com o Google e também tem um grupo de certificação do WhatsApp. Que a gente geralmente ensina um pro outro. Quando surge a necessidade, você pergunta. Algumas dúvidas também foram sanadas lá, mas praticamente foi tudo o Google, até porque o horário de estudo da gente é louco. Tem um tempo de uma demanda, de madrugada, de dia, fim de semana, não tem essa, e nem sempre você vai ter auxílio. Então, você vai ter que se virar com o recurso que você tem.

Pergunta 3: Na verdade o cliente nos deixava solto, até porque o cliente também acho que ele não tinha dimensão. Hoje eu entendo, eu estou em uma demanda atual do meu projeto que, principalmente, no que diz respeito a API/Mobile, o usuário, ele não tem a dimensão do poder que a responsabilidade dele na mão, tanto é que eu falo que se ele soubesse ele interferia na tecnologia que a gente vai usar, nas escolhas das pessoas nos times, porque isso tudo interfere, entendeu? E assim, tinha nativo, tinha o híbrido, que eu nem lembro, porque era o, se eu não me engano híbrido, é o que mistura o Java e uma parte de uma linguagem do Android, que eu não lembro agora porque tem, são duas. Mas, boa parte deles, eles são híbridos, porque você tem que, você é alimentado, você alimenta alguém que já vem de alguma outra linguagem. Provavelmente, geralmente é o Java? Boa parte do serviço hoje são em Java. Então, era híbrido, quase tudo. Eu não me lembro, teve um ou outro, mas tanto é que nem funcionava direito, chegou um tempo que ele ficou meio que obsoleto, o pessoal tipo, tá, vamos ter que abrir uma demanda para refazer isso aqui porque não vai atender, justamente porque foi feito nativo. Ele só atendia o Android, na época que precisou implementar como outros serviços, disponibilizar serviços para outras empresas, já não estava funcionando. Olha, eu lembro do React e eu não me lembro do nome da outra agora, mas o React é certeza que tinha. **Usavam alguma ferramenta para automatizar os testes?** Menina, eu fiz uma parte, eu tentei, fiz, alguns eu consegui usar o Selenium, mas porque a minha equipe era muito reduzida. Duas pessoas, para três aplicativos em andamento ao mesmo tempo, nunca dá tempo de fazer tudo. Aí o que é que eu fazia? E eu participava desde a da participação da elaboração dos requisitos, revisão por pares, mais execução de teste, mais documentação para homologação. Então

chegou uma hora que eu já tinha meio que decorado tudo porque eu já me aproximei de gente que sabia e eu me dedicava muito porque eu acho, não é que é fácil, mas o mobile ele tem, ele faz com que pessoas te vejam um pouco mais? Então, até ele te motiva a estudar mais, eu nunca, por mais que, eu lembro que era muito, muito puxado, era estressante, rolava muito atrito entre os times, mas sempre dava certo, porque o usuário sempre ficava feliz, porque ele tinha algo de qualidade na mão. Aí acabava que tipo pelo fato de não ter, não, não ter tempo, não ter time e dar bons resultados, eu, infelizmente, não conseguia automatizar tudo. Tanto é que, tipo assim, eu tinha automação de alguns logins, alguns sei lá, alguns logins, a parte dos menus, e como tinha muitas funcionalidades dentro, aí a gente tinha que deixar de menu, de submenu, perfis, aquela coisa toda. Aí, assim, o seu foi, login e a questão dos perfis, porque tinham, sei lá, tinha demanda, que tinha cinco perfis. Desde usuário comum a uma pessoa que podia fazer tudo, mandar e desmandar. Que podia bloquear ações superpoderosas nesse sistema. Mas enfim, boa parte foi manual, tanto é que na época eu quase peguei um, eu esqueci até o nome, mas de uma doença nos dedos, de tanto digitar. E por causa de tempo. Porque era muito mais prático pegar o até um emulador, era muito mais fácil testar três aparelhos, dois celulares e um tablet, do que um emulador. O tempo era muito curto. E é uma coisa que assim, por isso que eu até cobro muito do Adam o curso de Appium por isso, entendeu, um curso que tenha bastante Appium, porque eu fico, eu não posso ficar presa, porque eu fiquei um ano e meio trabalhando com Mobile. Eu preciso realmente aprender o Mobile do início ao fim, com todas as ferramentas, o máximo de ferramentas e práticas para que ele fique o melhor possível e o mais prático para gente, para eu tester, entregar o melhor sistema para o usuário. É uma visão minha de analista atual. Que até o fim do ano eu vou estar com essa expertise comigo.

Pergunta 4: Não tinha, a gente tinha um fluxograma mais ou menos de teste, que era até baseado no ATEL, eu fiz até com meu chefe, e eu, com base nesse fluxograma, eu fazia um documento de evidências com as principais funcionalidades ou com o caso do usuário. E em paralelo a isso, tipo assim, eu estou aqui terminando, eu disponibilizava para o usuário e já ficava, olha, qualquer coisa me chama. Porque pode ter detalhes que a gente não tenha visto direito, pode ser que alguma tela que a resolução não está de acordo, enfim, algum erro de português, mas assim, um trabalho colaborativo, que é uma coisa muito importante no mobile, mas que nas outras demandas. Justamente porque as tecnologias elas nos mudam muito rápido, assim, principalmente em questão de tela. No meu aparelho pode dar uma coisa, no seu, pode não dar. Eu não me lembro de ter entregado, de ter feito entrega, se sei lá. Eu vi um erro um dia desses, cara, gente, a pessoa deve estar se culpando muito. O alterar, voltava o código fonte da aplicação. Que acontece muito, alterar, excluir os botões dessas ações decisivas, o adicionar não tem muito isso, mas o alterar, o excluir, como ele já envolve informação que já existe para fazer uma ação com ela, ela é mais puxada, o upload de arquivos também, não é um, não, tipo, eu aprendi isso na raça, porque não podia vacilar, até porque nos sistemas web que a gente

tinha já acontecia muito isso. Eu, não vou deixar um negócio na mão do usuário que ele não vai conseguir usar. Era um software SKILL, eu não posso fazer isso com meu usuário. E é uma tipo ao mesmo tempo dedicação, empenho, parceria com o usuário, de mostrar: olha, eu dei o meu melhor, fiz o máximo que eu pude, mas eu posso ter falhado. E o que você puder, o que surgir de dúvida, o que aparecer, se por acaso estiver errado, por favor, nos informe que a gente vai resolver o quanto antes. Eu falo que, assim, no quesito, porque o Mobile e a API, a API não tanto, porque a gente, já sempre houve serviços, mas o mobile envolve tecnologia nova, gente inexperiente e um usuário com muita expectativa, ou seja, se não houver colaboração entre os envolvidos, a chance de desgaste, desperdício de dinheiro, tempo, tudo envolvido é muito maior do que em qualquer demanda web. É a minha visão hoje de quem já se frustrou muito, passou por muitos projetos complicados, enfim. O Mobile ele é lindo, maravilhoso, ele geralmente vai dar certo, porque as pessoas se empenham em deixar ele bom, principalmente deixar ele bonito, nem tudo tá bom, mas tem que tá com a cara bonita. Parece que o usuário, ele, infelizmente, ele não tem noção do poder que ele pode exercer com o aplicativo, e ele acha que tá bonito, tá legal. E não é o suficiente, porque tem muito aplicativo que é horrível, mas funciona muito bem tá totalmente desatualizado, mas não para, não te deixa na mão, não vou lembrar de nenhum citado agora. E tem muita coisa que tem um layout maravilhoso, e não tem essa qualidade, que é uma coisa que a gente quer de teste e ver entende melhor? O que pode, o que é que funciona direito, que funcionalidades podem ser agregadas. Assim, mas é aquilo que eu falo, é a minha visão de analista de quem passou um bom tempo com isso, aprendeu na raça. **Ferramenta específica para retestar aplicativo em cenário de evolução?** Sara, infelizmente, quase tudo era manual. Meu chefe ele fez um reteste de algumas, só que começou a dar problema com a versão do emulador, alguma coisa do tipo. Alguma coisa entre o emulador e o aplicativo que era. Aí tipo, pra não perder tempo, porque a gente tinha essa essa confiança do usuário de falar, olha, a gente pode ter errado, mas a gente vai deixar ok antes de disponibilizar, de tudo tá pronto. Então, era muito mais vantajoso pra gente, porque corria-se o risco de gastar duas, três semanas, que nem já aconteceu e perder o tempo, e ter que tentar fazer um negócio correndo em um fim de semana para homologar. Que nem a gente já teve que fazer. Por praticidade, eu assim, por uma economia, eu falo uma economia da empresa e também, porque aqui em Brasília, ainda é um pouco escasso, a galera do mobile, está chegando agora. E muita gente, porque eu, pelo menos, fui essa pessoa, a pessoa que se apegava ao projeto. Às vezes, é tipo? Não é o meu caso, mas muita gente nem ganhava tudo isso, mas ficava, ah, não, tá aqui, tá massa, tô aprendendo muito. Então, assim, tem essa dificuldade também, é de profissional. Por isso que eu, eu nem hoje em dia, eu não culpo tanto, a automação e assim, não vislumbro tanto ela, eu acho massa, vejo que a gente tem necessidade de aprender, mas se o, é muito mais interessante uma pessoa empenhada em fazer o sistema funcionar, do que você pagar um analista de automação caro. Porque pode ser que não dê certo, porque às vezes as empresas não tem nem máquinas suficientes para aguentar as ferramentas que tem para automação. É uma coisa que eu vivi muito.

Que eu já peguei, participei de três projetos, que, nossa, chegou aquele vislumbre da automação, e deu três meses, não tava saindo dando resultado, o cliente deu pra trás e mandou cancelar aqui. Porque o cliente ele quer pressa, ele acha que o teste, e o pessoal, infelizmente, tem uma visão que o teste, qualquer um faz. Aí até um certo receio, que assim, eu estou aprendendo a me jogar nas demandas, agora o mobile ele me fez o assim, crescer os olhos, e assim, me dá, mas até autonomia, de falar, olha, eu posso tentar fazer isso melhor, eu posso pedir ajuda, posso conversar com mais alguém, alguém que tenha mais expertise no negócio, tudo mais, e tentar automatizar. E vocês me dão essa autonomia, me liberam duas horas por dia, mas antes eu não tinha isso. Eu acho que veio a expertise de trabalhar com mobile, e também o fato de está um pouquinho mais experiente na área, que é uma coisa que o júnior, não tem, ainda não tem, entendeu? E o que é que o pessoal faz? Eles pegam o júnior, e quer que o cara aprenda o Selenium, e acha que o Selenium vai ser suficiente para automatizar a vida inteira. E não vai ser. O que você vai ter só um regressivo. E boa parte das coisas acontecem é nas fases iniciais do projeto. Porque assim, até você deixar um regressivo legal demora um pouco? É a minha visão. Não sou contra a automação, mas como eu não consegui até hoje implementá-la por recursos de máquina, pessoas e clientes que acha que qualquer um faz, que a gente faz rápido, que a gente tá demorando porque quer, que bota a culpa na gente, na gente empresa, e acaba que deixa passar, que é uma coisa que impactou negativamente na questão do meu mobile, eu falo que se eu tivesse, se a gente tivesse um time pouquinho maior, com gente de expertise de automação mobile teria sido melhor. A gente iria ganhar muito mais tempo, tinha evitado o desgaste entre os times atoa aquela coisa toda. Mas, o cara, o pessoal prefere se encher de programador do que de automatizador bom. É muito mais fácil para eles encherem uma fábrica de programador e falar que os caras estão trabalhando o dia inteiro, até mais tarde, do que pegar o automatizador que vai fazer a coisa funcional, no decorrer do tempo? Nem que demore um pouquinho, mas que vai ter resultado. Sim, é uma coisa que eu trabalho muito pra mudar, assim, nos gerentes que eu trabalho, nos clientes, de levar, gente, vale a pena o esforço, o sacrifício, o investimento, vale a pena a espera, o retorno é garantido, mas vocês tem que dar um voto de confiança e a pessoa que está ali também tem que estar empenhada em fazer dar certo. E assim, quando se trata de pessoas, a gente sabe que isso não é garantido, que talvez não de certo. Aquela série de fatores. A gente tá falando de um. E nós somos extremamente falhos.

Pergunta 5: Eu escolhia os principais, porque a gente tinha, as funcionalidades principais, focava nelas e a medida que, sei lá, e, geralmente, quando o time é bem engajado, sempre sobra um tempinho. Três, alguns dias, antes da homologação. E nesses três ou horas que sobrasse, eu atacava tudo aquilo que eu podia, ou que eu sabia que podia ser impactado maior, que podia ter um impacto maior. Sabe aquela funcionalidade que eu não sei se você trabalhou com isso, mas que a gente de tanto mexer com ela a gente sabe que ela pode dar mais problema? Essas daí eu já tinha

elas em mente assim, eu antes de começar qualquer execução, de ter qualquer plano de teste, qualquer coisa, eu pensava, não, independente do que aconteça, eu não posso deixar de executar tal cenário, porque ele pode me dar problema no futuro. De início, tipo, o número X limitado. E a medida que o tempo sobrava ou que aparecia alguma, umas falhas impeditivas que me parassem, aí eu ia para as outras que são, que eu sabia que isso não podia acontecer, que podia dar um impacto negativo, e depois as demais. Mas assim, é como tudo na vida, é impossível abranger todos os cenários. É humanamente impossível em uma equipe de duas pessoas.

Uma coisa que que você ressalta é a questão da quantidade de pessoas na sua equipe? Que isso impacta diretamente? E assim, é uma coisa que, eu não sei, eu nunca trabalhei fora daqui, eu não sei como é que é por aí, mas o pessoal tem mania de achar que, sei lá, o sistema é um pouquinho mais maduro, que ele não precisa de muito teste. Que teste bota duas pessoas, três pra fazer ali. Se tiver algum imprevisto, você pega emprestado. E faz acontecer. E não é assim, o ideal é que você tenha, no mínimo, três ou quatro pessoas engajadas nisso para ter assistência do início, meio, ao fim. Que são coisas que até hoje eu não tive, já tem quase sete anos na TI, eu não tive isso aí ainda. Porque faz do jeito que quer, quando quer e assim, eu desisti de bater de frente com o gerente, líder. Só quando realmente a coisa tá muito, muito feia que eu vou lá e brigo, e me ponho mesmo. Mas antes disso, o que eu vejo é que é uma cultura daqui. Eles têm a visão limitada, que eu já, e eu falo que assim, que eu sempre que eu tenho a oportunidade de trabalhar com esse tipo de gente, eu deixo bem claro. Meu bem, teste não é só saber ler. E assim, eu ainda brinco, eu sei um pouquinho mais que leitura, porque eu sei interpretar texto e eu sou muito exigente no que eu faço. E é por isso que o meu trabalho não é dos piores, mas se vocês dão um time de qualidade, se vocês dão máquina, suporte, assistência, salários melhores, a empresa de vocês deslança cem por cento, mas é uma mentalidade muito atrasada que tem, que teste qualquer um faz, que eu vejo que eu não, eu não tenho o poder de mudar. E é por isso que eu tento me moldar o máximo possível, aprender, mas não esquecer de tudo que eu passei lá atrás, pra não ir na pilha, de achar que é porque eu faço curso, porque eu tento fazer *networking*, porque eu tenho certificação, porque tudo que chega pra mim eu dou um jeito de homologar, então, que isso é o suficiente. Porque é uma visão que eu vejo do mercado de Brasília. Que teste qualquer um faz. E, geralmente, quando eu vejo esse tipo de gente, eu fico, relaxa, que daqui a pouco você cai do cavalo. Porque não é assim, gente. Não é qualquer pessoa que aguenta o estresse de homologação, o estresse de bater de frente com o chefe, o estresse de bater de frente com o colega de equipe, que você vê que tá fazendo de qualquer jeito. Não é qualquer pessoa, e não é qualquer analista que quer fazer isso, até porque, o pessoal acha que a gente, que a gente pode receber qualquer pouquinho, porque o que a gente faz qualquer um faz. E é uma coisa que eu não me permito mais ser assim, porque se eu trabalho tanto quanto o cara, um dev ou qualquer outro analista. Porque tipo, independente. Eu quero meu sistema funcionando. Eu quero ele funcionando direito. Eu ainda brinco, gente eu não passei o que eu passei na faculdade, para entregar um sistema meia meia boca. Se vocês sim, problemas

de vocês, porque eu vou cobrar. Independente, me chamem do que você achar, mas é errado. E é uma mentalidade muito errada que eu não tenho o poder de mudar, mas assim, eu sigo lutando para que as pessoas enxerguem que o que a gente faz é importante. Essencial, assim. E eu vejo, pelo menos, assim, de conversar com quem, com o pessoal que mora aí fora, que já está mudando lá fora. Aqui ainda não. Eu também fui da RSI, eu fui da fábrica de teste. E até na fábrica de teste, muitas vezes a gente tinha que se fazer de trouxa, de bobo. Porque o usuário não podia não gostar, ia ser ruim pro time, para a empresa. É uma questão cultural e assim, organizacional. Que é até uma coisa que a gente vê agora no LGPD. Porque não adianta ter a lei que for aquela coisa toda. Não adianta ter uma lei se você não consegue mudar a mentalidade das pessoas. A transformação digital, qualquer transformação que aconteça, ela se dá por pessoas, através de pessoas, como é que você muda alguém? Eu tenho ciência, eu não tenho esse poder de mudar ninguém, mas eu acho que o meu trabalho pode ressoar assim pra falar, olha, tem jeito, dá pra fazer diferente. Talvez o que ela fale tenha fundamento, só que assim, eu não tenho poder de fazer isso sozinha. Eu preciso de alguém, um nome um pouquinho superior ao meu para que se chegue lá em cima. Mas eu sigo tentando. Eu gosto muito do que eu faço. Tem uns colegas que falam: nossa, Raquel, você gosta muito de sofrer. Você podia arrumar outra coisa para fazer, eu, pois é, enquanto eu der conta disso aqui, está bom. No dia que eu achar que não, a gente vai arrumar outra coisa. Eu ainda acredito que dá pra deixar melhor.

Pergunta 6: Teste de regressão, na minha opinião, a gente vai lá, antes do aceite. Antes do aceite eu acho, eu fiz o aceite, o usuário deu mais ou menos uma olhada, não estava, sei lá, ou eu percebi, enquanto ele tava fazendo isso, eu percebi que tem alguma coisa que não está de acordo, que teve uma falha no requisito, o que o DEV ultrapassou, resolveu fazer alguma coisa no, resolveu fazer algo por conta própria, que acontece muito e isso já estava tecnicamente pronto ou pronto, para ser entregue, homologado, ou já foi homologado, está em uso pelo usuário e eu volto e faça um regressivo daquilo que tudo que tá envolvido, desde o sistema, se eu tiver acesso ao código eu posso tentar inspecionar, documentação, diagramas, até documento de banco eu já tive que fuçar para provar que realmente não estava de acordo, que alguém, por algum motivo, resolveu fazer, fugir do que estava sem justificar o porque ele foi feito. É a minha visão de regressivo, mas não é sempre que eu consigo fuçar tanto. Porque você tem que ir atrás do que, infelizmente, quase tudo, o pessoal só quer agilidade para homologar e pra faturar, mas na hora de fazer coisa acontecer, é no Hulk. Imagina, abrir dez, vinte documentos, e validar no sistema. É muita coisa. Um cenário de teste não é o suficiente. Nunca vai ser, em um regressivo, não é o suficiente. Algumas empresas, eu já trabalhei em empresa que adota, o regressivo. Tinha um regressivo próprio. Tinha um cenário para o regressivo. E eu tipo, quando eu comecei a ver aquilo, gente isso está errado, porque tá muito exato isso aqui. E a função do regressivo é justamente essa, para eu pegar aquilo que não está de acordo. Que fugiu a regra, que saiu da visão das

pessoas envolvidas por prazo, qualquer outra coisa. E por que é que isso aqui está tão fixo? Isso aqui tem que ser mutável, tem que ser adaptável conforme a gente descubra o que aconteça. Tanto é que o pessoal não gostou, achou ruim, eu achei depois você tem que rodar, mas no final eu consegui, eu consegui que me ouvisse, porque é mutável, tem que ser mutável e adaptado. Assim como as outras fases, porque se não, não vai atender as necessidades do usuário. Porque os sistemas, eles são isso. Tem, tem pessoas envolvidas, tem pessoas envolvidas, tem processos que se moldam, às vezes, muito rápido, ou que são feitos de qualquer jeito e tem software que tem atualização constante. Então, não dá pra ser fixo. A questão do, eu vinculo muito a questão do *mindset*, que ele tem que ser totalmente adaptado e tem que ser maleável de acordo com o que a gente viva. Como é que eu prego isso, mas eu acho que o sistema vai ser sempre aquilo. Que ele vai ter que funcionar naquela linha reta pra mim. É inadmissível. Como que conheceu o assunto teste regressão? Na faculdade. Eu tive, eu não me lembro o nome da matéria, mas foi um professor que ele, eu não lembro se ele trabalhava como analista de processo, mas eu lembro que ele focava muito nisso. Ele nem falava nem de teste, mas ele falava de regressão de arquivo. É uma coisa assim, eu não me lembro, ele era das antigas, ele falava de regressão de arquivos. E tinha um colega na faculdade que nessa época ele já era sênior, eu entrei na área um pouco mais tarde. Eu entrei na faculdade, tranquei um ano e acabou que sai um pouco mais tarde que a maioria. Desde a faculdade, tenho um colega, acho que foi até o Bruno, ele me deu um livro de teste básico de teste, aí eu já comecei a ver. E na época, tinha muita analista de requisitos. E os analistas de requisitos faziam testes e faziam teste de requisitos, e o teste delas era o regressivo. O sistema está pronto, o usuário reclamou, que não está de acordo, aí elas iam testar, elas iam fuçar de acordo com a visão delas para provar que estava de acordo. Desde a faculdade que eu vejo isso já.

