Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Como os níveis de testes automatizados são aplicados no front-end na prática

Autor: Youssef Muhamad Yacoub Falaneh
Orientador: Professora Doutora Carla Silva Rocha Aguiar

Brasília, DF
2023

Youssef Muhamad Yacoub Falaneh

Como os níveis de testes automatizados são aplicados no front-end na prática

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Professora Doutora Carla Silva Rocha Aguiar

Brasília, DF

2023

Youssef Muhamad Yacoub Falaneh
Como os níveis de testes automatizados são aplicados no front-end na prática/ Youssef Muhamad Yacoub Falaneh. – Brasília, DF, 2023-
38 p. : il. (algumas color.) ; 30 cm.

Orientador: Professora Doutora Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. testes-automatizados. 2. front-end. I. Professora Doutora Carla Silva Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Como os níveis de testes automatizados são aplicados no front-end na prática

CDU 02:141:005.6

Resumo

Este texto destaca a importância dos testes automatizados no desenvolvimento de software, especialmente no contexto do front-end. Muitos programadores enfrentam dificuldades em entender e aplicar corretamente os diferentes níveis de testes automatizados nessa área. O objetivo deste trabalho de pesquisa é analisar a aplicação prática dos diferentes tipos de testes automatizados no front-end. O estudo inclui uma revisão bibliográfica, definição dos níveis de testes automatizados e sua relação com o Atomic Design, além da criação de uma metodologia de pesquisa e um roteiro para entrevistas com especialistas. Os resultados das entrevistas ajudarão a identificar os obstáculos enfrentados pelos programadores e a compartilhar as melhores práticas utilizadas no mercado, preenchendo assim uma lacuna de conhecimento nessa área.

Palavras-chave: Testes Automatizados. Teste de unidade. Teste de integração. Teste de sistema. Atomic Design. Pirâmide de testes. Teoria Fundamentada Construtivista.

Lista de ilustrações

Lista de tabelas

Introdução

A qualidade de um software é um fator decisivo para garantir a satisfação dos usuários e osucesso de um projeto. Os testes automatizados tem uma grande importância nesse processo, pois ajudam a verificar e validar o comportamento do sistema (CATELANI et al., 2011). No entanto, quando se trata dos diferentes níveis de testes automatizados aplicados ao front-end, muitos programadores enfrentam dificuldades em entender e aplicar corretamente.

Visto que os testes automatizados são realizados pelos programadores durante a etapa de desenvolvimento, pois este artefato é comumente utilizado no mercado como Definição de Pronto de uma tarefa(DOD). Mas é comum a existência de dúvidas entre qual nível de testes automatizados estão sendo aplicados, ainda mais no contexto de front-end, tal prática acaba levando a muitos testes manuais deixando o processo de validação mais lento e repetitivo.

O objetivo deste trabalho de pesquisa é analisar como os diferentes tipos de testes automatizados são aplicados na prática no front-end. Para isso ele foi dividido em uma revisão bibliográfica sobre o assunto, definindo os níveis de testes automatizados e fazendo um paralelo com o *Atomic Design*. Além de definir uma metodologia de pesquisa utilizando da Teoria Fundamentada Construtivista (CHARMAZ, 2006) e pesquisa semi-estruturada (LETHBRIDGE; SIM; SINGER, 2005), criando um roteiro para a entrevista.

Através das entrevistas com os especialistas, será possível obter dados valiosos sobre a aplicação dos testes automatizados no front-end, identificando os obstáculos enfrentados pelos programadores e absorvendo as melhores práticas utilizadas no mercado. Dessa forma, esta monografia busca contribuir para preencher a lacuna de conhecimento dos programadores em relação aos diferentes tipos de testes automatizados no front-end.

1 Referencial Bibliográfico

1.1 Testes Automatizados

O teste de software, seja ele manual ou automatizado, é um mecanismo de garantia da qualidade por meio da validação de seus requisitos (CATELANI et al., 2011). Uma estratégia de testes bem sucedida deve sempre acusar modificações em uma funcionalidade e encontrar problemas de regressão (MUSTHAFA et al., 2021).

A automatização do teste é o processo de utilizar software para testar outro sofware, ou seja, executar de forma automatizada diversos testes programados para validação das funcionalidades. Ou seja, dado um conjunto de entradas, quando algo acontecer a saída deve suprir as expectativas (BECK, 2003). Dentre a categorização dos testes automatizados, o mais comum é a separação entre: unidade, integração e sistema (AXELROD, 2018).

Uma boa estrutura de um teste automatizado deve respeitar o *GivenWhenThen* (Martin Fowler, 2013):

- *Given*: Definição de todas as informações necessárias para executar o comportamento que será testado.
- *When*: Executar o comportamento.
- *Then*: Verificar o que aconteceu após a execução, comparando as informações retornadas com a expectativa que foi criada. Caso essa comparação falhar, o teste deverá acusar uma quebra na funcionalidade.

Os benefícios dos testes automatizados vão além de uma rápida validação do funcionamento, eles implicam diretamente em uma melhoria do código, pois uma boa cobertura depende de um código bem estruturado, limpo e de baixo acoplamento (FEATHERS, 2002). Além de viabilizar refatorações, pois uma refatoração eficiente deve ser precedida de testes automatizados (FOWLER, 2018).

1.1.1 Os níveis de testes automatizados e o Atomic Design

Nesta seção será definidos os tipos de testes automatizados e feito um paralelo com o *Atomic Design* para um melhor entendimento da aplicação dos testes no contexto de desenvolvimento *front-end*.