Pergunta 7: Do analista e até do próprio usuário, porque acaba que a gente aqui, meio que mais trabalha em paralelo com ele. Olha, usabilidade se ela fosse uma ferramenta automatizada que facilitasse para eu fazer alterações, as alterações que aparecem no decorrer do teste, uma, tipo, sei lá, um processo de automação que fosse otimizado, adaptado e fácil de mexer. Que não fosse aquela coisa, tipo deixa eu ver que seria, no caso, aquela questão nem de usabilidade mas de facilidade para o usuário. Porque a gente, no momento que a gente pega uma ferramenta nova, a gente é usuário. Porque a gente tem que aprender para melhorar o negócio. Aí então usabilidade, performance, acessibilidade, facilidade de manutenção, do manuseio, de um modo geral. E o ideal é que ela não fosse tão pesada, porque hoje, o que é que eu vejo. Todo mundo está cheio de ferramentas, cheio de processos, e tudo mais. E chega uma hora, tipo assim, eu falo por mim mesma. Nem trabalha com, eu trabalho no CURRUPE. Tem dia que minha máquina da meio dia, e ela não aguenta da quantidade de coisa que eu tenho abertas. As ferramentas que eu trabalho, navegador, aí, às vezes eu preciso de um documento no Word, aí tipo, tem a ferramenta de, é porque agora a gente está em home office, a ferramenta de

comunicação com o time. Chega uma hora que sobrecarrega tanto que tipo, tira a principal funcionalidade de um sistema que eu vejo que ele tem que ser prático, simples e que ele esteja tipo disponível para o usuário o tempo todo. Eu acho que é outra coisa que tem que ser tem que simples de usar. Fácil, para gente entender, entendeu. Não ficar, aquela coisa prática. Eu sou bem prática nas minhas coisas, nem tudo precisa de tanta informação, mas ela tem que funcionar para que, tipo, eu acho que até um conceito que eu vi muito na faculdade. A documentação do sistema, tudo que envolve o sistema, ele tem que ser fácil pra mim, que tenho o entendimento, e também fácil para a tia da limpeza. Porque senão como é que ela vai usar, se ela precisar um dia. Como é que ela vai achar, sei lá, um sistema de folha de pagamento, sistema de ponto. São coisas que a gente geralmente passa, nem que for uma vez ou outra. Tem que ser fácil para o usuário usar. Então assim, a heurística de Nielsen, assim, total. Tem que entrar nisso aí, porque não dá pra fugir disso. No momento em que a gente pega uma ferramenta nova, a gente é usuário. A gente vai ter que fuçar, vai ter que questionar, vai ter que buscar informação para fazer aquilo funcionar. E se é para ser um regressivo, tem que ser rápido, prático. Não ter muito, tipo. Beleza, com o tempo, deixar melhor, mais bonito, mais eu acho que o termo certo é intuitivo. Agora me veio o termo certo aqui. Quase tudo mobile, quase todo aplicativo que eu vejo, eu fico, meu Deus cadê a facilidade. Como é que a pessoa vai enxergar o que ela vai ter que fazer aqui. Cadê o intuitivo desse negócio? Não tem. Tanto é que pra, tem muito, esses aplicativos, Facebook, INSTAGRAM, tem todo um estudo de neurociência por trás. Por quê? Para facilitar, prender, sites de compras, para prender o usuário lá. Porque quanto mais fácil, mais bonito, mais legal de ficar, melhor. Mas se é algo novo, ele tem que funcionar, ser prático e leve, na minha visão. Nada muito pesado. Se não, minha, tipo, eu brinco, minha máquina não vai nem aguentar. Nossa, eu vou ter que chorar para pedir permissão, se for no trabalho, porque vem ser muito fechado. E se provavelmente eu posso fazer isso tudo e não conseguir usar, porque vai ser mais uma ferramenta que pode pesar na minha máquina e travar. E a tipo, meio que prejudicar o meu rendimento no decorrer do dia.

Pergunta 8: Eu acredito que sim, provavelmente sim. Assim, eu não consegui automatizar, mas até porque o processo de programação e de regressão, de teste automatizado, regressão ou aceite, porque são os mais feitos, eles têm que andar em paralelo. Se é isso que mais tem no mercado hoje, então, ela tem que seguir essa linha para facilitar os lados, senão vai ser um tempo investido em vão. Tempo, dinheiro, pessoas.

Pergunta 9: É para mim, é, tempo, qualidade do serviço que você vai entregar, tempo, qualidade do serviço que você vai entregar, e você diminui muito multa, problema com o usuário, porque você vai, consegue fazer a coisa no tempo hábil. E ao invés de estar perdendo muito tempo com o defeito que deixou passar, você pode estar buscando melhoria, e melhoria gera receita para a empresa. Gera receita para

a empresa, e conseqüentemente, gera usuário feliz. E usuário feliz é a melhor coisa que tem. Vai por mim. Eu tenho usuários que não são tão felizes e meu Jesus, dá muito trabalho.

Pergunta 10: Olha, se não foi feito na correria, de qualquer jeito, sem assim, não pode ser, na minha opinião, não pode ser feito na correria de qualquer jeito. E muito menos sob pressão, porque o usuário quer e porque o chefe está com pressa. Tem que ser focado, olha, a gente vai ter que deixar isso bom. Se está no regressivo, se é pra validar se o código, layout, se a gente não deixa passar uma regra ou nada, ele tem que ser feito com qualidade. Tem que ser do tipo: se eu me empenhei pra fazer o aceite, para fazer a homologação, o teste de homologação, tem que ser feito do mesmo jeito. O empenho é o mesmo. Tanto é que muitas empresas têm o hábito de reduzir o time na hora do regressivo. Onde já se viu isso? Se é para, uma vez que eu deixo ele bom, ele não, eu não vou ter que fazer um segundo teste, eu não vou ter o, eu não vou desgastar o meu time à toa e nem o meu usuário. Então, tipo, e assim, ao invés de ficar fazendo um regressivo, dois ou três, eu já tive que fazer quatro regressivos, porque a coisa não estava de acordo. Porque eu não tive tempo de fazer a coisa certa no início e fui na pilha dos outros. E acabou que o usuário perdeu totalmente a confiança no que a gente estava fazendo. Aí hoje eu entendo, é melhor que eu faço bem feito, no mínimo um regressivo, e depois eu vou buscar melhoria, sugestão para para implementar tudo o que a gente já tem. Ou pras demandas futuras, entendeu. Vale a pena, e quanto bem feito, melhor pra gente. Usuário feliz, é usuário que sempre vai dar demanda para o cliente, para a empresa, entendeu? Para o time, como um todo.

Pergunta 11: _____

3. PARTICIPANTE 3:

Pergunta 1: Na verdade eu trabalho mais com a parte de desenvolvimento em si. E como desenvolvedor a gente é requerido a implementar alguns testes para poder validar as nossas aplicações. Então, assim, o meu histórico de desenvolvimento, desenvolvedor, no caso, ele tem cerca de cinco anos. E desde então eu já venho tendo que praticar e executar alguns tipos de teste.

Pergunta 2: Primeiro lugar assim foi dentro do meu time. Então, com conversa com meu outro parceiro de desenvolvimento, ele já tinha algum conhecimento, aprendi um pouco lá, da parte dele. E depois tive que correr atrás. Então, realmente, atrás, de sites, de fóruns. Eu gosto muito de usar o MEDIA. O MEDIA e o próprio STACK OVERFLOW como fonte de sugestão, de dicas e afins.

Pergunta 3: São linguagens nativas. Atualmente estou trabalhando mais com Kotlin, mas, ainda temos código legado em Java. Então, basicamente é por decisão do projeto mesmo, para manutenção, por conta do time também. **Ferramentas:** Sim, eu trabalho mais com a parte de teste unitário e teste de interface. Atualmente estou usando uma biblioteca chamada MOCKK para mocar dados em testes unitários e o JUnit para executar os testes dentro do Android Studio.

Pergunta 4: Bom, a gente trabalha na questão dos testes unitários, a gente implementa os testes unitários e roda eles no primeiro momento. Depois disso, a gente cria um SAMPLE que a gente chama, que a gente também roda os testes em cima deste SAMPLE, que seria mais ou menos uma emulação do aplicativo na vida real. E depois a gente repassa para o nosso QA. Gera uma versão para passar para o QA e ele faz outros tipos de testes, como teste de carga, de integração. **Ferramentas para teste de novas versões:** Sim, a gente utiliza o Appium e o Selenium.

Pergunta 5: Em geral, a gente executa os casos de testes em cima da feature que a gente mudou. A gente trabalha com modularização. Em geral, os casos de testes, eles englobam todos os aspectos daquele módulo. Mas, quando a gente faz uma mudança específica dentro daquele módulo, a gente não retesta todo o módulo, a gente retesta os casos de teste que tenham haver com aquela funcionalidade específica. **A escolha dos casos de teste a serem reexecutados é automatizada, ou manual?** Nesse caso, ela é feita de forma manual. A gente também tem uma esteira, que a gente chama, para fazer *deploy*, que ela executa alguns cenários de teste que a gente projeta. Ela verifica se tem testes unitários, verifica também a cobertura de testes, e ela dá um parecer. A gente usa o JENKINS, para automatizar depois a questão da própria release. Mas aí, na hora de definir, esses casos são definidos pelo QA. Ele automatiza, mas, a definição é dele.

Pergunta 6: Tá, eu não sei se dizer, Eu não saberia te dizer.

Pergunta 7: Eu acho que seria interessante mapear todas as modificações, para eu ter uma noção de tudo aquilo que eu modifiquei. Depois seria mapear, dentro daquilo que eu modifiquei, quais têm testes vinculados, seja, testes unitários, principalmente testes unitários.

Pergunta 8: Eu acredito que sim. E além do que por exemplo, a gente precisa testar diversos cenários que dependem do ambiente, da plataforma.

Pergunta 9: Sim, é muito importante, porque por exemplo, quando a gente está fazendo manutenção, a gente está em geral mexendo com alguma feature que já tinha sido previamente testada. Então cada modificação que a gente faz, ela pode levar a um novo problema, então se eu não consiga, não refaça os testes, por mais

que eu acredite, que eu esteja fazendo tudo certo, e bom, sendo desenvolvedor, é um negócio que nunca compila de primeira, então assim, é muito importante, para saber se aquilo que a gente fez está certo, se está rodando, se não impacta em outro pedaço do meu aplicativo, em outra feature, em outra parte.

Pergunta 10: Com toda certeza.

Pergunta 11: _____

4. PARTICIPANTE 4:

Pergunta 1: Bom, então, isso é um ponto, eu trabalho, na verdade, diretamente com o desenvolvimento de aplicativos. Eu não faço exatamente os testes. Os testes são feitos por uma outra equipe de profissionais. Porém, por um bom período assim no meu trabalho, não existia esses profissionais, então acabava que qualquer desenvolvedor fazia os testes dos aplicativos. E eu vou ser bem sincero assim, eu não tenho uma base, uma formação em teste. Então, o que a gente entende por teste é meio que um teste de usuário. Teste de ali, tentar procurar no aplicativo falhas e corrigir elas antes disso ir para produção, para algo, para o uso mesmo efetivo. Então, minha experiência mesmo, como desenvolvedor, nos momentos que eu trabalhei com testes, foram nesse sentido, a maior parte do tempo. De um ano pra cá, mais ou menos que eu tive uma oportunidade de ir um pouco mais além, e trabalhar também, com teste unitário. Então, a percepção que eu tenho, é que a maior parte das empresas elas não consideram muito os testes unitários como uma prioridade. O importante é você fazer o desenvolvimento do aplicativo de acordo com tudo que foi planejado, com a regra de negócio, e no caso justamente às vezes, porque conta de prazo mesmo, porque o prazo é muito curto, ou então, é um prazo que não comporta teste, a realização dos testes, opta-se por não fazer os testes unitários. Isso, eu diria que eu acho que uma parte, uma parcela pelo menos dos produtos que eu trabalhei assim, foi xxxxxxxxxxxxxxxx, para fazer o desenvolvimento. Eu diria que tem exceções. Por exemplo, eu não cheguei a trabalhar, mas eu já participei de processos seletivos para algumas oportunidades, por exemplo, de desenvolvimento de aplicativos financeiros. Nesse caso, especificamente, eles consideram os testes unitários, no caso, que é os testes que eles realizam, como parte do desenvolvimento. Então, você tem que fazer os testes unitário, até porque é um pouco mais complexo, porque envolve dinheiro mesmo. Se ativar uma falha, alguma brecha ali no aplicativo, de um desenvolvimento que não foi tão bem testado, a pessoa pode perder, sei lá, de centavos a milhares de reais, dependendo, por exemplo, de um aplicativo de um banco ou XXXXXX. Então, nesses tipos de aplicativos eu sei que eles são um pouco mais críticos, eles levam isso mais a sério, digamos assim.

Pergunta 2: Tá, certo. Do meu curso de formação, na faculdade mesmo, eu não

tive nenhum aprofundamento em relação a teste. A gente teve uma base, mas uma base bem superficial, eu diria que não é algo que você conseguiria pegar por essa base e já implementar ou fazer seus próprios aplicativos, em larga escala, enfim. Então eu diria que foi mais do, no, durante o próprio processo de trabalho mesmo que eu fui pegando mais experiência, conversando com outros profissionais mais experientes, também pegando informações via internet mesmo, fazendo busca de artigos, informações sobre testes, amigos que já trabalhavam já com isso, então, um pouco de cada lugar assim. Acabou meio que moldando um pouco do que eu conheço hoje, como soluções, por exemplo, para realizar teste unitário no aplicativo.

Pergunta 3: Certo. Eu sempre trabalhei com linguagem nativa, ou Java, ou Kotlin, ou os dois. Com relação a critério, é meio que acho que antes, pré, por exemplo, vamos supor que a gente resolve desenvolver um aplicativo, a gente está naquela primeira ideia, a gente tem uma ideia de um aplicativo, mas não foi desenvolvido, não foi programado nada desse aplicativo, está pensando em como vai ser a distribuição dele, para qual público, coisas deste tipo. Nesse momento, pode se considerar que você vai se desenvolver de forma nativa ou de um, utilizando um outro, uma outra forma que seja híbrida, digamos, sei lá híbrido, uma forma que de para você disponibilizar em uma linha de desenvolvimento, IOS, Android, Web, de repente. Seria bem grande. **Utiliza alguma ferramenta para automatizar os testes?** Para automatizar os testes, eu não trabalhei diretamente com automação de testes, para ser bem sincero assim. No desenvolvimento de aplicativos, existem algumas ferramentas que você pode incluir no processo para ela rodar testes, como Jenkins, por exemplo, que você configura para fazer processo de UI, de tudo. Dai nisso você pode configurar para ele ler todos os testes unitários que estiverem no projeto. Ou testes instrumentados. Isso é possível, mas eu não trabalhei diretamente com isso, geralmente quando eu peguei, eu trabalhei em projetos que já tinham as ferramentas embutidas, já configurada, então, de certa forma pra mim era transparente. Então, por exemplo, se eu terminava de desenvolver alguma parte do aplicativo e a gente ia fazer um processo de ROLAUD, para subir para produção, alguma coisa desse tipo, já, esse processo de execução dos testes automatizados, ele já tinha sido pré-concebido no aplicativo. Eu não, não tinha participação nisso. Então, eu não sei exatamente como funciona. **Para rodar testes unitários?** É, tem duas formas. Na hora que você está escrevendo o teste unitário, você usa lá a IDE por exemplo, a mais comum no Android, a gente está falando de Android Studio, então você vai desenvolvendo o código do teste unitário, e quando você terminou lá um bloco de desenvolvimento, você vai e roda o teste pela própria ferramenta do Android Studio. Você roda para ver se está tudo confirmado, conforme você escreveu o teste. Isso seria a primeira coisa, o Jenkins, por exemplo, ele vai cuidar de mais coisas além disso. Ele vai cuidar de fazer o build da aplicação mesmo e em um momento específico ele vai correr em todos os teste unitários que tiverem sido desenvolvidos para aquele projeto e vai rodar todos em uma sequência para ver se todos estão ok. Como se fosse uma segunda garantia de que o que foi desenvolvido, mesmo que foi

lá uma ação, um pedaço pequeno do aplicativo, não vai impactar em nada do que já está desenvolvido antes, ou, durante por outro desenvolvedor, que também está juntando as coisas que ele desenvolveu. É uma parte do processo, mas, tem outras coisas.

Pergunta 4: Tá, no fluxo de entrega de release que você comentou. Nessa empresa que eu trabalhei, a última, que eu acho que eu vivi um pouco melhor esse processo, o que é que acontecia. A gente fechava lá, as features que seriam entregues nessa release, e a partir disso a gente congelava aquela *branch*, aquele fluxo de desenvolvimento para que não fosse feito mais nada, até esse momento, e por um período a gente executava teste em relação aquela versão. Os testes unitários eles aconteciam antes. Eles eram feitos durante o desenvolvimento, na hora que já apresentavam o que seria funcional para a release, já foram rodados os testes unitários. O processo que acontecia depois disso, era o de chamar a equipe de testes, eles recebiam a versão do aplicativo, uma versão candidata a release. E em cima dessa versão eles preparavam, ele viam quais foram as features que foram desenvolvidas, o que é que, quais as mudanças que teve no aplicativo, pegavam também o conjunto de mudanças que teve na release anterior e eles realizavam um teste, dessa, do que foi desenvolvido antes, que eram os testes regressivos que a gente chama, e testes do que foi desenvolvido agora e seriam subidos para a produção. Então, isso levava em torno de uma semana, vamos dizer assim. E é um processo realmente que mobilizava uma boa parte, até às vezes envolvia os desenvolvedores justamente para ajudar nessa parte para ver se nada, se tinha acontecido algum problema ali, uma quebra, uma coisa assim, que o aplicativo deixou de funcionar alguma parte. E daí, no caso, a gente usava o JIRA para documentação, e para organizar todos os tipos de pontos que a gente ia fazer. Nada disso era automatizado, era tudo manual.

Pergunta 5: É, então, quando a gente, em termos de teste unitário, a gente já executa todos os testes anteriores, mesmo. É um processo que ele não leva bastante tempo, porque depende da quantidade, por exemplo, o projeto que a gente tinha, talvez ao todo tivemos uns 1500 de teste unitário. E na hora de fazer uma versão, de geral uma versão de release, o procedimento era realizar todos os testes anteriores. Não era por amostragem.