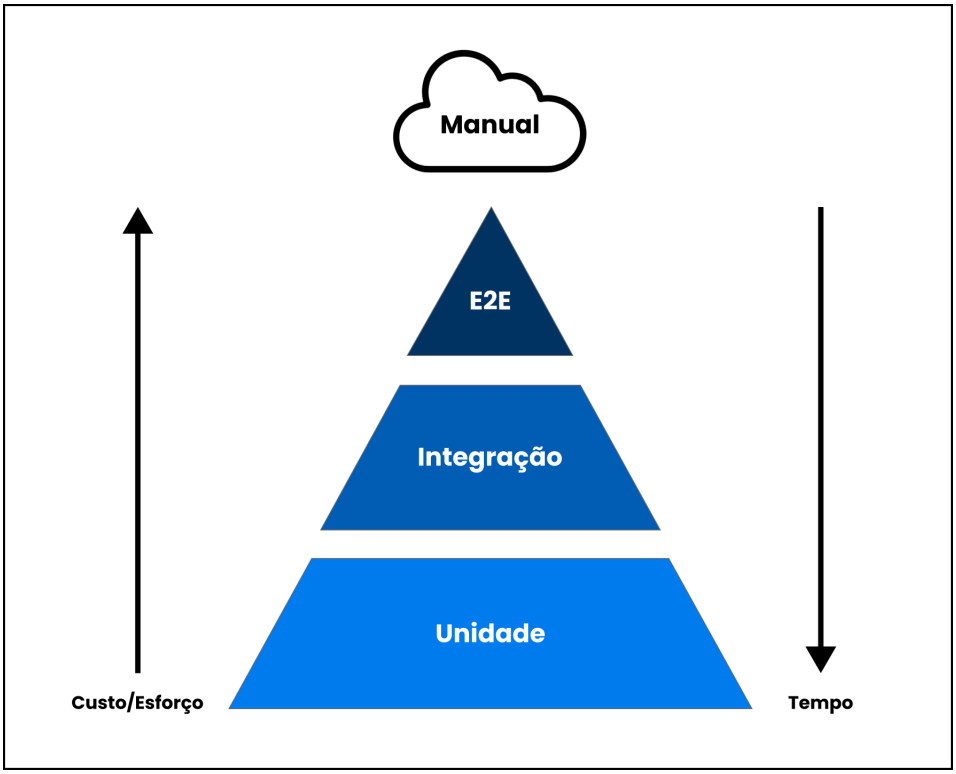

Figura 1 – A pirâmide dos níveis de testes - Fonte: (COHN, 2010)

1.1.1.1 Teste de unidade

Existe uma discussão sobre o quão profundo o teste de unidade deve ser, a recomendação dos autores mais renomados é que o teste deve garantir que todos os caminhos não triviais devem ser testados, incluindo o caminho feliz e os *edge cases*, ao mesmo tempo não devem ser muito atrelados a implementação, pois isso gera um grande estresse ao refatorar o código, pois o qualquer linha modificada já quebraria o teste (FOWLER, 2018).

Portanto, o teste de unidade deve cobrir uma boa quantidade de cenários(entradas, tratamento de exceções, fluxos de controle, etc), testar situações extremas(valores nulos, negativos, etc), testar situações de erros para validar sua reação, focar na regra de negócio em questão e ter uma boa manutenibilidade, pois ele é passível de futuras modificações (BECK, 2003).

1.1.1.2 Teste de integração

Os testes de integração são o próximo passo após a criação dos testes de unidade, pois eles validam como as diferentes partes isoladas de um sistema funcionam em conjunto, fornecendo a confiança necessária para garantir que essa integração não gere erros (BECK, 2003).

O conceito de teste de integração é comumente utilizado de forma errônea em

um escopo mais amplo, criando um grande teste para todas as integrações do sistema, o que acaba parecendo com um teste de sistema. Portanto um bom teste de integração deve testar a integração poucos módulos, pois o foco é validar como essa comunicação vai reagir a diferentes cenários (FOWLER, 2018).

1.1.1.3 Teste de sistema

O teste de sistema, ou comumente conhecido como *end-to-end*, é uma atividade de verificação em que um sistema é testado em por completo em um ambiente igual ao usuário final, abrangendo todos os seus componentes e funcionalidades, com o objetivo de garantir que ele atenda aos requisitos definidos e se comporte corretamente (JORGENSEN, 2018).

Para a criação e execução desse nível de teste é necessário uma infraestrutura mais complexa, por conta da necessidade de ferramentes de automatização da interface de usuário como o *Selenium* ou *Detox*, fazendo com que sua execução seja mais lenta. Portanto, seguindo a pirâmide de testes, eles devem ser a minoria mas são extremamente valiosos pois conseguem simular um fluxo real por completo (FOWLER, 2018).

1.1.1.4 Atomic Design

O Atomic Design é um modelo mental que tem como objetivo definir a hierarquia de uma interface do usuário, facilitando a separação de responsabilidades (FROST, 2016). Utilizar essa metodologia quando lidando com testes automatizados no front-end, facilita identificar qual nível de teste está sendo aplicado. Uma interface de usuário pode ser quebrada nas seguintes etapas:

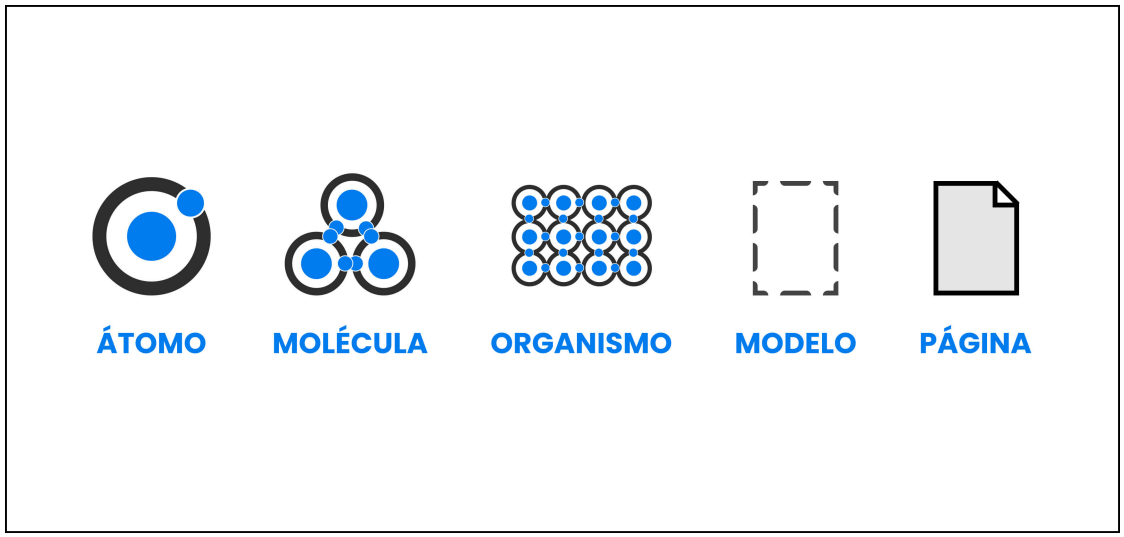

Figura 2 – Componentes do *Atomic Design* - Fonte: (FROST, 2016)

1.1.1.5 Átomo

Os átomos são elementos base de interface do usuário, não podem ser mais quebrados. Um paralelo seriam as tags básicas do HTML, como: *label*, *input* e *button* (FROST, 2016). Visto que um átomo é a menor peça possível, o único teste aplicável é o teste de unidade, mas como estamos lidando com linguagens de marcação como no caso do HTML, os próprios criadores já desenvolveram-os (LE, 2017).

1.1.1.6 Molécula

A molécula é uma composição de átomos, onde juntos funcionam como uma unidade. Um claro exemplo seria um formulário de busca composto de 3 átomos: 1 *label*, 1 *input* e 1 *button* (FROST, 2016). No caso da molécula, é possível testá-la como uma unidade, visto que nessa etapa não existe complexidade de implementação para testar uma integração dos átomos (LE, 2017).

1.1.1.7 Organismo

Os organismos são conjuntos de moléculas e átomos agrupados para formar um componente autossuficiente e reutilizável. Eles representam seções de uma interface, por exemplo um carrinho de compras que é composto por uma listagem de produtos (FROST, 2016).

Por conta do tamanho do componente e pelo fato de possuir uma lógica mais complexa que uma molécula é tentador a aplicação de um teste de integração, mas visto que um organismo é composto de várias moléculas e átomos e âmbos possuem testes de unidade, o que restaria testar seria somente a lógica do organismo em questão, e não a integração com as demais partes. Portanto, o nível de teste recomendado seria o de unidade, validando o funcionamento interno do organismo (LE, 2017).

1.1.1.8 Modelo

Na etapa de Modelo do Atomic Design são criado as estruturas de layout que servirão como base para a construção das páginas antes de ser preenchida com conteúdo real. Ele estabelece a estrutura e a organização dos elementos da interface, fornecendo consistência e coesão visual (FROST, 2016).

Durante a etapa de Modelo, são definidos os *grids*, sistemas de cores, tipografia, hierarquia de informações e outros aspectos visuais que guiarão o design das páginas. O objetivo é criar uma estrutura sólida e consistente que facilite a criação de páginas consistentes e escaláveis. Visto que essa etapa é desenvolvida por um designer o teste mais adequado seria o teste de aceitação com usuários reais do produto, fugindo do escopo desta monografia (FROST, 2016).

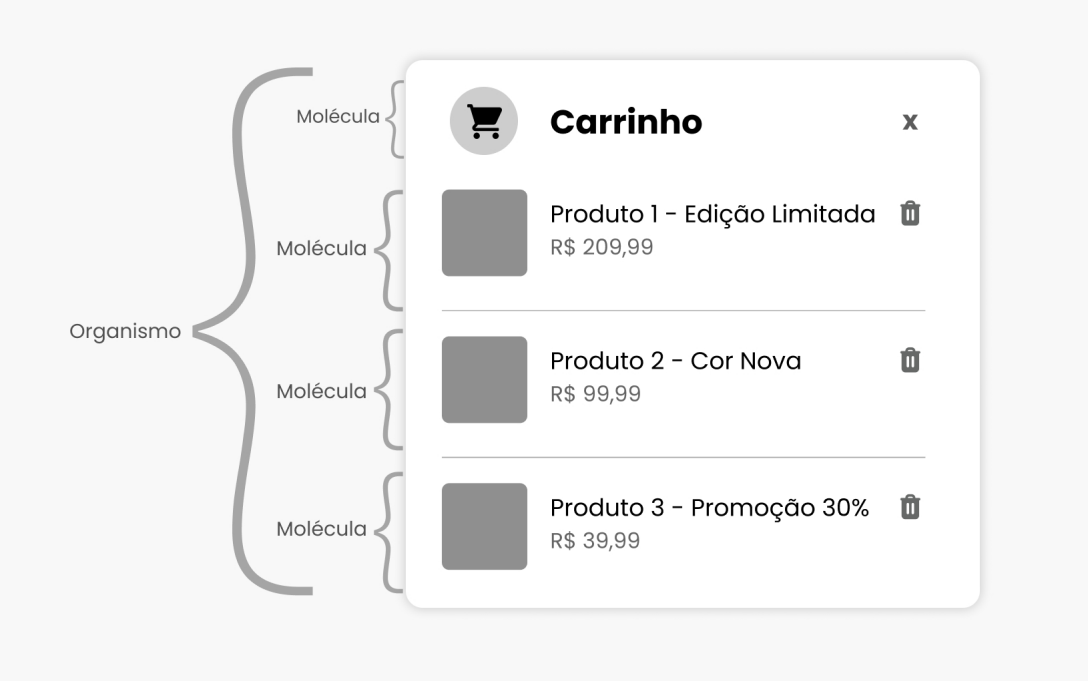

Figura 3 – Organismo e moléculas no *Atomic Design* - Fonte: (FROST, 2016)

1.1.1.9 Página

Por fim, a página é a etapa final do *Atomic Design*, é nela em que todos os componentes modulares e reutilizáveis passados são aglomerados de forma coesa e funcional conforme os requisitos da aplicação (FROST, 2016). Como nessa etapa todas as regras de negócio do sistema são implementadas de forma a interagir com os componentes menores, o teste mais adequado seria o de integração.