Pergunta 6: Então, eu entendo nesse momento, os testes regressivos como sendo aqueles testes, para executar funcionalidades que já foram entregues, para ver se não aconteceu nenhum problema nas funcionalidades, na hora da release. Então, não é nada de novo. É fazer um teste nas funcionalidades que já estão desenvolvidas, que já estão em produção, já estão executando, e não foi alterado nada em mente. Nenhuma mudança deve ocorrer, mas eu vou fazer o teste na verdade só pra garantir que vão continuar funcionando mesmo, pra não ter o risco de alguma coisa que vai na nova versão, impactou em alguma coisa que já estava funcionando. **Como você conheceu esse assunto, teste de regressão?** Foi mais na prática,

para ser bem sincero, foi eu acho que na minha última experiência profissional, que por conta de a gente ter esse período, de teste. É uma coisa que quando a gente leva um pouco mais a sério, eu fui entender que existia um tipo de testes, e um deles era os testes regressivos e quais áreas do aplicativo que a gente fazia esses testes regressivos, aí entendi a partir disso. **Possui alguma dúvida sobre esse tema?** Eu acho que não, não sei se eu tenho dúvida, acho que não.

Pergunta 7: É um pouco difícil para mim medir isso, porque a minha visão é um pouco como desenvolvedor, mas eu acho que a priori, e eu não sei se isso é uma característica dos testes de regressão, mas, eu acho que qualquer tipo de teste, eu integraria nesta ferramenta, uma forma de testar, de fazer os testes unitários do aplicativo, pra garantir mesmo, porque os testes unitários, também tem essa importância, que é garantir que o que estava funcionando antes, não deixou de funcionar. Inclusive, quando, por exemplo, entra um novo desenvolvedor na equipe, que não conhece do projeto, ou que vai mexer em alguma coisa que, de repente, foi desenvolvido há muito tempo e não causar algo crítico que deixe de funcionar alguma funcionalidade, então os testes unitários, eles garantem isso. Então, uma das coisas que deveria executar, seria isso, os testes unitários. Eu, então, aí teria uma coisa que teria que entender melhor como funciona o software de testes nesse sentido, porque eu acredito, que por exemplo, o teste que a gente fazia manual de regressão, que era o de abrir o aplicativo, naquela sessão do aplicativo fazer a interação do que era realmente para ele executar. Eu acho que tudo isso é possível fazer de uma forma automatizada, mas eu não tenho conhecimento de como seria, provavelmente, alguma coisa desse tipo, mas, eu não sei dizer como que teria que ser esse *script*.

Pergunta 8: É que eu não tenho esse conhecimento das linguagens híbridas, que eu não trabalhei com elas, assim, por exemplo, para poder dizer assim se existe vantagem, uma desvantagem, em realizar os testes. Eu não sei, na verdade, nem se é possível fazer testes em React-Native, e a outra que eu esqueci o nome agora. A opção que existe hoje para fazer desenvolvimento híbrido para IOS e Android. Então, eu nem sei, eu não conheço essa parte de testes, essas ferramentas, a minha percepção é que assim, quando você trabalha com nativo, você tem mais flexibilidade para aplicar coisas e realizar coisas durante o desenvolvimento. Então, por exemplo, com Java e Kotlin eu sei que eu consigo por teste unitário tranquilamente. O próprio Android Studio, na IDE, fornece bibliotecas, têm facilidades para eu realizar testes unitários. Já com isso híbridas, eu não sei, como ficaria, se existe esse suporte, ou se no fundo vai acabar precisando usar algo nativo para fazer os testes.

Pergunta 9: Sim. Na verdade eu penso assim, principal, eu acho que o principal ponto de automatização de teste pra mim eu acho que é ganhar tempo mesmo. Você, infelizmente, você fazer isso manualmente, é um processo meio, digamos assim chato, e é muito repetitivo e eu acredito que essas coisas que são repetitivas, de

alguma forma a gente consegue automatizar, para abrir espaço para produtividade. Então, eu acho que o principal ponto, realmente é ganhar tempo com automatização de teste pra gente não precisar ficar refazendo sempre a mesma coisa, na hora de fazer os testes. E eu acho que dentro do fator de organização aí que é importante porque, enfim, porque é legal ter os testes automatizados, mas tem que ter uma pessoa por trás, que crie esses testes de uma forma correta, pra ela também não ser prejudicada. Então, eu acho que seria meio que nesse sentido? Ter uma forma de garantir que os testes que foram escritos para serem automatizados, eles são testes fidedignos, eles não são com falhas, porque daí falha o próprio processo de teste.

Pergunta 10: Com certeza. Com certeza. Acho que é fundamental.

Pergunta 11: Não, na verdade eu estava, quando você me mandou, na verdade, o formulário, eu fiquei pensando, eu acho que não é exatamente minha área. Eu sou desenvolvedor, eu não sou tester, aí eu fiquei pensando, eu não sei se eu vou ajudar muito na pesquisa, mas enfim. E eu achei que as questões são, estão relevantes, inclusive eu acho que você pontuou esse questionário, principalmente pensando aí no projeto.

5. PARTICIPANTE 5:

Pergunta 1: Então, na verdade, eu já atuava com qualidade, com um geral, assim, em cinco anos atrás, mais ou menos assim, eu ainda estava na faculdade, indo para o último semestre e acabei, assim, um amigo de classe, me indicou a empresa dele e aí eu comecei a atuar com testes. E aí, no começo eu era só um executor de teste, então, eu basicamente lia *script*, fazia o que as ações pediam e tirava evidências. Então, era aquele cara bem cru aqui na área de teste. Não sabia nem o que era um QA de fato, como um todo. E eu comecei a trabalhar validando mainframes. Mainframes de bancos, instituições financeiras, arquivos posicionais, tal, e ainda não trabalhava com aplicativos, micro serviços, tecnologias mais atuais. Então aí nessa, eu vim, nessa, nessa jornada, e quando eu fiz mais ou menos dois anos e meio, três anos de carreira, eu transicionei para uma empresa e quebrei todo esse paradigma que eu tinha de tela preta e comecei a aplicar mais qualidade como um todo e sair daquele cascata, comecei a trabalhar com metodologias ágeis, SCRUM, KANBAN, e aí acabei sendo forçado, obviamente, tanto na questão tecnológica, quanto na questão do que eu estava acostumado a fazer, mesmo sendo na mesma área, e aí comecei a lidar bastante com aplicativos, tanto iOS quanto Android, questão de micro serviços, entre outras coisas que estão integradas ali, como um todo, que hoje basicamente quase todas as empresas aí que tem um negócio, tem a parte mobile e é bem importante.

Pergunta 2: A fonte, assim, número um, eu acho que foram cursos e em segundo

lugar, a empresa só que foi meio que num conjunto, é meio que, eu perguntava como certas coisas funcionavam, porque os desenvolvedores XXXXX são muito, tipo, parceiros, então, conseguia bastante coisa com eles e aí o que eu ficava, assim, não faz tanto sentido eu perguntar pra eles, porque entra mais no quesito teste, só que dentro de um ambiente Android, eu comecei a consumir bastante cursos. Então, e aí também eu já comecei a trabalhar com automação. Então, vindo para automação de testes, principalmente, da parte mobile, que a gente usa o Appium, eu tive que começar a entender um pouco mais como funcionava a estrutura dos aplicativos, como era construído *views*, *models*, *landing pages*, e partes da estrutura do desenvolvimento, tanto iOS quanto Android, mas também aqui, eu vou dar mais ênfase no Android. Então, eu tive que me forçar a aprender. E aí, nessa, nesse ponto de desenvolvimento, eu consumi muito, assim, tipo, questão de quatro a cinco cursos, ALURA, UDEMY, entre vários outros lugares, também, MEDIUM, tem vários artigos muito ricos, e os desenvolvedores estarem do meu lado durante o período aí, a jornada de trabalho.

Pergunta 3: Hoje aqui onde eu atuo, e na minha última experiência, são desenvolvimento nativo, em Kotlin. Tem algumas coisinhas ainda que estão no Java, e estão em processo de refatoração, uma parte mais legado, mas eu posso falar que 90 a 95% é Kotlin. E aí, hoje, onde eu atuo é só Android, não tem aplicações iOS. Eu não sei explicar como foi dado o contexto, da oportunidade de onde eu estou, que tem pouco mais de um mês que eu estou nesta empresa, mas basicamente assim, na maioria dos lugares eles optam pelo desenvolvimento nativo pelo suporte que sempre teve, desde que foi surgindo, e aí, hoje, mais atualmente, desenvolvimento híbrido, é uma coisa que está crescendo. Então eu acho que por um aplicativo já ter uma base, e ser um pouco mais insinco, um pouco mais de questão de tempo, no começo eles usaram Kotlin mesmo. Mas não tem um, eu não sei o porquê desta empresa ter escolhido desenvolver em Kotlin. Ferramentas para automatizar testes: Sim. Quando eu entrei eles tinham alguns testes instrumentais, que estava feito com Espresso, e a gente está tentando tirar esses testes porque empata no *build*, demora um pouco mais, porque eles têm que serem esgotado, e meio que antes de liberar a versão, mas que foram pós *build*, então a gente está tirando esta responsabilidade, e hoje a gente está usando Rubby como linguagem com Appium de ponte para gente criar os *scripts* de teste automatizado em GHERKIN e Cubumber, e montar uma STACK bem solidinha.

Pergunta 4: A gente não tem como se fosse uma receitinha de bolo, um step-by-step, mas, a gente tem um documento no CONFLUENCE, que mostra quais cenários devem ser validados, e ele cresce a cada release. Porque toda release a gente entrega uma feature nova, pelo menos, uma feature. Então a gente sempre vai incrementando este documento, e a gente está tentando é suprir essa quantidade de cenários que tem neste documento, com *scripts* automatizados com Rubby e Appium. Então a gente está tentando chegar em um regressivo automatizado, com

uma cobertura legal de 80 a 90%, para tirar também esta responsabilidade. Mas, hoje assim, dentro do contexto que a gente tem, se a agilista precisar de executar os testes, de fato só precisa de um *device* Android e consegue executar. A gente distribui versões pelo AppCenter, se quiser também, tem como pegar uma APK e fazer um build rapidinho. Tem uma DEVICE FARM que a gente consegue usar lá com o BROTHER STACK. Então a pessoa não precisa ter um *device*, ela pode executar tudo por essa ferramenta que a gente tem, e inclusive pegar a versão da play store, ou do próprio AppCenter ou de um APK que você faz upload e testar. Então é bem tranquilo, apesar de que hoje, fica só nas mãos do QA essa responsabilidade, mas qualquer pessoa conseguiria executar tranquilamente. **Ferramentas do processo de automação das novas releases?** Então o que acontece, quando a gente distribui uma nova release, na primeira vez que sobe, geralmente a gente ainda não consegue, a gente está dando passos para trás, para depois dar passos para frente. Então o que a gente faz, a gente está automatizando o que já existia, para chegar nessas features novas automatizadas. Então a gente está testando elas de maneira manual, exploratória, só para garantir que a feature está certinha como foi especificada nos requisitos, na documentação, e aprovada sobre os critérios de aceite, para a gente conseguir distribuir para o usuário final, e a gente vem caminhando para chegar em um ponto, que a gente consiga entregar na mesma sprint ou na mesma release, já esses cenários automatizados. Ainda não é o que acontece, mas, é o cenário perfeito que a gente quer chegar.

Pergunta 5: Depende. Se teria essa feature com mil casos de teste, a gente executaria um MOCK, mas, seria um pouco mais parrudo, por amostragem e dependendo da funcionalidade. Se for X casos de teste por funcionalidade. Mas no caso disso já estar automatizado, eu confiaria no tempo de execução que demorasse para rodar os testes automatizados. Se já estiver 100% automatizado, eu prefiro retestar tudo, porque eu deixo uma pipeline executando e depois ela só me envia um relatório ou me exibe um relatório e eu consigo ver quais cenários passaram, quais não, talvez executar eles apartados, ou ver, a gente colocou sei lá, agora tem uma mensagem nova, um *alert*, e esse *alert* quebrou cem testes meus, porque ele está no meio de um fluxo, então às vezes é uma coisa simples, que eu ajusto *script*, ou às vezes, é algo que de fato é um *bug* mesmo porque estou clicando em um botão e ele não está redirecionando, então é algo que precisa ser analisado. Mas, com uma automação bem solidinha feita, eu já trabalhei em outros lugares que a gente tinha uma automação um pouco mais sólida, no caso de a gente pegar um aplicativo desde o dia zero, desde a primeira documentação e chegamos na parte de 250 casos de teste, para cada versão, 250 para iOS e 250 para Android, e a gente conseguia confiar cegamente que 90 a 95% só no que ele extraia para gente de relatório, a gente passava uma versão exploratória mesmo, em um *device* físico para conseguir aferir a qualidade. Então era assim, deixava 4 horinhas rodando ali, e tranquilamente conseguia falar: olha, rodou aqui na automação, 99% dos cenários passaram, e esse 1% que não passou foi causa disso, disso e disso, pode subir sem nenhum problema,

e dava para confiar. E aqui a gente quer fazer o mesmo. Eu estou tentando trazer o máximo de automação para cá, para tirar essa responsabilidade das pessoas e deixar para a máquina.

Pergunta 6: É um teste de features que já foram entregues, que precisam continuar funcionando do jeito que elas foram desenhadas para funcionar. Então são testes de features que já foram entregues e continuam tendo sua funcionalidade e precisam continuar funcionando, então a gente tem que reexecutar para garantir a integridade dessa feature. Então a gente faz um teste de regressão, literalmente, é o que já foi entregue e que precisa continuar funcionando. **Como você conheceu sobre esse assunto?** Eu não vou saber exatamente como eu descobri o que é teste de regressão, mas, eu consumo muita coisa assim, quando termino meu dia, ou às vezes durante o dia mesmo, paro cinco, dez ou quinze minutos para tomar um café, ir ao banheiro, esticar as pernas, e aí eu estou lendo um MEDIUM, eu estou lendo bastante coisa, vendo vídeos, tem bastante coisa de referência de qualidade no mercado, tipo Júlio de Lima, entre N pessoas, PESSOALIZANDO, que eles conhecem muito tipo qualidade como um todo, para mim são tipo pilares de hoje o que a gente tem hoje como qualidade, e eles sugerem materiais incríveis. Então pode ser que eu descobri isso no trabalho com alguma palestra ou alguma pessoa de senioridade maior na época falando, ou pode ser algo que eu li e comecei a pesquisar sobre e fui vendo. Um exemplo, tipo de teste, MOCK teste, teste regressivo, teste de mutação, teste de estresse e carga da performance. Hoje assim, me considero um cara que sabe um pouquinho de cada coisa e sei me virar. Então, acho que isso veio naturalmente, conforme fui atuando na área de qualidade como um todo, eu fui absorvendo mais coisas. Você possui alguma dúvida sobre o tema Teste de Regressão: Eu acho que para mim é algo tipo natural já. Eu que metade da minha experiência trabalhei com a frente mobile, e é onde a gente mais aplica regressão, acho que afinca, porque às vezes em aplicação web, desktop, o pessoal faz teste regressivo, mas, não é core. Eu acho que sempre em aplicações mobile tem que ter teste regressivo. Constância de manutenção muito alta. Geralmente tem empresas que fazem duas *releases* em uma semana, outras uma por semana, e outras a cada quinze dias, mas assim, com duas *releases* semanais é quase que impossível você não fazer regressivo. Não tem como.

Pergunta 7: Eu conseguiria desenhar ambientes que acho que seria necessário. Então suportar tipos diferentes *devices* em diferentes versões de sistema operacional, diferentes tamanhos de tela, isso assim, só para começar a editar os mesmos *scripts* só que em *devices* diferentes. Teria tipo pipeline, disparo automático, extração de relatórios, se possível, medir a temperatura do *device*, *tracking* de rede, quais requisições, quanto eu mandei, o quanto eu recebi, era necessário tudo aquilo mesmo. Nossa, não o sei o que mais assim, mas, acho que num geral tem que ter, é um core, daria o maior suporte possível para tudo o que eu precisasse extrair, relatórios, BLOCKQUOTE, MOCK, principalmente para fazer cenários negativos,

tipo de falha de conexão, *retry* e afins. Poderia ter uma baita de uma *STACK*, coisa que a gente tem pedaços dela em diferentes ferramentas, mas, não é fácil achar tudo junto.

Pergunta 8: Eu acho que as ferramentas para automatizar testes de regressão com aplicações nativas, no caso Kotlin, elas são mais confiáveis, sólidas. O Appium já tem um tempo, e vem recebendo manutenções, manutenções não, atualizações constantes. Eu falo só do Appium porque ele é o único que consegue testar de fato um ponto APK já buildado, então, eu consigo testar de fato um artefato que se ele for validado, ele pode ir direto, só trocar a URL e ambiente, direto para o usuário. Eu até cheguei um pouco sobre, o Appium chegou a trazer uma versão do Appium para aplicações híbridas, que são baseadas em *Widgets*, acho que é Flutter, que funciona assim. E assim, é algo muito novo, principalmente para esta parte. Talvez, não tem nenhuma relação. Eu falo pela automação web, hoje a gente consegue escolher uma linguagem com bastante apoio dos desenvolvedores, por exemplo, se é JavaScript que eles fazem ou JavaScript Node, a gente consegue usar também uma *STACK* de JavaScript com SAIPRES ou com CodeSEPTIUM, ou outro *framework* ali no meio, mas, eu acho que tipo não tem relação ser Kotlin com a linguagem que eu vou usar para fazer os códigos dos testes automatizados. Eu ainda não sei se tem como fazer em Kotlin, mas, geralmente o que a gente mais vê é Rubby, JavaScript e Java, com Appium. O Appium que a gente tem hoje serve tanto para Android quanto para iOS, já para aplicações híbridas literalmente eu só conheço esse Appium Flutter Drive que chama. E ele trabalha em aplicações desenvolvidas em Flutter e é algo super novo, inclusive, coisa de um ano e meio a dois anos atrás, e tem pouco, eu não sei quão grande ele é, sabe. Ainda não consegui consumir tanta informação sobre, muito menos fazer alguma *POCK* sobre, agora com Appium .APKS e .IPAS eu consigo assim brincando, sabe. Acho que tem muito mais material.

Pergunta 9: Eu acho que é menos suscetível a falhas. Inclusive, eu como QA e minha visão de qualidade, eu creio que a automação tem que ser o máximo possível dos testes funcionais automatizados, porque se a regressão for, que seja só para o QA ou para o analista de teste da *SQUAD* ali da empresa, ele faz toda semana, a mesma regressão e uns testes novos da nova feature. A pessoa é suscetível a erros, o vício de execução: ah, agora eu vou testar só com um usuário, ao invés de testar com três, quatro, com diferentes massas. E pode ser por ignorância, pode ser por falta de tempo, release em cima da hora, enfim, N coisas. E eu acho que a máquina, ela não é suscetível ao erro. Se você faz um *script*, ele vai rodar do mesmo jeito. Então eu creio que a gente é menos suscetível a erros com isso, com processos sistematizado, padrões, documentações, boas práticas, se possível assim, *KRONS*, tudo bem estruturadinho.

Pergunta 10: Mas, com certeza, é impossível você ter qualidade e sim fazer teste de regressão. É um core.

Pergunta 11: Assim pra mim é algo muito novo, é a primeira vez que isso acontece. De eu participar do lado de cá, eu já estive do lado contrário, também, fazendo, mas foi pra produzir um artigo na época da faculdade. Então, eu entendo exatamente como é, e entre esses pontos e acho que é isso mesmo, tipo, não tenho algo, assim, direto, de fato, pra falar.

6. PARTICIPANTE 6:

Pergunta 1: Bom, eu comecei a trabalhar com Android em 2017. Com Android em 2017, e o que acontece, lá onde eu trabalhava na época, em Salvador ainda, não se fazia testes, na verdade. Os testes que a gente eram aqueles testes, roda, faz a feature, roda a aplicação, e testa ali para ver se está funcionando. Este era o teste que a gente fazia. Não era um teste automatizado, entendeu? Ai, o que é que acontece, aí passou 2017, 2018. Em setembro de 2018 eu vim para São Paulo, e lá para dezembro eu comecei a trabalhar na Click Bus, pronto. Ali, de fato, eu comecei a trabalhar com teste no Android, entendeu? E assim, aí lá, a gente tinha a seguinte estrutura, a gente usava mais testes unitários com MOCKITO, certo, e antes, até eu sai de lá, a ideia era usar o Espresso para fazer testes digamos assim instrumentados, nesse sentido.

Pergunta 2: Vou te falar. Na faculdade foi bem pouco. Foi uma coisa que eu senti falta na faculdade as pessoas, o que a gente aprende de teste na faculdade, é porque eu sou formado pela Católica, Universidade Católica de Salvador. Ou eu estou morando em São Paulo, mas, eu estou voltando para Salvador, mês que vem. E eu fiz pós no IFBA. Na Católica, no meu curso, era um curso muito massa assim, mas, assim, a gente não teve essa questão de você fazer os testes. A única coisa que eu me lembro, referente a teste, era algo assim, aquele teste de mesa, para gente logo no começo, quando a gente está aprendendo algoritmo e a gente faz aquele teste de mesa para ver se nosso algoritmo passou, mas, poderia aí nesse, aproveitar para ensinar teste, já desde o começo. Já que você está fazendo um teste de mesa, e que você está vendo se o algoritmo está funcionando ou não, é um bom momento para você aproveitar ali e ensinar testes. A pessoa já vai chegar para o mercado lá na frente, já que ele está aprendendo da base, preparada para teste, entendeu? Muito melhor. Eu aprendi teste apanhando. Eu lembro que um amigo meu, mandou um livro para mim, eu ia fazer na verdade, hoje eu trabalho na ThoughtWorks, mas eu fiz o enegrecer da ThoughtWorks. E hoje eu ainda estou tendo que estudar testes, que é uma coisa que eu ainda preciso melhorar. Então, eu fiz o enegrecer da ThoughtWorks, ali que eu vi de fato o que era testes. E aí eu apanhei, a fazer testes, porque eu lembro que um amigo mandou pra mim. Ele é Java Champion, ele é de Salvador também, hoje mora em Portugal, ele mandou para mim, ele é um cara super acessível, é um cara que se você conseguir contatar,

talvez ele te ajude também. E assim, ele mandou esse livro, que era Test Driver Development, eu só li um pouco para fazer a prova na época, na TW, que era o enegrecer, e aí eu tive uma vivência maior de teste na ClickBus, e hoje eu estou tendo uma vivência muito maior que é na ThoughtWorks, porque para você entrar na ThoughtWorks você tem que saber realmente alguma coisa de teste, e no seu treinamento você tem que demonstrar que você sabe, ou melhor, no seu, digamos assim, na sua avaliação técnica você tem que demonstrar que você sabe, independente do que seja, seja pareamento ou seja uma avaliação. No processo seletivo. **Você chegou a fazer cursos?** Não, não cheguei a fazer. Estou dando uma olhada na Udemy, e eu tenho o Alura. Sei que tem o curso de teste legal do Alura, tem outro na Udemy também. Certo. Eu dei uma olhada no começo, não cheguei a fazer. Cheguei a ver um pouco de teste no curso da Udacity. No curso da Udacity que eu fiz sobre Android, eu não cheguei a entregar meu projeto final, mas, eu cheguei até ver alguma parte disso, entendeu? Nesse curso, se não me falha a memória.

Pergunta 3: Hoje, Kotlin. No meu projeto em si não, eu só trabalho com Kotlin. Mas, tem outros projetos as pessoas utilizam por exemplo, React-Native. **Ferramentas:** Quando você fala automatizar os testes em Kotlin, você fala desde os testes unitários até o teste lá na ponta, o teste de UI? Agora em si, o que é que a gente faz, a gente, quais são as ferramentas que a gente usa. Para testes unitários, eu posso dizer, para testes unitários, não vou dizer não instrumentados porque a gente não está usando o Roboelectric, mas, para os testes unitários a gente usa o MOCKK, que é uma ferramenta muito boa, trabalha muito bem integrada com o Kotlin. A gente usa o MOCKK, o JUnit. Em um projeto eu tenho o Roboelectric, só que a gente não está usando, mas aí para testes não instrumentados, ou seja, para a gente tentar fazer um teste, fazer um teste de unidade, mas, como se a gente pegasse um comportamento de tela, e aí a gente usaria o Roboelectric. Está lá no projeto, mas, eu nunca usei. Já usei o Roboelectric para fazer provas, para fazer avaliações. Lá para cima, lá para os testes instrumentados, testes de UI, tem o Espresso, tem no projeto, mas, eu não usei ainda. E tem também, e lá a galera usa o UI Automator. Só um detalhe, lá para teste lá da ponta, lá na Bus a gente usava o Appium, mas, eu não usava, quem tinha era o pessoal de QA, que usava o Appium. Que usava tanto para iOS quanto para Android, só que eu já soube que ele dá alguns problemas, o Appium. Então, lá a gente usa o UI Automator.

Pergunta 4: Tá, vamos lá. Você fala um passo a passo para você testar a aplicação? Oh, a gente não chega a ter um passo a passo, mas, a gente tem por exemplo, no nosso READ ME, ou no GitHub ou BitBucket, ou alguma coisa desse tipo, a gente tem por exemplo, como rodar o aplicativo, mas, não os testes. Mas, a gente tem por exemplo, na nossa plataforma de CI, por exemplo, a gente tem, os testes são rodados, depois que você faz por exemplo, abre um *Pull Request*. Então tendo um teste, tem um CI rodando os testes, para saber, os testes unitários, para saber se aquele PR que você abriu, baseado acho nas coisas que você fez, se estão rodando

os testes que você fez, e também, se não me falha a memória, é rodado os testes instrumentados, se eu não me engano. No CI. No CI, e eu acho que tem alguma coisa no Jenkins também. Tem o BitWaze, acho que tem o Jenkins também, e duas pipelines para, agora eu não sei direito, qual roda o que, mas, eu sei que é rodado os testes unitários e os testes instrumentados. **Ferramenta específica para teste de regressão?** Por exemplo, a gente, deixa eu ver se eu entendi, a gente, vou liberar uma feature, certo, e aí a gente vai abrir. Não, eu acho que não, não existe não. Eu acho que são as mesmas ferramentas. Por exemplo, alguma ferramenta de CI que tem uma pipeline, que roda esse passo a passo, para ver se está pronto para ir para a deploy. Que aí é outra equipe que faz. Tipo Jenkins, tipo biterize, essa galera assim.

Pergunta 5: Vamos lá. Depende, pelo que eu consigo entender do cenário atual, é o seguinte. Vamos supor que você abriu um PR ali e você faz a feature e coisa e tal. Então, o que acontece? Você sobe aquele PR, mas os testes, por exemplo, os testes de unidade, acho que os testes instrumentados eles não são rodados todos. Mas os testes unitários, eu acho que eles são rodados todos. Pela pipeline do CI, ele roda todos os testes para saber se está funcionando. Eu só não me lembro se é rodado todos os testes que existem ou se é rodado todos os testes para a feature que você subiu. Eu acredito que seja todos os testes, ele é, ele roda todos os testes unitários para saber se todos estão rodando. E no caso esse esse processo é automatizado? Exato. Você tem uma pipeline, em um serviço de CI, e ele faz isso. Já deixa lá automatizado para que rode isso a partir do momento você submeta seu *Pull Request* para o GitHub ou para o BitBucket.

Pergunta 6: Teste de regressão, eu acho que ele é um teste que roda tudo, tipo, tá lá na ponta, você entregou, por exemplo, você faz teste unitário, teste instrumentado com Espresso, teste de UI. Eu acho que o teste de regressão, seria tipo um teste fim-a-fim, eu posso dizer isso. Seria pegar todas as telas e rodar, para ver se está tudo funcionando. Eu acho que seria isso. Você tem alguma dúvida sobre esse tema? Não. Não tenho não. Para mim é bem claro. Eu pessoalmente nunca fiz, mas, eu sei que funciona dessa forma que você falou. Você vai, vem de lá, você subiu tudo, você vem de lá até, tudo que você entregou, você passasse por todas as telas, para ver se está tudo funcionando, tudo junto. Para mim, eu só tinha esse conceito fechadinho, não tinha essa noção que você apresentou.

Pergunta 7: Olha assim. Eu acho que vai ser bem PUNK construir ela, porque assim, a gente sabe que teste de regressão, pelo pouquinho que eu conheço de teste de regressão, ele demora. Regressão que eu falo, eu não estou falando nem na questão de você rodar os testes de novo, mas, de você ter, testar toda a funcionalidade do app. Entendeu? Eu, na minha visão, demora muito, até porque testes instrumentado, testes de UI, ele demora. Então, eu acho que teria que ser algo bacana, de tal forma, que fosse um pouco mais rápido. Não sei como é que seria isso. Sinceramente, não sei. Não faço ideia. Mas eu acho que teria que se fazer,

teria que ser de tal forma que se fizesse isso um pouco mais rápido. Porque vamos supor, se o software for bem grande, ou você vai fazer isso para uma parte, porque se você for fazer por todo, às vezes nem dá. Dependendo. Você vai ter que cortar algumas partes para que você faça teste de regressão. Daquela parte ali, aí se você for fazer de um todo é complicado, pelo menos é o que eu penso aqui assim, pode ser complicado. Eu acho que é, que é isso, é você montar um mecanismo que te ajude a fazer um teste de regressão com mais agilidade. Eu acho que é mais ou menos por aí.

Pergunta 8: Mais automação nos testes do que uma linguagem híbrida, por exemplo. Aí, por exemplo, aí você fala, tipo o Appium, que para você automatizar os testes, você pode automatizar, tanto para o iOS quanto para Android. Olha, eu acho, que eu já vou falar, eu já vou partir em outro ponto. Eu acho que por ela ser uma linguagem que vem ganhando grande repercussão na comunidade Android, desde que o Google definiu ela como linguagem padrão, vem surgindo mais, e mais, e mais ferramentas para automatização referente ao Android em si. Então, acredito que, hoje, você tenha mais ferramentas relacionadas a teste, para Kotlin do que para React-Native. Eu acredito.

Pergunta 9: Se você tem uma padronização de como retestar, é uma garantia. Partindo do pressuposto que os testes em si já garantem que seu código está funcionando, massa. E você garantir fazendo teste de regressão, eu acho que é um, uma garantia a mais, de que aquilo está funcionando, e que você vai entregar um produto de qualidade para seu cliente.

Pergunta 10: Sim, sim porque às vezes a gente mexe em uma coisinha ou outra, e aqui a gente não percebe. E aí a gente pega, roda, roda aquele teste ali, sobe. Se você não tiver, por exemplo, mas está, você tinha mencionado, o teste de regressão para rodar todos os testes, quando esse PR sobe, eu não ia pegar isso, e isso poderia subir, e poderia no meu código, aparecer algum erro para o meu usuário, quando ele tivesse utilizando meu app. Então, é importante.

Pergunta 11: Eu tenho uma pergunta. Você está fazendo mestrado, você vai defender agora, vai defender depois, está perto, não está? Pronto, você tá, pelas suas perguntas, você está criando uma ferramenta voltada para teste, pra regressão? Então, necessariamente, não nesse momento, porque como eu já tô na etapa final, eu não teria tempo de, porque, pro mestrado, eu teria que criar uma ferramenta, levar pra ser testada e depois trazer esse feedback, pelo tempo provavelmente agora não é possível.

7. PARTICIPANTE 7:

Pergunta 1: Então, vamos lá, começou em 2019, eu tive a minha primeira oportunidade de trabalho como estágio na Prefeitura de Praia Grande, onde, primeiramente, eu comecei como desenvolvedor de aplicativos Ionic, então, aplicativos híbridos. Utilizando Javascript. Depois de uns três meses disso, isso foi em torno de fevereiro de 2019, e por volta de maio, eu tive uma oportunidade, na empresa em uma consultoria de São Paulo, onde eu atuaria, realmente, com os testes em relação a área de testes. E nesse ponto, como eu já tinha uma certa afinidade com o aplicativo, devido a minha experiência no estágio, o pessoal acabou me redirecionando, uma vez que eu fui contratado lá, que eu passei por todas as etapas, eles me redirecionaram para um projeto de escala global, onde eu efetuava testes para aplicativos nativos de Android e iOS. E eu também fazia nesse processo a automação desses testes utilizando o Appium. Então, foi em torno de 2019 que eu comecei na área de testes mesmo, e referente a área de testes de aplicativos.

Pergunta 2: Isso, perfeito. Então, assim, a vaga que eu entrei, ela era voltada para júnior e era num plano de formação, que a empresa dispunha pra gente aprender sobre a área de teste no geral. Então, o que é que eu fiz. No momento em que eu cheguei no projeto, que teve esse foco, eu comecei a procurar alguns cursos voltados para automação, em técnicas de teste especificamente para aplicativos mobile, certo. Então, e também convivendo ali com os desenvolvedores no dia a dia e com os outros QA testers que estavam lá comigo também. Então, eu pude adquirir conhecimento tanto na prática mesmo com o time, quanto buscando alguns cursos por fora, especificamente na plataforma da Udemy.

Pergunta 3: No caso, para a automação de testes, a gente utilizava Java, na minha antiga empresa e o pessoal de desenvolvimento utilizava Kotlin para Android e Swift para iOS. E na empresa atual que eu estou, é utilizado também Kotlin e Swift, porém, eu utilizo, hoje em dia, Javascript para automação de testes. Utiliza alguma ferramenta para automatizar teste? Sim, sim. Atualmente eu utilizo o CodeceptJS. Mas então, hoje em dia eu utilizo o CodeceptJS que é *framework* híbrido de automação, ele funciona tanto para Mobile, quanto pra web e testes de API, e ele utiliza como linguagem o JavaScript. Mas eu também já cheguei a utilizar o Appium, que acredito que é um dos mais conhecidos aí no mercado, com o Java. O Appium ele atua de forma de forma caixa preta, onde você recebe o aplicativo, cria o seu *script* e ele roda a partir daquele aplicativo na estrutura. Então foram essas duas ferramentas que eu cheguei a utilizar até o momento.

Pergunta 4: É, em relação ao documento não tem porque nos projetos que eu passei já era tudo utilizando a metodologia ágil, então, o passo a passo era muito definido por meio do Jira. Então, havia umas *plannings*, a gente dentro da *planning* montava os casos de teste, o plano de teste que a gente ia executar na determinada *sprint* e dentro disso, a gente separava quais casos de teste seriam manuais e quais seriam automatizados. E no momento em que o artefato era liberado, de acordo

com as histórias e funcionalidades, a gente efetuava o regressivo com automação e garantia os cenários novos daquela sprint atual de forma manual e exploratória. Ferramentas de automação para os testes da nova release: Isso, a gente utilizava lá na minha antiga empresa o Appium, com o Java e a gente tinha também o Jenkins para rodar, rodar esses testes em uma esteira de automação de processo e a gente também utilizava *WS Device Farm*, pra ter um pool de dispositivos em nuvem pra gente fazer essa execução.

Pergunta 5: Aí nesse caso é que a automação de teste tem uma grande utilidade para gente, então você tem duas formas de trabalhar aí. Tanto no escopo de automação de sprint menos um, que seria conforme os desenvolvedores estão fazendo a nova release, a nova versão, você está automatizando no mesmo momento, na mesma sprint os seus casos de teste passado. Isso ajuda muito a você ter uma cobertura grande e não se fazer um teste exaustivo em cima disso. Então, suponhamos, se tivessem mil casos de teste até o momento e vão ter mais cem casos de teste na próxima release. Você, no caso, só precisaria executar cem casos de teste, e os outros mil você executaria, você deixaria o seu robô, executando para você. E tem a estratégia que a gente consegue utilizar também que envolve uma comunicação mais forte com o desenvolvedor da gente ter essa automação esperando os identificadores padrão, já tudo conversado com o pessoal, então, para teste de UI tem os identificadores nos elementos. Para o teste de API a gente tem os contratos e o que é que vai ter envolvido na regra de negócio. E você consegue também ir automatizando ao longo do processo. Obviamente que com o planejamento em cima disso. Isso reduz muito nesse cenário que você comentou de não se tornar um teste exaustivo e muitas vezes fadado ao erro humano. **Mas de qualquer forma, por exemplo, mesmo que você faça isso de forma automatizada, mas normalmente vocês reexecutam, então, os casos de teste?** Isso, isso. A ideia é a gente ter uma estratégia de testes regressivos mesmo. Então, para a gente garantir que a saúde do aplicativo, ainda está OK. E nesse meio, você tem alguns cenários. Você tem os testes de sanidade e os testes regressivos. Os de sanidade são os casos de teste mais críticos que a gente tem onde o aplicativo não pode falhar e para ser considerado minimamente testável naquela versão. Então vamos supor, você tem um aplicativo onde você realiza o login, mas o login não está funcionando, você tem que ter teste de sanidade que será executado com mais frequência pra você poder garantir que aquela versão pode ao menos ir pra fase de testes e não precisa já ser corrigida antes mesmo disso, entendeu? E dentro disso a gente tem o regressivo também, que seria um teste mais completo com todos os seus casos de teste, preferivelmente automatizados. E muitas vezes os exploratórios também que são muito importantes, mas de uma forma menos recorrente do que os testes da sanidade. Mas a ideia é sempre a gente reexecutar esses cenários para garantir que tá tudo OK.

Pergunta 6: Isso, nesse caso eu posso só complementar mesmo. O teste de re-

gressão seriam todos aqueles nossos casos de teste, todos os cenários referentes as funcionalidades anteriores que a gente precisa executar para garantir que nenhuma modificação que foi feita ou nenhuma inclusão de novas features no nosso aplicativo feita pelos desenvolvedores prejudicou alguma feature anterior. Então, a ideia desse teste de regressão é realmente a gente garantir que a funcionalidade foi adicionada com sucesso e as funcionalidades anteriores do aplicativo não foram prejudicadas por isso. **Como você conheceu esse assunto, teste de regressão?** Foi muito no plano de formação que eu tive na minha primeira empresa. Esse plano de formação, ele foi muito bom, agregou muito para o meu conhecimento, mas também estudando muito através de leitura de artigos, do MEDIUM, pela conversa de outros QAs mais experientes, que eu podia conversar na empresa, então eu fui adquirindo esse conhecimento ao longo desses últimos dois anos que eu estou na área e tem dado certo esse método até o momento. **Você tem alguma dúvida sobre o tema teste em regressão?** Acredito que não. Acredito que especificamente sobre teste de regressão, não.

Pergunta 7: Acredito que isso vai muito na linha de pensar em que qualquer pessoa no time, deveria conseguir executar esses testes, principalmente o pessoal de negócio. Então, à primeira vista, assim, eu pensaria muito, que precisa ser uma ferramenta com uma usabilidade legal. Então, penso, assim, um, por exemplo, um *template* feito em HTML, onde contivesse os campos pra gente colocar quais são as features que eu quero rodar e onde elas estão contidas, no caso. E, no momento em que essa regressão é executada, ela deveria prover para gente um *report* do que falhou e do que passou também para a tomada de decisão. Então, eu acredito que falando especificamente da ferramenta, e excluindo um pouco o lado técnico disso, seria bem interessante nesse escopo, alguma ferramenta, algo desenvolvido, que permitisse a execução desses testes por qualquer pessoa.

Pergunta 8: Eu acredito que em relação a ferramenta, é indiferente o tema de qual linguagem você vai utilizar, tá. A Amazon utiliza até uma ferramenta de automação de teste onde ela se alinha com o conceito de NOCODE. Então o que seria isso? É uma ferramenta muito de Drag and Drop das suas opções, onde você não interage diretamente com uma linguagem para criar os *scripts* e assim, de acordo com esse artigo, posso até mandar pra você, depois pra você dar uma lida, funciona muito bem. Então, eu acredito que é indiferente a linguagem, tanto a para automação, quanto para a linguagem de desenvolvimento em si. É sempre uma boa prática e isso é dito mesmo pela ThoughtWorks, que foi quem surgiu aí com o Selenium, que teve maior fama no mercado, é sugerido sempre que a gente tenha, que a gente tenha, escolha uma linguagem para os *scripts*, que seja alinhada a linguagem que os desenvolvedores utilizam também, apenas como uma boa prática, mas eu acredito que a linguagem que é escolhida não, não tem tanta, tanta distinção, vai muito do seu contexto mesmo.

Pergunta 9: Não, eu acredito que é sempre importante a gente ter um roteiro de testes. Ter esses casos de teste levantados e aprovados pelo pessoal de negócio, seja no modelo cascata, ou no modelo ágil mesmo, mas eu acredito que traz muita facilidade a parte de você pegar esses testes repetitivos e realizar uma automação sim, existe, existe uma economia de tempo vinculada a isso e diminui-se o erro humano possível onde a gente pode ter nesses casos de teste. Então, eu acredito que é sempre válida a estratégia, sim, mas obviamente, depende do seu contexto, não é uma regra. De repente, você pode estar em algum escopo onde não tenha um plano de regressão definido, onde sejam testes pontuais e, mas assim, acredito que é sempre muito importante a gente ponderar sobre esse ponto, obviamente, de forma planejada e com um escopo de teste muito bem definido.