Ao aplicar o teste de integração na etapa de Página, é possível validar se os componentes estão sendo renderizados corretamente, simular diferentes cenários de carregamento, erro e sucesso para ver como a integração reage (LE, 2017). Sabendo também que dificilmente um sistema é composto somente de 1 página, a aplicação do teste de sistema também é recomendado, emulando o comportamento de um usuário em um determinado fluxo crítico de páginas (AXELROD, 2018).

1.1.2 Dublês de teste

No contexto dos testes automatizados no front-end, os dublês de teste desempenham um papel importante na criação de um ambiente de teste controlado e isolado. O dublê mais conhecido e muitas das vezes generaliza todos é o *mock*, mas ele é somente 1 dos 5 dublês de testes existentes (MESZAROS, 2007).

- *Dummy*: Os dublês do tipo Dummy são utilizados em testes de unidade para preencher parâmetros que não são relevantes para o teste em questão. Por exemplo, ao

testar um componente de formulário, pode-se utilizar um dummy para preencher campos de dados que não são relevantes para o caso de teste específico.

- *Stub*: Os stubs tem seu uso ao querer simular respostas de chamadas a serviços externos ou APIs. No front-end, isso pode ser utilizado para simular respostas de solicitações HTTP ou para substituir funções assíncronas por versões síncronas, tornando os testes mais rápidos e previsíveis.

- *Spy*: Os spies são úteis para verificar se determinadas funções foram chamadas e com quais argumentos. No front-end, pode-se usar spies para verificar se determinados métodos de um componente foram chamados durante uma interação ou evento específico.

- *Mock*: Os mocks servem para simular o comportamento de componentes ou módulos externos durante os testes. No front-end, eles podem ser usados para substituir componentes dependentes, como APIs de terceiros ou serviços externos, permitindo que os testes sejam executados de forma isolada e consistente.

- *Fake*: Os fakes são implementações alternativas simplificadas de componentes reais. No front-end, eles podem ser usados para substituir recursos complexos, como bancos de dados ou serviços em nuvem, por versões mais simples e controláveis durante os testes, proporcionando uma execução mais rápida e previsível dos testes.

2 Proposta

2.1 Problema

A maioria dos desenvolvedores front-end não sabem diferenciar os níveis de testes em suas aplicações, principalmente em startups menores, juntando todos os níveis de testes em um só o que dificulta o entendimento e futuras manutenções. Visto isso o problema a ser respondido com esta monografia é como os desenvolvedores front-end adotam testes automatizados na pratica?

2.2 Metodologia

Para conduzir o trabalho proposto na questão de pesquisa, é realizada uma pesquisa qualitativa, além de utilizarmos um processo metodológico baseado no trabalho de (LÓPEZ-FERNÁNDEZ et al., 2021).

No contexto atual, serão entrevistados desenvolvedores sêniors dentro de empresas de diferentes tamanhos para obter dados de como os diferentes níveis de testes são aplicados no contexto de front-end(mobile) e colher métricas sobre a distribuição dos níveis de testes automatizados dentro da empresa.

2.2.1 Revisão Bibliográfica

2.2.2 Entrevista Semiestruturada

A entrevista é um método amplamente utilizado na coleta de dados em pesquisas qualitativas (KALLIO et al., 2016), sendo também comum na área da Engenharia de Software (LETHBRIDGE; SIM; SINGER, 2005). A abordagem semi-estruturada oferece flexibilidade ao processo, combinando perguntas específicas com a oportunidade de explorar questões abertas e não planejadas (HOVE; ANDA, 2005).

Existem algumas habilidades que podem tornar o processo de entrevista mais natural, como a postura não julgadora do entrevistador e a formulação concisa das questões. Além disso, o uso de ferramentas de gravação pode auxiliar o entrevistador a se concentrar nas respostas do entrevistado (HOVE; ANDA, 2005). A análise dos dados ocorre a partir da transcrição das entrevistas.

2.2.3 Teoria Fundamentada Construtivista

A Teoria Fundamentada Construtivista (TFC) é um método que envolve a análise de dados não estruturados de entrevistas, por meio da extração de códigos dos dados até que uma teoria seja construída. A TFC consiste em um conjunto de diretrizes organizadas para coletar e analisar dados qualitativos, com o objetivo de construir teorias fundamentadas com base nesses dados (CHARMAZ, 2006).

Dentre as vertentes da Teoria Fundamentada (TF), foi selecionado o trabalho desenvolvido por (CHARMAZ, 2006). De acordo com (EASTERBROOK et al., 2008), o construtivismo afirma que o conhecimento científico não pode ser separado de seu contexto humano, e um fenômeno pode ser totalmente compreendido considerando as perspectivas e o contexto dos participantes envolvidos.

A vertente da TFC difere da versão original da Teoria Fundamentada proposta por (GLASER; STRAUSS, 2010), que afirmam que a teoria levantada por meio dos dados não é influenciada pelo envolvimento do pesquisador, mesmo este tendo uma perspectiva própria e única.

2.2.4 Coleta de Dados

A primeira etapa da Teoria Fundamentada Construtivista (TFC) consiste na obtenção da fonte de dados. São utilizados dados provenientes de entrevistas não estruturadas com o público-alvo composto por organizadores de programas de diversidade. Esses indivíduos são escolhidos devido à sua participação ativa nas comunidades, o que os torna mais acessíveis para a coleta de dados e relevantes para responder às questões de pesquisa.

Na TFC, é fundamental adquirir e reunir os dados de pesquisa por meio de observações e materiais coletados durante as entrevistas, como a transcrição das mesmas. Simultaneamente, realiza-se uma análise inicial dos dados, onde ideias e hipóteses preliminares são propostas. Para facilitar o processo de coleta, as entrevistas serão gravadas usando ferramentas digitais, sempre com a devida autorização do entrevistado. Isso permitirá uma transcrição mais detalhada e uma escuta mais atenciosa durante a análise dos dados.

Como parte do procedimento para obter os resultados desejados, são realizadas comparações constantes e uma codificação inicial das transcrições das entrevistas por meio de uma abordagem de codificação qualitativa. Essa etapa corresponde à segunda fase da TFC.

O *coding* é um anexo de rótulos para os dados, sendo feita uma classificação pelo assunto para depois realizar comparações e buscar padrões em outros dados (CHARMAZ, 2006). É possível coletar mais dados para apoiar ou refutar a teoria atual. À medida que é feita a análise dos dados são criadas também notas analíticas, ideias ou reflexões, chamadas

de memos. Esses registros auxiliam na identificação de categorias e suas propriedades (CHARMAZ, 2006), além de serem comparados com outros dados de outras entrevistas.

Em seguida, os códigos iniciais são agregados e as categorias mais frequentes ou importantes são selecionadas para classificar e relacionar os dados. Para auxiliar nesse processo, pode-se criar um *Coding Book*, que é um dicionário contendo os códigos e suas definições. Com base nos resultados obtidos, uma teoria é desenvolvida.

A partir da combinação e análise desses elementos, é possível identificar as ideias que melhor se adequam ao processo e realizar a interpretação dos dados para sustentar a teoria formulada. Para facilitar a coleta de dados, as entrevistas serão gravadas em ferramentas digitais, desde que haja autorização do entrevistado. Isso permitirá uma transcrição mais detalhada e uma audição mais atenta.

Quando não são identificadas novas categorias ou relações entre elas, atinge-se a saturação teórica, o que significa que os novos dados não geram mais interpretações adicionais para a teoria em questão. Por fim, as saídas dessas fases incluem a codificação, os memos, as categorias e a teoria desenvolvida.

2.2.5 Roteiro de Entrevista

Foi criado um roteiro de entrevista como parte do planejamento para guiar o processo de coleta de dados e assim formar uma teoria fundamentada, considerando que as seguintes questões de pesquisa(RQs) deverão ser respondidas:

- *RQ1*: Quais são as boas práticas para definir os níveis de testes automatizados no front-end?
- *RQ2*: Quais ferramentas são utilizadas para a realização de cada nível de testes automatizados no front-end?
- *RQ3*: Como é feita a avaliação da qualidade dos testes automatizados em cada nível?
- *RQ4*: Qual a distribuição dos níveis de testes automatizados no front-end das empresas?

2.3 Resultados

Tendo em vista o roteiro de entrevista(apêndice A) citado anteriormente, foi necessário achar o público-alvo para respondê-las de forma a obter um maior retorno com base em suas experiências. Para isso 4 desenvolvedores front-end(3 sênior e 1 pleno) foram selecionados para participar da entrevista semi-estruturada.

A busca dos participantes desta pesquisa consistiu em um filtro no LinkedIn das conexões do autor levando em conta anos de experiência(5 ou mais) e nacionalidade brasileira. A única exceção foi a participante Mariana Piccolo, pois ela foi a participante do piloto da entrevista, mas como foi possível obter respostas relevantes a sua entrevista foi utilizada para essa pesquisa.

A coleta de dados consistiu em uma entrevista semi-estruturada onde o autor criou um transcrito com as repostas dos participantes de forma a não perder dados e focar nas questões propostas, além de agilizar o trabalho futuro na aplicação da Teoria Fundamentada Construtivista. O roteiro e os transcritos podem ser encontrados nos apêndices dessa monografia.

Tabela 1 – Relação dos participantes com o transcrito

Entrevistado	**Cargo**	**Experiência(anos)**	**Transcrito**
Mariana Picolo	Dev Front-end Pleno na Swan Bitcoin	4	Apêndice B
Ricardo Soares	Dev React Sênior na Revelo	10	Apêndice C
Christian Hess	Dev Front-end Sênior na Miratech	6	Apêndice D
Guilherme Bruzzi	Dev Front-end Sênior na Coinbase	13	Apêndice E

2.3.1 Codificação

Após a realização de todas as entrevistas semi-estruturadas e seus respectivos transcritos, o autor iniciou o processo de codificação aberta da Teoria Fundamentada

Tabela 2 – Codificação dos transcritos

Entrevistado	**Códigos**
Mariana Picolo	"abordagem em pirâmide", "testes unitários", "testes de integração", "testes de interface do usuário", "cobertura", "velocidade dos testes", "complexidade do sistema", "requisitos de negócios", "experiência do usuário"e "ecossistema de desenvolvimento".
Ricardo Soares	"testes end to end", "testes unitários", "testes de integração", "cobertura", "importância das funcionalidades", "equilíbrio entre quantidade de testes e manutenção".
Christian Hess	"análise abrangente das funcionalidades e requisitos", "abordagem em pirâmide", "testes unitários", "testes de integração", "testes de interface do usuário", "complexidade das funcionalidades", "criticidade dos fluxos de trabalho", "agilidade", "cobertura de testes"e "pressão para lançar rapidamente".
Guilherme Bruzzi	"abordagem replicável", "áreas de maior impacto", "testes unitários", "testes de interface do usuário", "complexidade dos fluxos de trabalho", "funcionalidades críticas", "equilíbrio entre níveis de testes", "ferramentas utilizadas", "qualidade dos testes"e "distribuição dos níveis de testes nas empresas".

Construtivista com o objetivo de analisar os dados de forma detalhada e aberta, identificando conceitos e categorias emergentes.