Pergunta 10: Eu acredito que sim, mas, não somente. Porque eu digo isso? Você está garantindo funcionalidades anteriores de acordo com seus casos de teste validados com a equipe de negócio e você está garantindo que, independente das mudanças que tenha, que essas funcionalidades não vão ser prejudicadas. E caso sejam prejudicados, caso o teste venha a falhar, devem ser corrigidas de acordo com prioridade, e priorizadas pelos desenvolvedores. Porém, acredito que não é somente isso que define a qualidade como um todo. Existe um ponto onde não é possível automatizar que seria realmente o teste exploratório. O que é que seria esse teste exploratório? Você vai com a sua visão de usuário, mesmo em cima daquele aplicativo e você testa ele de diversas formas, sem seguir um roteiro previamente definido. Isso é muito bom, porque tem coisas que um *script* não consegue pegar pra você. Ele é algo repetitivo, é algo que está sempre executando a mesma coisa. Então, para um momento de tomada de decisão em relação, muitas vezes, a usabilidade, a como está a funcionalidade, a cenários que, de repente, não foram mapeados, que seria algo muito específico, como, por exemplo navegar, ir e voltar em algum fluxo várias vezes e vê o que acontece. Então tudo que a gente necessita da tomada de decisão de um humano para entender acredito que ainda é muito importante. Mas sim o processo de regressão como um todo, ele é importante também para qualidade, mas não é a única forma da gente garantir.

Pergunta 11: Acredito que só agradecer mesmo, esse momento, é sempre interessante participar desse desses pontos assim, mesmo que trazendo, um pouco da visão que eu adquiri e faz a gente pensar um pouquinho fora da caixa, lembrar alguns velhos conceitos que muitas vezes no dia a dia a gente acaba indo no automático, então, só agradecer mesmo, por esse momento e me coloco à disposição para qualquer dúvida que você venha a ter depois, é só chamar também.

8. PARTICIPANTE 8:

Pergunta 1: Teste de aplicativo, começaram no passado, mas ali no final do ano,

por volta de outubro, novembro do ano passado. Logo de início, o primeiro momento foi mais a nível de evolução pessoal, de conhecer mais, a questão da automação mobile. Já trabalhava com automação web há alguns anos, mas a parte mobile ainda era um pouco obscura.

Pergunta 2: Embora eu tenha mexido com esses testes a partir do ano passado, o primeiro contato com o desenvolvimento Android já faz alguns anos. O contato com o desenvolvimento foi na faculdade, e o contato com os testes mobile foram através do curso. Uma empresa chamada CODER. Uma empresa que fornece esses cursos.

Pergunta 3: Em Java. Não utilizamos Kotlin por enquanto. **Ferramentas?** Sim, nós temos um *framework* que é pautado em Selenium. Selenium com Java. E ele fornece um caminho para automação tanto web quanto para automação mobile. Para a automação mobile o Appium.

Pergunta 4: Geralmente nos projetos que envolvem esse tipo de equação, sim. Sempre é definido um processo. Isso pode variar de projeto para projeto. Dependendo do projeto a gente pode tentar instrumentados para fazer execução. Temos projetos que tem, não necessariamente mobile, mas temos projetos que trabalham com pipelines, com a integração contínua, onde os próprios clientes podem fazer as requisições, via JENKINS, para poder executar esses testes. Pode variar muito de projeto para projeto. **Ferramenta específica para automação do teste de regressão?** Appium.

Pergunta 5: Isso varia muito com o desejo do cliente. Se o cliente estiver focando no teste regressivo, a gente pode executar a bateria completa, mas se ele tiver uma pressa maior, querer executar um escopo mais fechado, a gente pode fechar só esse resto, esse finalzinho.

Pergunta 6: Olha, um teste de regressão, assim, falando muito, muito informal. O cliente, ele passa por alguns fluxos principais, que seriam aqueles fluxos que não podem quebrar de forma alguma, são testes muito mais simples, e a tentativa dele é de garantir que não houve retrocesso daquelas features que foram corrigidas de alguma forma ou que pra ele tem uma importância muito grande. Então, é um escopo bem definido, dos casos que vão ser re executados, e sempre focando em evitar que erros antigos voltem a acontecer. **Como você conheceu esse assunto?** Olha, eu já ouvi falar de teste de regressão, em lugares diferentes, em vários lugares. Já ouvi na faculdade, já ouvi no trabalho, já ouvi em cursos. **Você tem alguma dúvida sobre esse tema, sobre teste de regressão?** Dúvida, não.

Pergunta 7: Então, eu acho que pode ser assim, varia muito daquilo que o cliente

vai desejar. Eu penso o seguinte: você tem vários tipos de testes diferentes. Você pode fazer testes unitários, você pode fazer testes de UI. Dependendo de qual nível de teste for desejado, até sobre uma visão assim de qualidade, a gente poderia até interagir com o cliente, de forma que a gente pudesse ter até mesmo explicar pra ele qual era a necessidade dele. Mas precisava ser um pouco mais específico, de quais os níveis para a gente poder ter um plano um pouco melhor para ele. Não sei se eu fui claro. Você usa o Appium hoje. O Appium hoje, atende a sua demanda de executar teste de regressão, ou você acha que ele poderia ter alguma outra funcionalidade que tornasse essa atividade de teste de regressão mais rápida ou mais fácil de ser realizada. Tanto o Appium quanto o WORD BRAD, que é, eles não são muito resilientes, então, se você pretende fazer qualquer teste, tanto a nível mobile, quanto a nível web, você precisa ter, criar como se fosse uma biblioteca, uma base de execução que não fique com uma execução puramente do Appium, ou puramente do Selenium, por exemplo, essas tratativas que são necessárias para aguardar um elemento aparecer, as esperas tanto implícitas quanto explícitas, hoje nenhuma das duas ferramentas, elas por si só, seria o suficiente pra você conseguir fazer automação completa com qualidade. Você precisa pensar de uma forma mais macro para poder atribuir a ela resiliência. Se você pensar por exemplo no Appium, ou no Web Selenium por si só, a execução dele de forma linear, como se fosse um *script*, ela pode acabar fracassando por intermitentes do ambiente que poderiam ser facilmente tratadas se você tiver pensando na resiliência destes testes.

Pergunta 8: Eu acho que linguagem funcional, pelo menos no mercado, ela não é muito difundida. Então, você conseguir profissionais que conheçam bem de linguagem funcional, ou até mesmo Kotlin, especificamente, é mais difícil. Então, no caso que, tanto em termos de paradigma, quando a pessoa conseguir mudar de um paradigma para outro, quanto de você achar alguém que já esteja pronto para o mercado. É mais difícil de encontrar.

Pergunta 9: Eu sou muito favorável a implementação de processo, que a gente consegue perceber algumas falhas que podem ser corrigidas com certa facilidade se um processo for definido. E principalmente, quando você trabalha com uma equipe, com quantidade de pessoas relevantes, não é só um desenvolvedor, é muito bom que você tenha um processo para se trabalhar, se não cada um pode pensar de uma forma diferente e acabar não tendo o mesmo, a mesma tratativa com os problemas que forem acontecendo. Eu sou muito favorável, da implementação de processos na própria automação.

Pergunta 10: Muita coisa. Por exemplo, se você estiver trabalhando com um fornecedor que ele não tenha uma boa gestão de versionamento. E aí você corrige um *bug*, e depois aquele *bug* volta a acontecer, isso vai ser encontrado no teste de regressão. Precisa ser encontrado nesse momento. Então, quando se puder identificar, garantir, que determinado problema, ele não vai ter uma recorrência, você precisa

de um teste de regressão. Ou, pelo menos, naquilo que você consegue entender que, cara, esse esse fluxo, ele não pode quebrar de forma alguma. Da mesma forma que a gente não consegue fazer testes manuais na sua quantidade total, em tudo aquilo que o sistema precisa fazer, na automação também não é diferente, então a gente não consegue testar tudo. E quando você fala de teste regressivo, a gente precisa saber exatamente quais são esses fluxos que são mais importantes, que não podem quebrar de forma alguma, e aquilo que de fato vale a pena ser incrementado por conta das recorrências de erros anteriores.

Pergunta 11: Não, eu acredito que não. Nada, eu que agradeço. O que eu tenho visto até hoje na área é que alguns clientes, não são todos, especialmente os mais novos em termos de automação, eles não entendem, como a automação pode agregar a vida deles, na questão de trabalho. Então, às vezes eles pensam em coisas que automação ela não vai atender. Elas podem ter no imaginário como se fosse resolver todos os problemas. E aí, por exemplo, a cada história, vamos acrescentar todas as histórias que aparecerem numa automação, ou vamos, fazer um teste massivo, mas aí ele nem sabe para que que ele está fazendo aquilo. Acho que o grande *feeling* é que o fornecedor ou aquela pessoa que entende de informação, ela consiga transmitir para o cliente, como a automação vai agregar e as limitações dela, para ele ter sempre os pés no chão e conseguir não ter uma expectativa frustrada acerca desse ponto. Só para eu te dar uma última contribuição, se um teste, ele nunca for, se for muito raro de ser executado, ele não deveria ir para a automação. Se ele não for um teste massante que venha fazer parte de um fluxo importante ou principal, ele também não deveria ser automatizado, tem várias, não vamos dizer regrinhas, mas boas indicações para quem está trabalhando com automação indicar para o cliente.

9. PARTICIPANTE 9:

Pergunta 1: Eu comecei a trabalhar em 2011. Então tem dez anos aí, a princípio foi com software web. Hardware software. Futuramente aí, depois de alguns anos que eu comecei a trabalhar com testes tanto Android quanto iOS. Nesses dez anos acho que eu trabalhei muito mais com automação de *back-end*, API, depois web e depois foi caindo para mobile. Mobile eu cheguei a montar a estrutura no banco Pan, quando eu trabalhei lá, tanto para Android quanto para iOS. Trabalhando com Selenium Web Driver, Java, Appium, meio que o básico.

Pergunta 2: Não, na verdade, o Google pesquisando alguns tutoriais do YouTube, mas basicamente, cem por cento pesquisa mesmo. No Google e no YouTube.

Pergunta 3: Hoje, eu acho que é React, que eu acho. Se eu não me engano.

Pergunta 4: Então, é um processo dinâmico, existe uma ferramenta de acesso a essas versões. Eu não me lembro agora o nome, mas existe uma ferramenta de gestão de versões. Então, no meu time, quando eu tô dentro do meu time, da minha *SQUAD*, quando eu tenho uma versão, eu sou avisado pelos desenvolvedores e tenho acesso a versão para instalar no meu dispositivo e testar. **Existe uma ferramenta específica para automatizar ou pra de teste dessa nova versão?** Não, a cada versão você vai ter uma feature nova, você vai ter uma funcionalidade nova, então é necessário você automatizar primeiro essa funcionalidade, para depois você conseguir colocar isso na esteira da pipeline de desenvolvimento.

Pergunta 5: Depende muito do *framework*, da velocidade de automação. Se a velocidade for legal, dá para executar tudo, você consegue dividir em vários dispositivos, ou senão, você pode fazer um MOCK teste. Você tem as duas opções. Você consegue paralelizar, tipo, 100 testes para cada dispositivo, e isso você ganha muita velocidade.

Pergunta 6: O teste de regressão é você testar todas as features que já existem. **Como você aprendeu sobre esse assunto?** Na verdade, tudo que eu sei sobre teste, eu aprendi trabalhando, porque eu entrei pra fazer uma outra atividade na Serasa Experian e já tive uma bagagem técnica. Então eles acabaram me aproveitando no suporte nível três, fiquei algum tempo e na área específica que eu testava o software de emissão de certificado digital. Então foi ali que eu comecei a conhecer o que era teste, o que era metodologia, tudo. Então assim, foi na vivência, foi acompanhando os outros colegas, foi pesquisando bastante, foi dessa forma que eu conheci, que eu aprendi. **Você possui alguma dúvida sobre esse tema?** É, especificamente teste de regressão? Não. Pelo menos no dia a dia é isso. Não tem muita mudança.

Pergunta 7: Hoje, depois que a gente cria automação no *framework* X, Y, Z, que tem N *frameworks* diferentes hoje, que a gente constrói os cenários, automatiza os cenários, normalmente a gente tem o papel, de em algumas empresas de criar a pipeline no Jenkins, subir e criar os jobs no Jenkins, referentes a esses respectivos testes, e por aí, ele já é uma ferramenta visual onde você consegue rodar seu regressivo. Então, é muito prático. **Então, você acha que o que tem no mercado hoje já te atende, não teria nada?** Sim, sim, sim. Não, não, muito bem. Atende bastante.

Pergunta 8: Não faz sentido nenhum. Independente da tecnologia, do aplicativo, a automação, ela trabalha em paralelo com a tecnologia dela. Então, você está trabalhando com Java, com o Selenium Web Driver, ou você está trabalhando com Ruby, com Python, tanto faz. É um *script* a parte que você está criando, com relação, pensando no front com os identificadores da aplicação internamente, então você só tem interação com os identificadores. Você não tem interação direta com o

código fonte. O código fonte só quem tem acesso são os desenvolvedores trabalhando com o TDD, ou teste unitário em si. A automação de teste funcional que é o mais visual, que é o do *back-end*, e tem os testes unitários, que os desenvolvedores fazem e tem algumas ferramentas que eles podem utilizar para automação nos testes deles também. Então tem essa diferença.

Pergunta 9: É, na verdade, sim, pra você colocar dentro da Pipeline de integração contínua. Para você ter meio que uma automação do processo, então, o desenvolvedor, ele desenvolve, ele consegue fazer o *build* dele, e ele tendo já um *JOB*, em paralelo de teste, então ele consegue seguir com a pipeline automaticamente até chegar no processo do P.O. Isso ajuda bastante no ganho de velocidade, no ganho também para identificar problemas o mais rápido possível.

Pergunta 10: Sim, sim, claro. Ele ajuda na velocidade e quanto mais velocidade, mais você consegue encontrar o problema antes, é claro, e resolver esse problema antes e isso ajuda muito na qualidade. Então, é o tempo que você perderia fazendo uma regressão manual, encontrando vários *bugs* em tempos bem mais longos, você consegue ganhar muito tempo fazendo isso, com isso você ganha qualidade também, em relação às entregas. Você, se você tá fazendo o manual, você vai demorar pra entregar, você pode esquecer de testar algum cenário. Então, com a regressão, ele já tá robotizado. Tipo ele já está bem controlado ali, então você bota pra rodar, ele roda todos os cenários responsáveis, pela aquela aplicação, e isso é um ganho legal.

Pergunta 11: Não, eu não tenho nada em mente agora. Eu dou monitoria também para algumas pessoas que tão começando assim, e o mundo real é completamente diferente do acadêmico. Eu sou formado em análise e desenvolvimento de sistemas. E lá teve a parte de testes, tal, mas, o que você vive no mundo, o que você passa no mundo real ali das empresas e tal, é completamente diferente. Você vai ter muitos fatores que, que vai correr contra o que você aprendeu da faculdade, porque você tem velocidade de entrega, você tem metodologia, você tem empresas que não são estáveis, então você tem, você está trabalhando o produto e amanhã você pode ter que está trabalhando em outro produto. Então, tem a questão da velocidade na automação para você ajudar a entrega mais rápida. Então, tem muitos fatores que influenciam. O que você aprende certinho dentro duma faculdade do seu dia a dia. É muito obstáculo, são muitos fatores que aparecem do nada e você não consegue seguir perfeitamente com o tradicional, o certinho. Eu até explico uma coisa, por exemplo, referente a uma certificação CTFL de teste. Eu já fiz duas vezes. E na segunda vez, eu não passei, porque eles seguem uma estrutura genérica. Ele segue uma estrutura genérica acreditando que o mundo real vai utilizar aquilo, mas na verdade no dia a dia é diferente. Não acontece isso. E o pior é que tipo, cada empresa que você passa, é uma cultura um processo diferente. Eu passei por algumas empresas e é muito diferente uma da outra. Então, não tem jeito certo ou errado. Você vai ter que se adaptar ao que a empresa faz, o que a empresa trabalha. Da

forma que ela precisa, a necessidade dela. Sim Já trabalhei em banco, hoje estou na SKY agora, com outro tipo de aplicação. Então, no banco você não pode errar muito, porque em segundos eles perdem milhões. Então, a cobrança é maior. E, assim, eu já trabalhei com testes de web, mobile, *back-end*, com aplicações de realidade virtual, agora com aplicações para *SMART TV*. Então, são contextos bem diferentes, que você precisa se adaptar, entender, conhecer, pesquisar, fazer muita pesquisa, é bastante importante.

10. PARTICIPANTE 10:

Pergunta 1: Bom, já me senti um pouco deslocado, mas bom, eu comecei a trabalhar com Android. E no início não tinha nenhum teste, nem nada, mas ultimamente, no último ano, a minha empresa criou, tem como boas práticas fazer teste nos aplicativos. Aí, foi nesse contexto.

Pergunta 2: É, na verdade, a gente tem uma prática na empresa de ter reuniões semanais, que é onde a gente troca conhecimento, a gente sempre propõe um tema, sempre propõe algum desafio entre os colaboradores. E eu uso muito Stack Overflow e documentação. É a documentação do Android e Stack Overflow.

Pergunta 3: Os dois, Kotlin e Java. Assim, teoricamente, tudo novo. todo código em Kotlin, mas como eu dou suporte no código antigo e então eu vou colocar como Kotlin, pensando nos testes, porque eu teste mais as novas funcionalidades. Mas é, Kotlin. **Você utiliza alguma ferramenta para automatizar seus testes?** A gente ainda não conseguiu fazer tudo perfeito ainda, mas estamos em vias de usar um BDD. Na ferramenta, mas o conceito de Behavior Driven Development com Cucumber.

Pergunta 4: Bom, como o aplicativo já é mais, digamos, antigo, e eu trabalho na parte legada, fica um pouco mais complicado. Mas, qual é o procedimento? Durante a criação assim, durante o desenvolvimento, de funcionalidades, quando a gente tá discutindo, já nos critérios de aceitação, a gente define os casos principais, os casos felizes, e alguns bad cases, alguns casos “não felizes”, algumas exceções. E aí, toda segunda-feira que a nossa release é sempre feita na segunda-feira, toda segunda-feira a gente tem o teste de regressão. E aí tem uma planilha com todos os testes e a cada nova funcionalidade que a gente ve, porque a gente tem um conceito de teste AB também, mas as funcionalidades a gente sempre acrescenta. E aí depois que o teste AB for validado, ou aquela função deixa de ser utilizada, a gente remove da matriz. Essa matriz, nada mais é do que um excel. Antigamente, em outras empresas, a gente já usou EXCEL READER, com o JIRA e aí definia-se o caminho, usando CURQUER, a linguagem. E aí a gente fazia desta forma.

Pergunta 5: Então, para cada nova funcionalidade, a gente testa a funcionalidade em si. Só a funcionalidade. Então a funcionalidade mil e um, a gente revalida algumas coisas que podem ser relevantes, mas todos os mil casos de teste são refeitos na segunda. E o que nós estamos propondo, o que está sendo implementado, devagar e sempre, mas, está sendo implementado, os testes de UI e os testes automatizados, e a ideia é de sempre rodar eles tanto na quarta, quanto na segunda, porque aí a gente está pensando em fazer duas *releases* por semana.

Pergunta 6: Bom, o que, o que a gente define como teste de regressão é testar todas as funcionalidades do sistema. Desde o princípio que foi criado até hoje. Não desde o princípio, porque novas funcionalidades, acabam eliminando antigas. Mas testar todas as funcionalidades ativas no sistema, de forma sistemática. **Esse conceito de teste de regressão, onde você aprendeu na no trabalho mesmo, em algum curso?** É, eu já trabalhei um período como tester, não aprendi tanto nessa época, mas no meu último emprego, digamos assim, a gente tinha que fazer testes. E como o software era muito grande, aí, tinha sempre esse conceito de criar o conceito de teste, a gente não chamava de regressão, mas teste para casos legados. E aí entre cursos e palestras que a gente faz internamente, eu fazia também, aí falava, esse é o conceito de teste de regressão, é testar as funcionalidades antigas e como as novas funcionalidades afetam elas. Você acaba pesquisando um pouco, por curiosidade. **Você possui alguma dúvida sobre esse tema? Fecha regressão? Ou é pra você um tema tranquilo?** Assim, eu confesso que o meu conhecimento é leigo nesse sentido. Digamos, não vou falar leigo, é raso. Eu já comecei a estudar, por exemplo, a parte de fazer testes automatizados, e acaba indo pra esse conceito, de estudar mais todo o conceito ou tudo que engloba, mas acabou que entre uma demanda e outra, você vai empurrando para frente.

Pergunta 7: É, bom, eu quando eu penso em ferramenta, aí eu vou usar termos mais técnico, não sei se isso vai agregar ou não, mas quando eu penso em ferramenta eu penso no conceito de macro do Excel. Eu sempre penso no conceito de macro do Excel. Que é detalhar de forma relativamente clara, todos os passos que foram feitos. E quando eu penso em uma ferramenta que poderia me ajudar, seria esse mesmo conceito. Hoje o Expresso tem isso no Android, só que é só no Android. E acaba que a equipe de teste fica sobrecarregada com uma ferramenta só. Respondendo a pergunta, o que eu penso? É conseguir guardar de alguma forma os passos, ou, eliminar repetição sistemática, e ter isso de forma automatizada e, mas, de uma forma, onde eu possa validar que cada efeito foi feito no cenário mais próximo do real. Acaba que como desenvolvedor, eu faço muito MOCKS, em situações fictícias. E aí o resultado muitas vezes é maquiado também. Pode acontecer algum erro pelo estado. Um dos produtos que a gente tem que é bem recorrente, é o conceito de um teste unitário muito bem definido, mas não tem o teste de UI bem definido. Então, a funcionalidade funciona como ela deve funcionar, mas quando troca de estado, por exemplo, passar da classe A para classe B, eu acabo tendo alguma falha

por não conhecer o produto muito bem. Ou por fazer alguma coisa errada, mas as funções por si só elas respondem de forma correta. Então, quando eu penso em uma funcionalidade, era conseguir fazer com que esse fluxo completo fosse feito, algo próximo da integração. Mas hoje ainda não conheço nenhuma ferramenta boa o suficiente pra isso.

Pergunta 8: Não. Não porque eu conheço o Robot *framework*, se não me engano é esse o nome. É porque eu sempre confundo com o Robot Mongo, mas acho que é Robot *framework*, que faz em Python, e quando o código está bem definido, ou pelo menos a interface gráfica está bem definida, ele tem uma performance muito boa, e reduz muito o trabalho. Eu sei que tem também o Appium, se não me engano, que pode usar com isso. Mas eu nunca usei. Então eu não acho, eu acho que como desenvolvedor Android, eu tendo a ir para o, para o Kotlin, por ser, ser o meu dia a dia. Eu não troco o contexto, entre entre duas linguagens. Mas se eu, se me falar assim, escolhe uma ferramenta que vai ajudar todo o cenário, eu acho que alguma ferramenta híbrida seria muito mais relevante.

Pergunta 9: Bom, é, eu acredito que sim. Eu vou fazer um comentário, desculpa. Não se sinta ofendida. A pergunta me faz querer responder sim. Apesar de eu achar que sim é a pergunta que me faz querer responder sim. O que é que eu acho? E isso eu tenho questionado muito nos últimos dias por estar fazendo muito teste de UI. A gente tende a fazer o teste, eu pelo menos, como desenvolvedor, a gente tende a fazer um teste viciado, para sempre dar certo, porque eu estou validando a minha funcionalidade, então eu penso muitas vezes no que eu desenvolvi. Por que eu acho que a sistematização seria boa, mas não só pelo lado do desenvolvedor. Seria mais uma validação. Eu seria, eu acho, que eu descubro que tenho que fazer o teste de UI e tenho que fazer teste de regressão a parte disso. Algo como caixa preta. Mas na minha cabeça eu sempre fico na dúvida de como é que eu valido, eu ouvi muito essas afirmações na minha vida, então eu sempre fico com ela na minha cabeça, como é que eu valido que eu estou fazendo a coisa certa e que isso é a coisa certa a ser feita. Sim. Então eu sempre fico pensando nisso, eu acho que a sistematização, ela é boa para validar a coisa certa, mas não me garante que a coisa que eu fiz corretamente o que deveria ser feito. Não sei se eu consegui ser claro nesse sentido.

Pergunta 10: De novo, me faz querer responder que sim, porque me colocaria em uma situação constrangedora se eu falasse que não. Mas eu acredito. Não, é porque eu, realmente, acho que sim. Depois eu faço esse feedback. e depois me repete essas perguntas, por favor. O que é que eu acho? Eu acho que ela valida, dependendo da complexidade do problema, ela aumenta a confiabilidade. Sim, mas ela não consegue cumprir todos os casos. A medida que o programa vai crescendo, eu acho que vai se perdendo. Se não for bem estruturado ou automatizado desde o princípio. E tendo uma manutenção regular que normalmente na minha experiência não vejo isso acontecer, ou nunca me foi dito assim: “não isso aqui nós temos que

ter uma reunião para ver o que foi, vou usar deprecado”, não sei se essa palavra existe em português, bom, deprecado. Aqui em Portugal o povo me xinga muito, se essa função deixou de existir ou não. Eu nunca vi esse debate acontecendo pelos menos, nas minhas empresas. Então, por que que eu falo que a resposta me leva a falar que sim? Porque sim, é um filtro. É. Que seja de carvão, ou seja, um filtro grosso. Mas eu só acredito que o teste de agressão seria muito válido, se ele fosse bem mantido. E eu não, eu nunca vi esse cenário. E também ter a manutenção. Eu posso dar um exemplo da minha empresa que aconteceu semana passada, a gente fez uma release e uma das funcionalidades mais bestas do meu aplicativo, é busca. E passou no teste de regressão, porque muitas vezes o Estado interfere no resultado. E fazer a busca no estado A, e tentar resultado, e fazer a busca no estado B e tentar o resultado. A busca funciona? Funciona. Mas ela funciona igual nos dois casos? Não. E aí tivemos esse problema. E me fez questionar isso: olha, o teste de regressão ainda carece de mais refinamento, e isso envolve tempo, porque o produto é grande. Eu posso fazer o adendo agora, ou faço no final. Uma frase do meu professor de faculdade, e me faz lembrar até hoje essa frase é: um bom gestor com um bom time de desenvolvimento, inclusive, todos do desenvolvimento, por muitas vezes eles fazem essa segregação, mas sim jogando todos, não garante um bom software. É isso que eu sempre penso. Um bom gestor, com um bom time de desenvolvedor não garante um bom software. Um mau gestor, com um mau time de desenvolvimento, com certeza não é um bom software. Então, o teste de regressão para mim, ele passa por isso. É. Não quer dizer que todo mundo aqui é bom, quer dizer que vai sair um bom, mas quer ser um pouco melhor do que o ruim. E por isso que eu acho que a regressão, ela é tão trabalhosa quanto a criação de novas funcionalidades. Mas as empresas não dão esse valor.

Pergunta 11: A forma como você fez a pergunta agora, dando feedback para você. É, a forma como você me colocou a pergunta, a forma como ela foi montada me faz, é tipo assim, você vê a importância em fazer, você vê a importância de fazer a secreção para que aumente a qualidade do software? Isso. Quando você coloca qualidade de software na pergunta, importância e qualidade no software? Na minha cabeça, talvez, seja só na minha, mas eu acho que não, ele já cria a sensação de que o importante é a qualidade. E testar sempre, aumenta a qualidade. É esse o dito popular, digamos assim. Claro que a gente sabe que isso é uma verdade, mas o dito popular, mesmo que a gente não faça, é o dito popular. Então, me faz pensar assim, se eu responder que não, eu estou indo contra uma base gigante, e eu não vou ter fundamentos suficientes para isso. Então, é mais fácil eu responder que sim, porque eu só vou com a maré. Eu não sei se era esse o conceito de tentar entender a lógica do programador ou não. Eu não sei o tema da sua tese, nem nada, mas se for esse o tema, a sua explicação me força a me falar que sim, pra não ter que se confrontar uma pergunta mais difícil e eu não saber responder. Por mais que eu concorde, acho que nas minhas respostas eu fui claro em responder que, sim, eu vejo muita importância. Mas numa pesquisa com mais pessoas, talvez a pessoa fale

que é importante, mas, não consiga dissertar sobre isso, porque, na verdade, ela acha que não é tão importante.

11. PARTICIPANTE 11:

Pergunta 1: É, então, na verdade, eu era analista de sistema, de TI. A parte de teste meio que eu fazia, mas não era minha atividade principal. Eu comecei a fazer atividade principal, mas já vindo com essa experiência de analista, quando eu comecei na CIDIA, e lá a gente fazia o teste dos *smartphones*, e nos *tablets*. Então, foi a partir daí que eu fiquei focado nesta atividade de testes.

Pergunta 2: Inicialmente foi por conta própria, mas logo quando eu entrei na CIDIA, lá tinha treinamentos e tudo mais. Eu tive uma orientação mais específica e posteriormente eu comecei a me preparar para tirar o CFTL, e aí já ficou mais profissional, minhas atividades.

Pergunta 3: É uma linguagem híbrida. O pessoal usa React-Native nos projetos. **Existe alguma ferramenta para automatizar os testes que vocês utilizam?** Tem, só que a gente ainda tá, ainda tá estudando, que é o DETOX. Porque a automação tá mais com os desenvolvedores no momento no Android. Eu já fiz alguns testes com APPIUM mas no momento eu não uso com uma frequência assídua.

Pergunta 4: Tem, geralmente. Depende muito do caso, do tipo de teste. Mas, geralmente, a gente pensa em um conceito, assim, de entrega contínua, passando ali por alguns testes unitários, integração. Chegar ali no nível de sistema, e se for necessário fazer um teste manual para depois disponibilizar essa produção. **O teste da nova *release* é automatizado ou manual?** Não, é como eu falei. A gente tenta usar o JET/JUnit nos níveis unitário e integração. E de sistema, uma boa parte é manual.

Pergunta 5: Dependendo da situação talvez se faça necessário, mas acho que de modo geral, um MOCK teste eu acho que resolveria. Poderia selecionar aí alguns pra ver as funcionalidades principais e acho que resolveria. É claro, é bem genérica essa pergunta, mas se tratando assim de um curto espaço de tempo, eu ficaria com o MOCK teste. E esse processo, por exemplo, de escolha? Vocês fazem manual ou o processo de escolha de quais testes vocês precisam executar? Então, nesse caso, a gente tem que ter um alinhamento, como os desenvolvedores, stakeholder, então, não é assim, tirado do nada. Mas alguns testes principais são frutos de dados que a gente já tem, de algumas falhas recorrentes e tudo mais. Então, se por exemplo, se tratando de um teste de regressão em algumas áreas que as falhas são recorrentes, é muito provável que a gente já tenha alguma automação, e em alguns outros casos é feito manual, pensando que tenha sido uma funcionalidade nova, que ainda não

tenha alguma automação. Trabalhado mais ou menos assim, mas tem um alinhamento, assim, dos times, não é só uma decisão do QA.

Pergunta 6: Teste regressão parcial, seria tipo, testar pela funcionalidade. Agora me fugiu o conceito, mas eu sei que tem teste de regressão parcial e o completo. Mas o teste de regressão completo seria testar a aplicação toda, e o parcial, focar em algumas funcionalidades.

Pergunta 7: Cara, eu até não sei se é generalizada, também não sei se eu tenho que aprofundar a pesquisa, mas para sistemas híbridos é sempre uma dor de cabeça. Sempre, não é fácil, por algum motivo. Às vezes, assim, por exemplo, se eu vou fazer um teste web usando Selenium, parece que tem uma documentação, tem um suporte, tem muita coisa que ajuda aqui, o desenvolvimento acaba sendo rápido do teste. Para Android eu não sinto essa facilidade assim, a gente tem sempre muita dificuldade. Então, eu acho que a primeira coisa que a ferramenta, a ferramenta é bom que tenha uma boa documentação, que tenha espaço pra comunidade ir reforçando as falhas e tudo mais, ter um diálogo ali com a galera, e a nível funcional que ela suporte sistemas híbridos. Porque, às vezes tem uma uma ferramenta que são muito específicas e se a pessoa usa um React-Native às vezes fica confuso.

Pergunta 8: E na minha opinião assim, na minha experiência ainda, eu encontro mais dificuldade. Então, às vezes a gente acaba apelando para o manual, por questão de prazo, e tudo mais.

Pergunta 9: Antes de tudo, garantir a eficiência. Acaba trazendo mais tempo pro teste, focar em outras coisas e tal.

Pergunta 10: Existe sim, uma leve, mas se tratando de Android tem a questão dos modelos, das versões. Então isso acaba aumentando uma complexidade, assim, então é difícil falar porque eu fiz aqui, que realmente vai funcionar em todos os modelos, todas as versões. Porque são muitas. São muito distribuídas aqui nesse lado. Então, no cenário geral eu vejo uma boa relação, mas no cenário prático de muitos tipos de usuário que tem celulares de última geração e celulares mais aquém, principalmente no poder de processamento, aí depende. A gente tem casos, por exemplo, de teste de performance/regressão para determinada versão, para determinado dispositivo que tem um processamento tal, todos passam tranquilo, mas, aí entrando em outros modelos de processamento mais baixo, aí acaba tendo outras complexidades, por causa de uma performance mais baixa, e tudo mais. E aí acaba meio que trazendo um retrabalho, porque então a gente vai ter que fazer a verificação ali manual, é montar um escopo pra, pra testar aquele, aquele tipo de cenário, e aí meio que o teste de regressão funcionou, em uma situação, mas acabou não, não considerando tudo. Então, eu vejo assim que depende, no cenário ideal é

sempre bom ter, mas na prática não garante pra todos. Não necessariamente ter, vai garantir para todos os cenários.

Pergunta 11: Acrescentaria essa questão que eu acabei de falar, porque eu acho que para os testes, acho que todo mundo passa por isso, celular de alto poder de processamento e baixo poder de processamento, é um outro tipo de teste basicamente. Até o teste manual, quando eu trabalhava na CIDIA a gente mexendo em vários tipos de dispositivo. Até o teste manual, funciona diferente, porque o celular é meio lento. Então, vai consumir mais tempo do teste e tudo mais. E aí, o engenheiro de requisito tem que pensar já diferente pra quem vai fazer isso. Eu acho que acrescentaria isso, porque Android é meio genérico, no sentido porque tem muitas versões. Eles têm em diversos dispositivos. Eu acho que trazer isso seria, talvez, aprofundar mais algumas questões. É interessante que é uma análise combinatória imensa? Você estava falando, eu estava fazendo essa reflexão, porque, por exemplo, você faz um aplicativo web, teoricamente, quando a gente programa para a web, ainda tinha aquela questão, ah, você está programando, pra rodar em qual browser. Vamos supor, internet Explorer, Mozilla e Google e Chrome. Mas quando a gente vai pra Android, você tem, sei lá, N *devices* diferentes, e N versões do Android. E aí, você vai ter uma combinação, dessas várias possibilidades. **Nossa, é complexo? Demais. Na prática, é bem mais complexo que na teoria?** Sem dúvida, sem dúvida. Porque, tem eu não sei dizer qual, agora é de cabeça, mas tem questões, por exemplo, de que o Android é uma versão bem atualizada, mas o *device* ele meio que foi obrigado a aceitar aquela versão e ele fica lento, já começa a ficar. Então, é muito bem, bem complexo mesmo. Acaba sendo até genérico demais pra falar Android.

12. PARTICIPANTE 12:

Pergunta 1: Olha, na área de testes, eu estou há três anos, testando. Eu já iniciei testando web e mobile juntos, que era um aplicativo para bancos. E é isso, faz três anos. Acho que foi três anos, acho que é três anos mesmo. Faz dois anos que eu estou na empresa atual, que completou agora. Isso. É isso mesmo.

Pergunta 2: Olha, eu peguei mais foi por curiosidade mesmo, quando eu entrei na área de testes eu não sabia nada mesmo. Eu entrei de gaiato, porque eu fazia parte de um suporte técnico anteriormente, de estar, de trabalhar nesse ambiente de testes, e aí foi mais por curiosidade pra conhecer o que um analista de testes pode fazer até eu entrar na área. Aí eu estudei pelo YouTube, fui vendo lendo artigos sobre. Só.

Pergunta 3: Então, eu posso falar que Kotlin, em uma boa parte em Kotlin. Existem ferramentas para automatizar os testes: Na verdade, a gente automatiza, a gente não está automatizando. Você está falando de teste unitário, ou de teste

de front mesmo? Geralmente, o que é que a gente faz. A gente faz os testes, se for para testar essa parte de front, essa parte do analista de teste, para poder fazer a automação, a gente utiliza Appium junto com o Java. O programa é java e a gente utiliza Cucumber também para poder fazer essa automação. E a parte de testes unitários, eu não lembro muito. Fica mais com desenvolvedor, os testes unitários? É, fica com eles. Mas, eu tenho, eu acompanho algumas coisas com eles, mas eu não lembro qual que é o nome. JUnit.

Pergunta 4: Não, a gente não tem nenhuma padronização de testes. O QA, sempre é, ou tester, não não sei como você dá o nome, mas, a gente é livre de ferramentas. Então, a gente faz os nossos processos, que a gente vai realizar testes, entendeu? Então, eu, por exemplo, eu sigo os meus cenários, que eu acredito que são, é importantes para a minha aplicação, entendeu? Tem uma parte automatizada que, que são os mais importantes mesmos, os mais importantes cenários, e algumas coisas eu acabo fazendo manual, mas mesmo para ver se não quebrou alguma coisa, se está queixando alguma coisa, mas bem pontual, esses testes. Então, no caso você falou, por exemplo, você segue esse fluxo, outro, tipo, se for outro analista, ele pode criar um outro fluxo para eles, não é padronizado, no caso? Isso. Eu sigo esse fluxo, da minha equipe, entendeu? Porque a gente trabalha em equipe, cada QA em uma equipe, entendeu? E aí ele tenta ver o que é melhor para o time dele. A gente tem essa autonomia. **Você utiliza alguma ferramenta específica para automatizar essa etapa do processo, ou é a mesma que você utiliza anteriormente como Appium e Cucumber? É a mesma.**

Pergunta 5: Bom, eu selecionaria os casos que têm maior impacto para o usuário. Que são, realmente assim, imprescindíveis para dar erro, por exemplo. A gente já faz isso na automação, porque a gente não automatiza tudo. Tipo, não dá. Não é legal automatizar tudo. Porque querendo ou não, a gente gasta tempo pra isso, e também para executar também um bom tempo também.

Pergunta 6: Então, o que eu acredito que seja, que são testes que eu vou aplicar, tanto na versão mais recente do software, são aqueles casos que eu faço testes com aqueles casos de teste mais novos, na versão mais recente, para ver se não surgir, os novos efeitos, ou até mesmo, eu posso pegar para poder testar antigas *releases*, por exemplo, fazer, passar por todo o fluxo. É porque caso ele mexer, fazer alguma manutenção, ou alguma coisa assim, eu tenho que garantir que todo o fluxo, ele está funcionando normalmente. Ai eu tenho que fazer os testes para evitar, essas aplicações críticas, assim, para e testar com a mesma frequência, essas funcionalidades que foram alteradas. **Como você conheceu este assunto? Sim, no trabalho. Possui alguma dúvida sobre o tema? Não.**

Pergunta 7: Nossa, nunca parei para pensar nisso. Nunca parei para pensar, mas, então. Eu importo muito com a usabilidade do usuário. Assim, para que não tenha

impactos com eles, mas, deixa eu ver aqui, mas, eu não sei responder essa pergunta. **As ferramentas que você usa hoje para testar as novas *releases*, elas te atendem bem?** Sim. Eu não vejo nenhum problema com elas não, sabe. Eu acredito que está funcionando bem. Eu não vejo outra proposta para poder mudar agora, qual é sabe, porque eu acredito que cada, para o que eu estou fazendo, para mim é essencial para o que eu quero, sabe. Agora, por exemplo, se eu for testar outra coisa, ai eu vou partir para uma outra ferramenta, que eu acho que não tenho um caso, tipo assim: “a, eu quero uma que faça tudo de uma vez”, eu acho que é desnecessário.