2.3.2 Categorização

Tabela 3 – Categorização dos códigos

Entrevistado	**Categoria Emergente**
Mariana Picolo	"Abordagem em Pirâmide e Níveis de Testes", "Fatores Considerados na Definição dos Níveis de Testes", "Desafios na Definição dos Níveis de Testes".
Ricardo Soares	"Abordagem em Pirâmide Testes no Front-end", "Fatores Considerados na Definição dos Níveis de Testes", "Ferramentas Utilizadas para Cada Nível de Teste", Avaliação da Qualidade dos Testes Automatizados", "Distribuição dos Níveis de Testes nas Empresas"
Christian Hess	"Abordagem na Definição de Níveis de Testes", "Fatores Considerados na Definição dos Níveis de Testes", "Desafios na Definição dos Níveis de Testes".
Guilherme Bruzzi	"Abordagem na Definição de Níveis de Testes", "Fatores Considerados na Definição dos Níveis de Testes", "Ferramentas Utilizadas para Cada Nível de Teste", "Avaliação da Qualidade dos Testes Automatizados", "Distribuição dos Níveis de Testes nas Empresas".

2.3.3 Proposição Teórica

Após as etapas de codificação aberta e categorização, temos diversas palavras-chaves emergentes para que com elas consigamos criar uma teoria fundamentada em dados. Com isso o autor inicialmente separou as lições aprendidas de cada entrevistado, e por fim agrupou-os para cada RQ proposto no roteiro de entrevista.

2.3.3.1 Lições Aprendidas - Mariana Picolo

Ao definir os níveis de testes automatizados no front-end, os profissionais devem adotar uma abordagem em pirâmide que inclua testes unitários, testes de integração e testes de interface do usuário. Essa escolha deve levar em consideração a complexidade do sistema, os requisitos de negócios, a experiência do usuário e o ecossistema de desenvolvimento. No entanto, enfrentar desafios como equilibrar a cobertura ideal e a velocidade dos testes, bem como manter os testes atualizados com as mudanças na interface, é crucial para garantir a eficácia dos testes automatizados.

2.3.3.2 Lições Aprendidas - Ricardo Soares

Na definição dos níveis de testes automatizados no front-end, é comum adotar uma abordagem que prioriza testes unitários e de integração para garantir a estabilidade

dos componentes individuais e minimizar o esforço de manutenção. A importância das funcionalidades para o negócio e a busca por um equilíbrio entre a quantidade de testes e o tempo de manutenção são fatores essenciais na escolha dos níveis de teste.

Ferramentas como Jest e Cypress são frequentemente utilizadas para testes unitários e de integração, com vantagens e desafios específicos. A avaliação da qualidade dos testes é realizada com base na cobertura de cenários essenciais, capacidade de detecção de problemas e estabilidade ao longo do tempo. A distribuição dos níveis de testes varia entre empresas, com empresas ágeis priorizando testes unitários e de integração, enquanto empresas orientadas para a qualidade investem mais em testes end to end, embora manter o equilíbrio seja um desafio constante.

2.3.3.3 Lições Aprendidas - Christian Hess

Ao definir os níveis de testes automatizados no front-end, é crucial adotar uma abordagem que comece com uma análise abrangente das funcionalidades e requisitos do projeto. A abordagem em pirâmide, priorizando testes unitários na base e avançando para testes de integração e de interface do usuário, é amplamente aceita. A escolha dos níveis de testes é influenciada pela complexidade das funcionalidades, a criticidade dos fluxos de trabalho, a necessidade de agilidade no projeto e o impacto que os testes podem ter na velocidade de desenvolvimento. Enfrentar desafios como o equilíbrio entre a cobertura de testes e o tempo necessário para mantê-los é comum, especialmente quando há pressão para lançar rapidamente.

2.3.3.4 Lições Aprendidas - Guilherme Bruzzi

Na definição dos níveis de testes automatizados no front-end, é importante adotar uma abordagem replicável que comece com uma análise das áreas de maior impacto para o usuário e o negócio. A priorização de testes em áreas críticas é comum, independentemente do nível de teste (unitário, de integração ou de interface do usuário). A complexidade dos fluxos de trabalho e a criticidade das funcionalidades orientam a escolha dos níveis de teste, mas o equilíbrio entre esses níveis é um desafio. Quanto às ferramentas, Jest, Mocha e Detox são comumente utilizados para diferentes níveis de testes no front-end. A avaliação da qualidade dos testes envolve a verificação da eficácia em identificar problemas e a estabilidade ao longo do tempo. Manter uma distribuição equilibrada de níveis de testes nas empresas pode ser desafiador, especialmente quando as prioridades do projeto mudam com frequência.

2.3.3.5 Lições Aprendidas - Compilado Geral

Após a criação das 4 lições aprendidas separadas por participantes, foi realizado uma síntese e um agrupamento dentre os RQs propostos no roteiro para facilitar o entendi-

mento. Essa síntese incorpora as principais informações e práticas relacionadas à definição de níveis de testes automatizados no front-end com base nas respostas dos participantes. Ela destaca a importância da análise, priorização, escolha de ferramentas adequadas, avaliação de qualidade e considerações sobre a distribuição de testes nas empresas.

Abordagem na Definição de Níveis de Testes

Tabela 4 – Abordagem na Definição de Níveis de Testes

Item	Lição Aprendida
1	Começar com uma análise abrangente das funcionalidades e requisitos do projeto.
2	Priorizar testes unitários na base, seguidos por testes de integração e de interface do usuário.
3	Avaliar as áreas de maior impacto para o usuário e para o negócio.
4	A escolha dos níveis de testes é influenciada pela complexidade das funcionalidades, a criticidade dos fluxos de trabalho e a necessidade de agilidade no projeto.
5	Equilibrar a cobertura de testes com o tempo necessário para mantê-los é um desafio comum.
6	Complexidade, criticidade e importância das funcionalidades orientam a escolha dos níveis de teste.

Ferramentas Utilizadas para Cada Nível de Teste

Tabela 5 – Ferramentas Utilizadas para Cada Nível de Teste

Item	Lição Aprendida
7	Utilização de ferramentas como Jest, Mocha e Detox para testes unitários, de integração e de interface do usuário.
8	A seleção das ferramentas é influenciada pela experiência da equipe e pela facilidade de configuração.
9	Vantagens específicas associadas a cada ferramenta, como familiaridade, simplicidade e simulação realista de interações de usuário.