Pergunta 8: Para mim, eu não vejo diferença não, para ser sincero. Porque, por exemplo, eu programa em Java, eu posso fazer as coisas também em Kotlin, e não tem diferença. Eu acho que não, que não tem diferença nenhuma isso.

Pergunta 9: Nossa a importância é grande. Eu acredito que a gente tem que ter uma organização referente ao que a gente quer testar, o que é importante naqueles testes sabe, qual é a precisão, e saber, o impacto daquilo, que pode ocasionar de usabilidade para o usuário, ou até mesmo para o próprio cliente sabe, que é o responsável pelo aplicativo.

Pergunta 10: Eu acredito que uma porcentagem boa, sim. Tipo assim, não total, mas uma grande parte é bem favorável. Porque igual eu te falei, quando a gente automatiza, a gente automatiza cenários que a gente acredita ser o principal para aquela aplicação. Então, se eu não, por exemplo, eu não vou automatizar todos os cenários, então pode ser que destes outros cenários que eu não acredito ser tão importante, possa ser que tenha algum probleminha lá e eu não vou conseguir pegar porque não foi automatizado, entendeu? Então, eu acredito assim que é legal, mas não é completo, não é cem por cento de garantia.

Pergunta 11: Não, achei legal, achei bacana, me fez pensar algumas coisas, que eu não tinha pensado assim, sabe. Sobre o que pode melhorar sobre, em relação a essa parte de ferramenta, de automação. Eu achei bacana a pergunta, porque é algo que eu nunca parei mesmo pra pensar. É que a gente acaba ficando automático, refém de uma coisa, tipo, por exemplo, de ferramentas que a gente possa, como é que eu posso dizer, que agente possa automatizar daquela forma, entende? A gente nunca viu: “A será que pode ser melhorado essa ferramenta de algum jeito?” Nunca passou por essa curiosidade assim, apenas pegar e usar, entendeu? E também tem muita coisa para melhorar também. Assim, tanto na academia, quanto no trabalho. Por exemplo, eu passei por duas universidades, é que não tem nada falando sobre testes, sabe, a fundo. Então, isso eu acredito que deveria ser acrescentado nas universidades, ser mais falado, entendeu? Por exemplo, eu estudei na Universidade Federal daqui de Uberlândia e hoje eu estudo na IFTM, que é o Instituto Federal do Triângulo Mineiro. Na primeira eu fiz Ciência da computação, só que

eu não finalizei, faltava um ano para formar, eu preferi sair do curso. Hoje eu faço licenciatura em Computação. Então, nenhum dos cursos teve matéria referente a isso, sabe, referente a testes e tudo mais, sempre focado ou na parte administrativa ou na aprendizagem de desenvolvimento, sabe? Então, tem algumas coisinhas que eu acredito que deveriam ser acrescentadas. É uma outra coisa, por exemplo eu, para poder melhorar, muitos dos meus testes, para poder compreender um pouco, assim, no início, o que é o Android, e tudo mais, aí eu tive que forçadamente puxar matéria de mobile para eu poder entender, sabe. Porque antes eu não sabia nem o que é que era um componente, por exemplo, eu ficava assim: “componente, como ele é formado, como assim?”, para poder entrar na minha cabeça. Então, foi algo assim que eu tive que resgatar do, correr atrás no ensino para poder melhorar um ponto que seria desenvolver mobile para que eu possa testar e melhorar a minha qualidade como testador ou até mesmo como analista de qualidade para aquele meio. Por exemplo, a gente tem uma matéria de engenharia de software que fala muito pouco, resumidamente, sobre teste de software. Para você ter ideia, eu de todas essas universidades que eu passei, nenhuma universidade virou e falava sobre o mercado, sobre a profissão de analista de qualidade ou sobre teste e tudo mais. A gente tinha uma visão de testes unitários, sabe? A gente nunca tinha uma visão que teriam pessoas para realizar testes. Então eu não sei se hoje tende a melhorar e se essa parte da academia, para poder passar, como é que fala, a ter esse conhecimento, sabe? Tipo, a passar esse conhecimento pros alunos, sabe? Que eu acredito que é uma deficiência tremenda. E o quão, analista de teste, ou até mesmo o analista de qualidade de software é importante para um bom funcionamento, e para qualidade do que vai ser desenvolvido. Então, a gente, eu por exemplo, eu observo que a qualidade mesmo de testadores e de analistas de testes, vem muito pela formação quando inicia no trabalho, para ser sincero, porque antes disso, realmente, a gente fica muito ligado a testes manuais, por exemplo, e até mesmo a procura de testes automatizados, é muito grande e você não consegue encontrar com facilidade é QA, ou até analista de testes, que automatiza com facilidade, porque ou não teve um contato tão específico com linguagem de programação ou até mesmo não tinha na cabeça a importância de saber automatizar isso futuramente, sabe? QA a gente já consegue ver hoje, mas, desenvolvedores sendo contratados para automatizar o sistema. Não tendo, por exemplo, uma visão analítica, sabe, sobre qualidade de software. Na minha visão, por exemplo, um analista de qualidade, ele consegue ser um desenvolvedor com facilidade, facilidade que eu falo assim: “a, eu preciso desenvolver, e tudo mais, estudar, tudo mais e desenvolver.” E ele não vai deixar aquelas práticas, por exemplo, de qualidade para trás. Ah, eu preciso fazer isso e tal, isso. Já um desenvolvedor, já com o tempo de trabalho, pra ele ser apontado como um analista de qualidade, na verdade, não um analista de qualidade, mas como automatizador do processo, sabe, ele não vai ter uma visão tão coerente quão um analista de qualidade pelas formas de pensar, pelo, como eu posso falar, pelo tempo de trabalho que um analista de qualidade teve, entendeu? Então o conhecimento sobre o que esperar dessa qualidade aqui, a, o que eu espero que esse software, sei lá, que seja um aplicativo, espera para que tenha qualidade, entendeu? Então, assim,

são muitos pontos de interrogações sobre esse assunto. Mas é uma diretiva assim, que eu acredito, sabe? Que é mais fácil, um QA se tornar um programador sem seus vícios, mas tendo suas preocupações sobre qualidade, sobre a qualidade daquilo que está sendo desenvolvido, do que um desenvolvedor, automatizar e esquecendo de teorias importantes sobre qualidade de software.

13. PARTICIPANTE 13:

Pergunta 1: Bom, eu trabalho com manutenção de sistema, atualização deles, já tem bastante tempo. Principalmente Android, tem quase dez anos que eu trabalho com Android. E como eu trabalhava com sistema de força de vendas e havia muito cálculo e mudança de tributação, sentia necessidade de automatizar esses cálculos, porque havia muitas mudanças constantes no código e validar isso era um trabalho herculino. Então, o jeito que a gente achou foi começar a implementar testes.

Pergunta 2: Comunidade em geral, STACK OVERFLOW, YOUTUBE, pequenos Meetups e palestras. A gente começou a ver algumas coisas relacionadas ao assunto e fomos procurando em sites, tutoriais.

Pergunta 3: Prioritariamente Kotlin. Mas tem algumas coisas em Java ainda, mas a prioridade é Kotlin. **Existe alguma ferramenta para você automatizar testes nessa linguagem?** Sim, ele possui a ferramenta, o Kotlin, o próprio Android Studio possui um player do expresso, onde você consegue gravar as interações na tela e ele roda aquilo automaticamente. Eu uso muito pouco essa parte, mas a maior parte do processo é via código mesmo, é usando testes do Gradle, e pipelines chamando elas.

Pergunta 4: Sim, temos um passo a passo das coisas que tem que ser feitas. Algumas coisas são os testes unitários mesmo, que é, para cada funcionalidade nova tem implementado. E daí vem validações minhas, como desenvolvedor mesmo, que aí eu faço validação em umas quatro versões do Android antes de subir na loja. Agora a gente tá tentando usar uma FARM de dispositivos para fazer, começar a rodar alguns testes automatizados, mas ainda tamo, estamos encaminhando pra isso. Esse processo é possível ter acesso a uma cópia ou um processo confidencial? É um processo confidencial. **Existe alguma ferramenta específica para automatização dessa parte de teste de novas versões, ou vocês usam, por exemplo, no caso, o próprio *framework* do Android, você falou de teste unitário, que você usa a própria ferramenta do Android? No caso, o JUnit que você roda. Você utiliza essa mesma ferramenta para testar novas versões ou alguma ferramenta específica?** Sim. Há um estudo para a adesão de uma empresa terceira para esse serviço, mas ainda há estudo. Tem a questão do Teste Farm da WS e o do Firebase. Então, a gente está estudando ainda qual

vai ser mais viável adotar. No momento revalida com as mesmas ferramentas que utilizam no processo de desenvolvimento.

Pergunta 5: É tudo retestado.

Pergunta 6: Bom, basicamente verificar a compatibilidade entre as modificações da aplicação nas versões anteriores. Eu vou verificando as modificações aqui, na versão anterior, como é que o comportamento de cada uma delas é voltando no tempo, eu não tenho certeza se é isso, já deixar claro. **Como você conheceu esse assunto, na sua atuação na empresa ou em algum curso?** Mas na atuação na empresa mesmo. **Você possui alguma dúvida sobre esse tema? Teste de regressão?** Sobre testes, muitas. Eu conheço muito pouco da área de teste. Foi mais questão prática mesmo, de pegar coisa e implementar. **Alguma dúvida sobre especificamente teste de regressão? Algo assim que você gostaria de saber ou perguntar?** Nesse assunto acho que eu sou tão leigo que eu acho que eu não tenho pergunta. Eu precisaria estar mexendo mais para entender.

Pergunta 7: Bom, com relação a ferramenta de testes que eu vi, a dificuldade que eu tive com implementação de testes de interface e tudo mais. Uma coisa que eu sinto de grande dificuldade, principalmente, no Android é mudança de comportamento da interface. Dada as atualizações constantes, do Android, a gente tem muita atualização que gera mudança no comportamento propriamente do widget, pede configuração extra, pede modificação extra, implementado uma nova camada de segurança, é feito alguma coisa extra. Então, toda vez que você tem que fazer uma atualização para API nova, onde você vai começar a contemplar dados de uma API nova, você fazer todas essas atualizações e verificar se nas anteriores quebram, é muito complicado, e eu não sei se isso entra nessa categoria da pergunta. É porque as atualizações que todo ano tem que ser feito, constante no Android, é a atualização da API alvo do aplicativo. É muitas mudanças que entram nas APIs novas. Então tem mudanças de comportamento, componentes que deixam de ser utilizados e são substituídos por componentes novos. E muita das vezes você vai fazer essa atualização no código, você codifica pouco, às vezes você troca a versão da biblioteca e muda uma chamada ou outra, mas você não tem, tipo, muita das vezes, como prever o comportamento que aquilo vai ter. Se o comportamento se mantém igual ao comportamento anterior ou se você vai ter que fazer mais coisas para manter o comportamento, sabe.

Pergunta 8: Bom, eu acredito que o fato de usar linguagem nativa, você ganha algumas vantagens. Mas, no final, eu acho que o resultado tende a ser o mesmo.

Pergunta 9: É muito importante, entendeu. É igual, sistematizar é uma ideia que as pessoas têm de burocratizar, mas não é burocratizar. Ela ajuda muito a entender

como as coisas funcionam. E eu acho que tem que ser aplicado sempre, tem que ser bem definido os passos de cada coisa para entender o fluxo. Para atualização de conhecimento, para treinamento de novos funcionários, para um monte de coisas.

Pergunta 10: Diretamente ligado. Eu já vi muitas vezes, muitas modificações, implementação de coisa nova, estragar coisas que não tem nada a ver com aquela modificação. Então, eu acho que é algo essencial.

Pergunta 11: Bom, pra mim tá bem tranquilo. Acho que não.

14. PARTICIPANTE 14:

Pergunta 1: Bom, então, eu já sou desenvolvedor há algum tempo e migrei para área mobile tem dois anos. E na área de mobile eu fui de iOS para Android, e hoje eu, eu atuo somente no Android. Atuando como desenvolvedor, a gente não só desenvolve novas funcionalidades ou constrói aplicações, mas a gente também se envolve em testes. De que forma. Geralmente um desenvolvedor de Android faz testes. Geralmente faz testes unitários, e a cada funcionalidade que a gente constrói, a gente constrói também testes unitários, para com que valide as informações que vão transitar ali naquela aplicação. Então, geralmente, o desenvolvedor em si ele utiliza da ferramenta dos testes unitários para fazer os testes. São testes não muito visuais, mas sim de funcionalidade. O que é bem de mais baixo nível do que simplesmente de fluxo de tela. Vamos dizer assim.

Pergunta 2: Então, eu sou formado em ciência da computação, eu formei pela Universidade Fumec, só que na universidade em si, não foi muito utilizado o teste lá não, sabe. A gente não passou por esse tipo de aprendizado não. Na verdade, eu acho que o teste em si é uma área específica da computação que eu acho que é bem desvalorizado até, pelo mercado, pela academia. Porque hoje já está sendo criado uma uma cultura já mais de ter um QA, que vai poder fazer aquele serviço, mas antes eram feito por pessoas que ou às vezes não tinha muito conhecimento da área, ou eram as próprias pessoas que faziam. Então, geralmente geram testes viciados. Eu também, pra aprender a fazer alguns tipos de testes, eu tive que me recorrer a cursos extracurriculares, vamos dizer assim, em sites de cursos, tipo UDEMY, UDACITY, e no próprio serviço mesmo com pessoas dando suporte, ensinando mesmo o que é que fazia. No começo eu não tinha muita certeza do que eu estava fazendo, e depois a gente vai fazendo, entendendo para que que serve, para que eu estou fazendo aquilo. Então foi mais ou menos assim, eu aprendi mais mesmo no serviço, com a ajuda das pessoas mais experientes, e buscando informação nesses cursos extras que a gente diz. Nenhuma certificação não. Não busquei nenhuma certificação.

Pergunta 3: Eu utilizo Kotlin. O teste unitário em si, ele tem a responsabilidade do próprio programador, durante a implementação, então, eu não consigo muito automatizar esse teste pelo fato de que eu ainda estou construindo a aplicação, mas, em questão de ferramentas para teste automatizado, eu já ouvi falar do Selenium, mas eu não tenho certeza, se ele é utilizado para o desenvolvimento Android. Como eu já passei por C, Java, eu não sei direito.

Pergunta 4: Então, a cada versão que a gente lança, existe no fluxo, existe uma etapa que a gente chama de regressivo. O que esse regressivo é, são algumas rodadas de teste que a gente faz de cada equipe do aplicativo. Como onde eu trabalho, tem várias equipes, vários produtos, vários projetos, então cada projeto tem uma equipe responsável, com a ferramenta JIRA. Então tem várias demandas no JIRA, e cada demanda é um tipo de teste específico para fazer daquele produto. Então, por exemplo, eu sou do Seguros. Então, SEGURO VIDAS, aí tem uma demanda de teste regressivo do Seguro Vidas, é simplesmente ver o texto na tela Home do Seguros. Outra etapa é o endereço onde a pessoa mora, quando você digita o CEP, ele busca para mim automaticamente aquele endereço daquele CEP. Então assim, cada round desse regressivo, tem X quantidades de testes para aquele projetinho ali, e aí a gente divide entre a equipe, e cada um fica responsável pelo produto geralmente, é aí, manualmente a gente faz esses testes hoje. Só que a intenção é de automatizar isso aí de alguma forma, porque são testes às vezes que não requer uma habilidade muito de raciocínio, de trabalho mental, mas, físico, de clicar em um botãozinho, ou esperar algum tipo de resposta, então que da para automatizar. Só que como a equipe está sendo formada do zero, então a gente ainda está no inicial, e como ainda a gente ainda precisa lançar versões, a gente ainda está fazendo desta forma, por enquanto. Mas a intenção é de automatizar. Então tem esses rounds, que aí testa tudo. Acabou o round 1, ve o que é que deu problema, corrige. Ai vai para o round 2. Testa tudo, novamente, tudo de novo, todas essas demandas de teste. Acabou, ve se tem algum erro. Se tiver, corrige. Ai, terceiro round, ai testa tudo de novo. Ai, se não tiver erro, vai para produção. Geralmente a gente faz assim.

Pergunta 5: A gente atualmente retesta os mil, só que a gente da uma atenção maior, a gente tenta notificar as pessoas que estão fazendo aquele teste daquela versão nova, o que tem de diferente da anterior, e o que é que pode ocasionar. Por exemplo, se teve alguma alteração naquela tela, o que pode fazer com o que a aquela tela quebre, ou o que de de problema naquela tela. Ou, se vai esperar algum tipo de resposta do *back-end*, o que que a aquela tela deveria fazer ou não. Então assim, a gente retesta os mil casos de teste, só que a gente da uma prioridade maior, para aquelas alterações, uma atenção maior, não vou dizer prioridade, porque é a mesma prioridade, mas, atenção para aquela que teve alteração.

Pergunta 6: Olha falando assim, teste de regressão, até mesmo porque onde eu trabalho, eu utilizo o termo regressivo, eu considero isso como uma técnica de ga-

rantir que o que já tem pronto não sofra com as alterações, vamos dizer assim. Eu, considero esse tipo de regressão, no sentido onde eu trabalho, é testar o que já funciona hoje, para ver se o que foi feito de alteração, até mesmo fora daquele escopo ali, não cause problema. Não sei se ficou claro. **Como você conheceu este assunto?** Foi no trabalho mesmo. **Possui dúvidas sobre o tema teste de regressão?** Assim, eu não sei nem responder se eu tenho alguma dúvida. Eu tenho o conceito na cabeça e tenho um pouco da prática, mas, pode ser que tenham informações dentro do tema, que conseqüentemente eu não sei, mas, eu não saberia nem perguntar, vamos dizer assim, pela falta de conhecimento.

Pergunta 7: Pergunta difícil. Mas, uma coisa fundamental, é questão de parametrização. O que é que eu tenho de passar talvez, sabe, é difícil falar assim. Como eu nunca utilizei, nunca tentei utilizar nenhuma ferramenta disso, é difícil falar o que precisaria ter. O que é que eu vejo, é assim, é o que eu tenho geralmente nos aplicativos que eu preciso, eu tenho um envio de dados para o *back-end* e esse *back-end* me retorna algum tipo de informação. O teste em si deveria saber analisar essa informação que eu estou retornando dos testes e saber exatamente o que eu preciso enviar. Se eu estou enviando certas algumas informações e se eu estou recebendo certas algumas informações, e com base nas informações que eu recebi, o que eu devo fazer. Acho que a palavra parâmetro, seria explicando isso tudo. Parâmetros no recebimento e parâmetros no envio. Porque que eu falo isso, porque às vezes quando eu digito alguma coisa na tela do aplicativo, passa por um longo caminho, um processo, até chegar na ponta da lança que eu digo, que vai ser enviada para o *back-end*, então pode sofrer alguma alteração dos dados, ou nas informações que eu estou enviando. Então eu acho que seria mais ou menos isso. O mais importante destes testes. Porque em questão de layout é bem tranquilo, não seria o problema dos testes não.

Pergunta 8: Depende na verdade do teste, porque as vezes o teste, igual, por exemplo que eu te falei, teste unitário, aí eu faço na linguagem nativa que eu estou utilizando, mas se for um teste de layout, depende da ferramenta que você vai utilizar de teste, mas eu não vejo problema de ser linguagem X ou linguagem Y, não. Acho que é a dificuldade em si é na construção, em si, não na linguagem, sabe. Mas no que precisa ser testado ali, na regra de negócio e tal, não na linguagem. A linguagem eu acho que é o menor fator de problema. Eu acho que nem interfere em nada.

Pergunta 9: Eu acho que tem importância sim, para manter a qualidade. Eu acho que teste, é uma coisa bem pouco difundida e muito pouco valorizada, então acho que se for deixar, se não for sistematizar, eu acho que boa parte abre mão disso, e conseqüentemente acaba perdendo qualidade. Eu acho que é importante ser sistematizado.