Avaliação da Qualidade dos Testes Automatizados

Tabela 6 – Avaliação da Qualidade dos Testes Automatizados

Item	Lição Aprendida
10	Avaliação da qualidade dos testes envolve revisar regularmente os cenários de teste e a cobertura de código
11	Observação da frequência de falhas em produção relacionadas às áreas cobertas pelos testes.
12	Manter os testes alinhados com funcionalidades críticas e garantir uma boa organização e legibilidade para facilitar futuras manutenções.
13	Incentivar revisões de código dos testes entre os desenvolvedores e a realização de testes exploratórios manuais.

Distribuição dos Níveis de Testes nas Empresas

Tabela 7 – Distribuição dos Níveis de Testes nas Empresas

Item	Lição Aprendida
14	Variação na distribuição dos níveis de testes nas empresas de acordo com a cultura e requisitos do projeto.
15	Empresas que valorizam a agilidade tendem a priorizar testes unitários e de integração.
16	Empresas orientadas para a qualidade podem investir mais em testes de interface do usuário e end to end.
17	Manter uma distribuição equilibrada de níveis de testes é um desafio, especialmente quando as prioridades do projeto mudam frequentemente.

3 Conclusão

Neste trabalho foi realizado uma investigação detalhada sobre as práticas de teste automatizado aplicadas no front-end, com base nas respostas de profissionais experientes no campo. A pesquisa revelou insights valiosos sobre como os testes automatizados são planejados, implementados e avaliados no desenvolvimento de aplicações front-end.

Uma das principais conclusões é a importância de adotar uma abordagem abrangente e equilibrada na definição dos níveis de testes. Os profissionais destacaram a necessidade de começar com uma análise minuciosa das funcionalidades e requisitos do projeto. A abordagem em pirâmide, priorizando testes unitários como base, seguidos por testes de integração e de interface do usuário, é amplamente aceita. No entanto, equilibrar a cobertura de testes com o tempo de manutenção continua sendo um desafio constante.

As ferramentas desempenham um papel fundamental no processo de teste automatizado no front-end. As escolhas de ferramentas são influenciadas pela experiência da equipe e pela necessidade de integração eficiente com o fluxo de desenvolvimento. Jest, Mocha e Detox foram mencionados como ferramentas populares para testes em diferentes níveis, cada uma com suas vantagens específicas.

A avaliação da qualidade dos testes é crucial para garantir a eficácia dos processos de teste automatizado. Isso envolve a revisão regular de cenários de teste, cobertura de código e a observação da frequência de falhas em produção relacionadas aos testes. A manutenção da organização e legibilidade dos testes também é enfatizada.

Por fim, a pesquisa destacou que a distribuição dos níveis de testes nas empresas varia de acordo com a cultura e os requisitos do projeto. Empresas focadas em agilidade tendem a priorizar testes unitários e de integração, enquanto aquelas orientadas para a qualidade podem investir mais em testes de interface do usuário e end to end.

Este estudo oferece uma visão abrangente das práticas de teste automatizado no front-end, fornecendo valiosas lições aprendidas que podem orientar os desenvolvedores e equipes de qualidade na melhoria dos seus processos de teste. À medida que as aplicações front-end continuam a desempenhar um papel crucial no cenário digital atual, a implementação eficaz de testes automatizados é essencial para garantir a qualidade e confiabilidade dessas aplicações em constante evolução. A pesquisa apresentada neste TCC fornece uma base sólida para futuras pesquisas e desenvolvimento de melhores práticas no campo dos testes automatizados no front-end.

Apêndices

APÊNDICE A – Roteiro da Entrevista

Título: Como os diferentes níveis de testes são aplicados no front-end na prática?
Autor: Youssef Muhamad Yacoub Falaneh.
Duração: aproximadamente 40 minutos.

Introdução:

1. Saudações e agradecimentos pela participação na entrevista.
2. Breve explicação sobre o objetivo da pesquisa: investigar as práticas de definição dos níveis de testes automatizados no front-end(unidade, integração e sistema).
3. Coletar dados relevantes sobre o entrevistado: nome, sexo, ensino superior, experiência profissional, cargo atual e tamanho da empresa atual.

RQ1: Boas práticas para definir os níveis de testes automatizados no front-end

1. De acordo com sua experiência, quais são as boas práticas utilizadas para definir os níveis de testes automatizados no front-end?
2. Quais critérios e considerações você leva em conta ao determinar os níveis de testes automatizados para um projeto de front-end específico?
3. Quais são os principais desafios que você enfrenta ao definir os níveis de testes automatizados no front-end?

RQ2: Ferramentas utilizadas para cada nível de teste automatizado no front-end

1. Quais ferramentas você utiliza para a realização de testes automatizados em cada nível no front-end?
2. Como você seleciona as ferramentas adequadas para cada nível de teste automatizado no front-end?
3. Quais são as vantagens e desvantagens das ferramentas que você utiliza para cada nível de teste automatizado no front-end?

RQ3: Avaliação da qualidade dos testes automatizados em cada nível

1. Como é feita a avaliação da qualidade dos testes automatizados em cada nível no front-end?
2. Quais critérios e métricas você utiliza para avaliar a eficácia e confiabilidade dos testes automatizados em cada nível no front-end?
3. Quais práticas são adotadas para melhorar a qualidade dos testes automatizados em cada nível no front-end?

RQ4: Distribuição dos níveis de testes automatizados no front-end nas empresas

1. Como os níveis de testes automatizados são distribuídos no front-end das empresas em que você trabalhou ou conhece?
2. Existem variações na distribuição dos níveis de testes automatizados no front-end em diferentes empresas ou setores? Quais fatores influenciam essa distribuição?
3. Quais são os desafios relacionados em manter uma distribuição dos níveis de testes automatizados no front-end?