Pergunta 10: A com certeza, acho que auxilia sim. É igual ao que eu falei, como muitos vezes, isso, o programador, em si, faz de uma forma que as vezes pra ele tá certo, acho que o teste de regressão é muitas vezes, na verdade, o pior caso é quando eu vou mexer em um código que outra pessoa já mexeu. E quando ela estava, estava tudo OK. E aí eu faço alguma alteração, que pode interferir ali na frente. Então, assim, eu não sei, porque eu não conheço toda regra de negócio do sistema. Então, verificar as funcionalidades que já haviam anteriormente, se estão tudo OK, eu acho muito importante, porque tudo que eu posso mexer aqui, pode tá influenciando em outro lugar do entendeu, sem que eu saiba. Então, acho importante sim, ainda mais que onde eu trabalho, por exemplo, existem múltiplas bibliotecas, ou seja, cada produto é uma biblioteca. Então, se eu mexo aqui, às vezes, está interferindo em outro produto lá, que às vezes eu nem sei, porque tá usando referências minhas. Então, acho que o teste regressivo auxilia e muito na qualidade.

Pergunta 11: Não. Acho que é tranquilo. Até mesmo porque eu vou te falar a verdade, eu tô, eu sou até bem ligado, em quesito academia, a gente tem que correr atrás nessa área de TI, porque não adianta, sempre tem uma coisa nova. Entretanto acho que a academia, ela diverge muito no acompanhamento do mercado. Então, o que que eu vejo? Muitas das vezes, principalmente um, em faculdades menores ou particulares, elas não conseguem acompanhar a dinâmica de mercado. Então, muitas vezes utilizando uma linguagem já que não se utiliza tanto no mercado, entendeu. Assim, ou tem uma, um jeito de trabalhar, que no mercado, cara, é o pessoal bombardeando ali o dia todo pra eu chegar uma coisa. Se eu for levar todas as qualidades que a academia propõe, eu não consigo entregar. Então, assim, eu acho importante envolver ambas as partes andarem juntas porque senão fica muito distorcido uma coisa da outra, sabe. Enquanto um tá andando pra direita, outro tá andando pra esquerda, um tá andando pra frente, outro tá andando pra trás, ou pra esquerda, pra direita, achando pra baixo. Então, assim, acho que tem que andar juntos. Sabe.

15. PARTICIPANTE 15:

Pergunta 1: Eu acho que o primeiro projeto que eu participei com teste Android se não me engano, foi em dois mil e dezoito, acho que foi julho de dois mil e dezoito que eu comecei oficialmente mesmo teste aplicativo no geral, Android e iOS. E aí eu venho trabalhando com isso até dois meses atrás, que aí eu mudei XXXXXX, foram aí esses dois anos e quase três, trabalhando com Android e iOS junto.

Pergunta 2: Junto com a empresa, ela sempre indicava cursos para a gente fazer, cursos gratuitos mesmo, ela dava assim: a, aprende, estuda esse site, bota aqui, agora estuda isso aqui, agora, estuda isso. E foi assim que eu fui aprendendo antes de começar. A empresa, ela preparava a gente, para o que vinha. Então, como

na maioria da parte de desenvolvimento era de aplicativos, então, a gente sempre tinha esse incentivo de aprender tudo do contexto. Então já tinha aprendi, já sabia a teoria, e aí depois que eu comecei realmente a aplicar a teoria no time.

Pergunta 3: Esse meu primeiro projeto eu usei Java, mas foi só nesse. Depois, basicamente todo o resto foi Ruby, com Appium e agora, no final, eu estava automatizando nativa mesmo, com o Espresso, no caso do Android. O desenvolvimento do aplicativo era sempre em Kotlin.

Pergunta 4: Não, era tudo no mesmo time e a gente desenvolvia e testava basicamente em conjunto. Normalmente os devs, já escreviam o código, e a partir do momento que eles faziam um push para o GitHub, se fosse algo que já fosse minimamente testável, nem que seja uma regra, uma tela, um campo de formulário, já começava o teste. Não tinha fluxo, assim: a, isso está pronto, depois testa, o teste era ali feito no meio do desenvolvimento. **Ferramentas para automação das releases?** Sim, para casos de automação sim, mas normalmente é automação com essas ferramentas. Exceto no último projeto que eu já automatizava em Espresso.

Pergunta 5: A gente sempre selecionava alguns, os mais importantes, no caso, a gente chama de MOCK teste. Não dava pra fazer tudo, mas assim, a gente geralmente o que que a gente fazia, às vezes, não sempre. A gente executava este MOCK para uma primeira validação, mas depois a gente deixava até rodando tudo, mas não era, vamos dizer, a obrigatoriedade rodar todos, isso não, o principal era, eram esses MOCKS. *Qual critério para seleção?* Eu sentava junto com o(a) P.O., dependendo, e a gente selecionava o que que é imprescindível não quebrar. Era esses que a gente rodava. O resto, claro, era necessário testar algumas coisas, mas a gente rodava esses que eram imprescindíveis, que se quebrasse, vamos dizer assim, daria merda.

Pergunta 6: Teste regressão é o que a gente faz das funcionalidades já preexistentes. Resumidamente. **Como você conheceu esse assunto?** É eu não sei, foi na empresa do meio em que eu trabalhei, que isso eu acho que são conceitos básicos, de teste. Não me lembro quando que eu fiquei sabendo disso, eu sei que é, vamos dizer assim, fiquei sabendo. **Você possui dúvida sobre esse tema, teste de regressão?** Não.

Pergunta 7: Acho que a rapidez, porque dependendo da quantidade dos testes isso pode demorar horas, ou até dias. Dias é muito, mas demora bastante, e algum mecanismo, alguma maneira de não dar falsos positivos ou falsos negativos. Principalmente os falsos negativos, que a gente chama de FLAKY, e isso rola muito porque, por exemplo, se a gente fizer algum botão num carregou direito, o tester clicou no lugar errado e aí falhou, porque clicou no lugar errado, não, porque a

funcionalidade foi quebrada, se tivesse, eu acho que alguma, acho que seriam essas duas coisas. Rapidez, e termos testes menos FLAKY.

Pergunta 8: Não acho que não tem relação, que é o *framework*, usando é mais fácil você usar uma linguagem que você domine. No caso, como eu falei, na maior parte dos meus projetos foram em Ruby, então, automatizar em Ruby vai ser mais fácil. Mas pegando questão de nativo, usando lá o Kotlin ou React-Native, usar um Detox que é Java Script, ou um Flutter. O Flutter eu não tive experiência ainda, mas é só uma questão de você aprender uma nova linguagem. Não vejo a dificuldade até, no caso, pode até ser um pouco mais fácil, porque as coisas já são feitas para serem integradas. Você não tem uma terceira parte. E já tá tudo ali no pacote da linguagem. Então, eu até achei mais fácil um pouco automatizar em Espresso no caso, usando Kotlin, algumas coisas, do que usando o próprio Ruby, ou Java, ou outra linguagem.

Pergunta 9: Sim, sim, isso é importante sim, até para a gente ter certeza do que o que foi as novas funcionalidades tão funcionando, como o esperado e que não quebraram as outras. É importante sim a gente fazer.

Pergunta 10: Sim, é uma reação direta, mas não é única. Se você não fizer o teste de regressão, você não garante a qualidade, mas só um teste de regressão também, não garante qualidade.

Pergunta 11: Não, achei de boa, achei que fosse mais perguntas. Achei que iriam ter questões mais filosóficas.

16. PARTICIPANTE 16:

Pergunta 1: Antes, de eu entrar aqui na FRITPRAIS, eu só desenvolvia projetos pessoais e terceiros, então, eu praticamente não fazia teste. Então, quando eu entrei aqui, vai fazer dois anos, mais ou menos, que eu comecei, realmente, a aplicar teste. Quando eu entrei aqui o pessoal já tinha uma estrutura madura, sabe, de teste, e uma cultura madura sabe, de teste, então, foi a partir desse momento que eu comecei a, realmente, aplicar testes no desenvolvimento dos meus aplicativos.

Pergunta 2: É, eu uso muito o Android Developer, o site oficial. Tem a comunidade também lá, que o pessoal, pessoal, já teve alguns eventos, falando sobre testes, então, eu também acompanho, tem aquela comunidade Android Dev. Então, eu costumo ser ativo lá na comunidade e artigos, mídias e, é assim que eu me atualizo, sabe? E, troca de experiência com colegas de trabalho. E também comunidade Android DEV BR, mídia, gosto muito de eventos também. Antigamente estava participando de eventos presenciais, mas agora mais eventos virtuais.

Pergunta 3: Aqui a gente usa, nós estamos com acho que noventa e nove por cento Kotlin, já, eu praticamente comecei já com Kotlin. É, eu não comecei no Java. Eu até cheguei a fazer um pouco de Java, antes mesmo de migrar de área. Porque eu migrei de área no desenvolvimento, eu não desenvolvia para Android, eu tenho doze anos de experiência, mas eu não desenvolvia sempre pra Android. Então, quando eu migrei, quando eu realmente mudei minha função, assim, virei desenvolvedor Android, eu já comecei em Kotlin. Utilizam alguma linguagem híbrida? Aqui na empresa não. Eu fiz um projeto pessoal e um, fiz um freelancer em Flutter, só para estudar, pra aplicar, entendeu. Tenho conhecimento de Flutter, acho bem interessante a ideia também. **Utilizam alguma ferramenta para automatizar os testes?** A gente usa os testes unitários do Android mesmo. A gente roda no Android Studio ou via linha de comando. A gente também roda no CI. Nosso CI/CD a gente roda os testes unitários lá. Então, quando você abre um *Pull Request* se não passar os testes unitários, aí você já quebra seu PL lá, então, a gente roda no CI/CD, também, e a gente, a gente trabalhava com testes instrumentados também. Quando, quando a gente não tinha uma equipe de QAs, agora a gente já está com uma equipe QAs bem grande. E a gente começou a migrar, esses testes instrumentados, que a gente fazia com o Espresso no Android. Agora, a equipe de QA aqui, eles estão trabalhando com Appium. Eu acho que a linguagem que eles estão, não sei se é Ruby ou se é, eu não lembro qual é a linguagem que eles estão usando lá para automatizar os testes com Appium. Então, já está na STACK dos QAs. Acho que a gente já tá com quarenta por cento de cobertura, desses testes, end-to-end, que é bater no *endpoint*, abrindo a aplicação e tudo mais. Antes a gente fazia tudo mocado. A gente ainda tem, a gente não tá só assim, a gente não tá rodando tanto eles mais, sabe? A gente meio que transferiu a responsabilidade pros QAs e vamos focar mais nos testes unitários.

Pergunta 4: Não, a gente não tem nenhum procedimento fechado não, fica a cargo da *SQUAD* ou do *CHAPTER*. No caso aqui do nosso *CHAPTER* Android, a gente tem só um acordo do time? Como nosso time aqui somos cinco devs Android. Então nosso time é pequeno, é mais um acordo. A gente já sabe mais ou menos como que a gente vai fazer. A gente pega, primeiro relaciona tudo o que vai entrar naquela release e tipo gera uma anotação mesmo, simples, sabe, nada muito complexo não, do que é que está entrando, a gente pega os cards lá no JIRA, e o que é que vai entrar nessa release. A gente anota, compartilha com todo mundo, tal e fecha a release. A gente manda pra teste interno na loja, na Google Play. E aí a partir desse momento a gente, o nosso, antigamente era a gente mesmo, nós mesmos devs ou o PMO que fazia. Mas como foi crescendo muito a empresa e aí veio a equipe de QA, aí a gente passa pra eles testarem. Os QAs testam e quem quiser também pode testar. A gente comunica para as *SQUADS* que estão, estão envolvidas no aplicativo Android. E e eles dão um OK pra gente e aí a gente, a gente começa a subir para a produção, mas a gente vai sempre subindo parcial, a gente nunca sobe, tipo, a gente sobe para dez por cento, aí vai acompanhando o CRASHDAY,

e aí a gente vai subindo aos poucos, entendeu. Se a gente ver que tem acontecendo alguma coisa, a gente já abre o FIX, para o lançamento, e já sobe o FIX. **Existem ferramentas específicas para automação das novas releases?** É, nós usamos aqui o BitWise? Então, todos nossos BITS estão rodando lá no BitWise, e aí, por exemplo, eu, eu vou fechar, vou, vou ter uma feature normal, e a gente vai testar. A gente usa também o WebCenter, então, a gente, com o BitWise, ele envia pro WebCenter pra gente, então, fica lá o stories de vários APKs lá, que o pessoal testar. E na release a gente também manda o build, toda vez quando você, eu abro um *Pull Request* para a master, eu criei uma *branch* release, e abro um *Pull Request* para master, ele automaticamente roda o build lá no meu BitWise que manda pra Google Play, entendeu? Então, pra gente é bem simples, não tem nenhum processo assim, muito manual, é só basicamente abrir um, criar uma *branch* release e fazer um, abrir um ML para master. Automaticamente, ele é tringado lá no BitWise e já manda para a loja pra gente. O único processo que a gente faz manual é, é sair de teste interno para produção, a gente ainda faz manual, a gente tomou decisão aqui, como o time ainda é pequeno, a gente não automatizou essa parte, é a única parte assim do processo que não está automatizado. Aí lá na Google Play também tem aqueles testes de robô. Da própria Google Play. Então, a gente tem um usuário e senha da Google Play, dos nossos aplicativos e eles fazem tipo um, eles saem navegando, tipo um com robô automatizado, que sai navegando e clicando nas coisas. Aí a gente, isso roda lá por parte do Google Play também.

Pergunta 5: Pelo que está sendo planejado aqui pra ser feito, que é nessa cobertura que os QAs vão fazer agora. Aí eles até passaram um *overview* para gente, mas é assim, vai ficar uma escolha. Eu acho que não vai rodar todos os casos de teste, a gente vai rodar nas telas principais do aplicativo, sabe? Porque leva bastante tempo pra rodar todos os casos de teste, não sei se vai ser muito viável, para toda a release a gente rodar. Então, eles, pelo que eles mostraram lá, eles conseguem configurar quais casos de teste eles vão rodar naquela release. O critério de seleção seria pela os cenários mais complexos ou focando na parte do código que foi alterada? Não, não tenho certeza, não tenho certeza. Qual foi o critério que eles usaram, eu acho que é em relação as principais features, entendeu, que a gente tem. É o core assim, as features core. Isso, a gente tinha um aplicativo de frete. E a gente é um *marketplace* de frete. Então, a parte de frete é tipo, ela não pode quebrar de jeito nenhum. E a nossa feature mais utilizada, então, onde é o principal ponto, sabe. Então, a gente fica a definir, alguns, nem todos os casos venham ser rodados pra fechar, pra testar no início. Eu não sei aí, nossa cobertura ainda está baixo baixa assim em relação a esses novos testes. Está quarenta por cento do que a gente previa, e é capaz que ainda tá rodando rápido, sabe? Mas, antes quando a gente usava o Expresso, a gente rodava todos. Aí a gente rodava todos os testes, mas a gente não tinha uma cobertura muito grande, sabe, de teste, com Expresso.

Pergunta 6: Teste de regressão eu entendo, como eu vou explicar. É ver, ver se o

que funcionava, continua funcionando, por exemplo, se o meu estado anterior, o que eu tinha, o que eu validava anteriormente, continua funcionando, sabe. Se não teve nenhuma quebra neste caminho, durante o desenvolvimento das features. **Como você conheceu esse assunto, teste de regressão?** Mais no trabalho, na troca de experiência de um colega com outro, tal. Quando eu entrei aqui realmente eu tinha um pouco, muito pouca experiência com o testes. Infelizmente a gente acaba realmente os testes ficando em segundo plano quando você está nos seus projetos pessoais ou em um freelancer, e acaba ficando em segundo plano. **Você possui alguma dúvida sobre esse tema, teste de regressão, especificamente?** Não, só se o conceito realmente que eu tenho, se está certo.

Pergunta 7: Eu acho assim, pensando bem, bem pra frente, sabe, é a parte de testes planta, hoje já tem bastante, o Firebase, a XXXLOJA tem, mas se a gente não precisasse escrever os testes, se os testes XXXX conseguissem trazer os cenários para gente era uma coisa bem assim, interessante. Mas acho que esse é um ponto mais pra frente. Pensando mais no atual, se eu pensasse, se eu pensar só no que a gente fazia antes no Expresso, é bem complexo, sabe. Não é, é bem demorado, demora você escrever os testes, você tem toda a parte de mocar os serviços, de preparar seu ambiente, identificar todos os IDs e tudo mais. Eu vi assim, quando eu vi o vídeo que os QAs, não posso falar muito em relação a isso, mas pelo que eles mostraram pra gente, o projeto em si, a configuração do projeto demorou um pouquinho, mas depois que eles estruturaram todo o projeto, as coisas ficaram mais fácil pra eles, sabe. Variáveis de ambiente, então, essa parte de configuração, é o que eu acho que demorou mais para eles, mas pelo que eu vi, as ferramentas estão bem evoluídas, sabe? As ferramentas atualmente, tão bem evoluídas.

Pergunta 8: Eu acho que não, pela minha experiência até em Flutter, e como lá em Flutter, no Android a gente trabalha com XML, ainda. Não, agora chegando o composer vai mudar um pouco a história. Mas no Flutter, como a gente escreve a interface via código, então você consegue via teste unitário, testar sua interface. Então, não vi dificuldade nenhuma de se testar no Flutter, e como no Flutter você tem também um STACKs ali que meio que faz o meio do caminho do dispositivo com o Flutter então, meio que você está blindado do tipo do dispositivo. Quando você vai no Android pode quebrar em N positivos. Tipo, você muda o dispositivo, pode quebrar. Então eu acho que essas ferramentas híbridas e essa cultura de teste elas já vieram prontas para teste, entendeu? É, eu não tenho *Know-How* para falar tipo de um React-Native, porque eu não tenho *Know-How*, mas para o Flutter, eu sei que é bem mais simples de você testar a sua interface Flutter do que no Kotlin, entendeu? É, agora quando você parte para um Appium, que ela trabalha mais em uma camada mais acima da aplicação, então, aí acho que fica até mais fácil, às vezes, para você testar até várias plataformas, independente se é Android, se é iOS, entendeu?

Pergunta 9: Se pudesse estar tudo automatizado, ia ser lindo. Para mim, eu queria apertar um botãozinho ali e a release já está disponível. Rodar uns testes ali, já tem a resposta na hora. Tem uma dificuldade bem grande, assim, para você integrar tantas ferramentas que você precisa, tipo lá está usando, os QAs estão usando BROWSERSTACK, para rodar, rodar os testes, rodar todo o pipeline de testes. Eles podem rodar para local também, mas aí você tem que subir um, sobe um emulador, ou sobe um *device* físico, e isso consome muito recurso da máquina, e aí você joga isso pra nuvem, é maravilha? Então, você tem bastante ferramentas. É, o que é que, às vezes, falta um pouquinho, é que você tem que trabalhar o lado XXXXX seu? Você tem que trabalhar, conhecer um pouco de CI/CD, conhecer as configurações, aí, às vezes, tem uma linguagem diferente, por exemplo, lá no BitWise você tem que usar YALM. Então, você já, às vezes, você tem uma quebra, você tem que trabalhar esse lado seu XXXX, pra você configurar um CI/CD, fazer as conexões com outros serviços, às vezes estou no Bitwise e precisa conectar no AppCenter, eu preciso conectar no BrowserStack. Mas quando você deixa tudo prontinho ali, acho que é bom demais? Você aperta um botão, roda os testes e a confiança que você tem, que você não vai quebrar nada em produção é outra.

Pergunta 10: Sim, ajuda. Eu já tive caso de testes pegando *bugs*. Então, eu sei disso, pode ser um caso que for. Ele já me salvou. Então, eu peguei isso e a gente assim, tá pegando muito assim, os QAs eles tão pegando muito, rodando os testes, eles já pegam alguns problemas, entendeu. Então também tem, tem identificado erros durante o desenvolvimento, ainda mais agora que a gente tá crescendo muito *SQUAD*, muita feature, rápido. Então, está cada vez mais, mais importante. O que a gente tem que tomar cuidado é de não criar cenários falsos positivos. Você testa uma coisa que nunca vai dar erro. Então, essa é uma preocupação que a gente tem que ter. Que se não adianta ser só cobertura e os testes não, não acharem os *bugs* mesmo.

Pergunta 11: Eu achei super legal, depois eu vou sair de ver os resultados, da pesquisa, depois se puder divulgar. Eu estou te acompanhando nas redes sociais também? Provavelmente você deve divulgar. Mas eu achei bem legal e depois você divulga lá na comunidade Android Dev BR também que é bem legal, a comunidade tem bastante desenvolvedor Android e parabéns pelo trabalho e espero que colha frutos aí.