Conclusão:

1. Agradecimento final pela participação na entrevista.
2. Solicitação de qualquer consideração adicional que o entrevistado queira compartilhar.
3. Encerramento da entrevista.

APÊNDICE B – Transcrito Mariana Picolo

Nome: Mariana Picolo.
Sexo: Feminino.
Ensino superior: Bacharel em Engenharia de Software - UnB.
Experiência profissional: 4 anos de experiência com desenvolvimento front-end.
Cargo atual: Engenheira de Software Pleno na Swan Bitcoin.
Tamanho da empresa atual: 40 funcionários.
LinkedIn: https://www.linkedin.com/in/marianapicolo/

RQ1: Boas práticas para definir os níveis de testes automatizados no front-end

1. Com base na minha experiência, as boas práticas para definir os níveis de testes automatizados no front-end envolvem seguir uma abordagem em pirâmide. Começamos com testes unitários para garantir que os componentes individuais funcionem corretamente. Em seguida, testes de integração para verificar como esses componentes se encaixam. Finalmente, testes de interface do usuário, para simular interações reais. Isso ajuda a equilibrar a cobertura e a velocidade dos testes.

2. Ao determinar os níveis de testes para um projeto de front-end específico, considero a complexidade do sistema, os requisitos de negócios e a experiência do usuário. Áreas críticas, como fluxos de pagamento, exigem mais testes de interface, enquanto áreas menos críticas podem ser cobertas por testes de integração e unitários. Também levo em conta o ecossistema de desenvolvimento e as tecnologias utilizadas.

3. Um dos principais desafios é encontrar o equilíbrio entre a cobertura ideal e a velocidade dos testes. Testes de interface do usuário podem ser lentos, o que afeta a agilidade. Além disso, manter os testes atualizados à medida que a interface evolui é um desafio constante. Precisamos garantir que nossos testes reflitam com precisão as mudanças no aplicativo.

RQ2: Ferramentas utilizadas para cada nível de teste automatizado no front-end

1. Para testes de unidade no front-end, gosto de usar o Jest, pois ele é bem integrado com as estruturas de front-end e oferece uma configuração simples. Para testes de integração, o React Testing Library é a minha escolha, pois permite simular interações no código e é amplamente utilizado atualmente. E para testes de interface

do usuário, o Selenium é uma ferramenta valiosa devido à sua capacidade de testar em vários navegadores.

2. A escolha das ferramentas depende das necessidades do projeto e da equipe. Procuro ferramentas bem documentadas, com suporte ativo da comunidade e que se integrem bem aos nossos processos de integração contínua.

3. As vantagens variam para cada ferramenta. O Jest oferece um ambiente familiar para testes de unidade em JavaScript, enquanto o Cypress oferece uma experiência visual rica para testes de integração. O Selenium, apesar de sua configuração inicial mais complexa, oferece a vantagem de testar em vários navegadores, garantindo a compatibilidade.

RQ3: Avaliação da qualidade dos testes automatizados em cada nível

1. Avaliar a qualidade dos testes automatizados envolve uma combinação de revisões de código, relatórios de cobertura e execuções regulares dos testes. Garantir que os testes estejam atualizados e refletindo cenários reais é fundamental.

2. Utilizo critérios como cobertura de código, estabilidade dos testes e taxa de falhas para avaliar a eficácia e confiabilidade. Também considero métricas como tempo de execução e frequência de atualizações dos testes.

3. Para melhorar a qualidade dos testes automatizados, priorizo a colaboração entre os desenvolvedores e a equipe de QA. Revisões regulares dos testes, automação da execução de testes e foco em cenários reais de uso contribuem para uma melhoria contínua.

RQ4: Distribuição dos níveis de testes automatizados no front-end nas empresas

1. Em minha experiência, a distribuição dos níveis de testes varia. Geralmente, testes de unidade são a base, seguidos por testes de integração e testes de interface do usuário.

2. Sim, existem variações entre empresas e setores. Empresas em setores regulamentados, como saúde, tendem a dar mais ênfase aos testes de interface do usuário. Startups podem focar mais em testes de unidade para garantir agilidade.

3. Manter a distribuição dos níveis de testes é um desafio. Mudanças nas prioridades do projeto podem levar a desequilíbrios. Educar as partes interessadas sobre os benefícios de manter essa distribuição equilibrada é fundamental para enfrentar esse desafio.

APÊNDICE C – Transcrito Ricardo Soares

Nome: Ricardo Soares
Sexo: Masculino
Ensino superior: Bacharel em Análise de Sistemas de Computadores - Estácio
Experiência profissional: 10 anos de experiência sendo full-stack com ênfase em front-end.
Cargo atual: Desenvolvedor React Sênior na Revelo
Tamanho da empresa atual: 500 funcionários.
LinkedIn: https://www.linkedin.com/in/ricardo-soares-lima/

RQ1: Boas práticas para definir os níveis de testes automatizados no front-end

1. Na minha experiência, embora não seja um grande fã de testes end to end, acredito que seja essencial ter uma boa cobertura em testes unitários e de integração no front-end. Isso ajuda a garantir que os componentes individuais estejam funcionando conforme o esperado e que as interações entre eles não causem problemas.

2. Ao determinar os níveis de testes automatizados para um projeto de front-end, levo em consideração a importância das funcionalidades para o negócio. Áreas críticas merecem mais atenção, enquanto partes menos cruciais podem ter menos testes automatizados.

3. Um dos principais desafios é encontrar um equilíbrio entre a quantidade de testes end to end e o tempo necessário para mantê-los. Eles costumam ser mais lentos e exigem mais recursos de manutenção, o que pode afetar a agilidade do desenvolvimento.

RQ2: Ferramentas utilizadas para cada nível de teste automatizado no front-end

1. Eu uso principalmente ferramentas como Jest para testes unitários e Cypress para testes end to end. Para testes de integração, procuro ferramentas que possam verificar a interação de diferentes componentes.

2. Minha seleção de ferramentas se baseia principalmente na facilidade de uso e na integração com as estruturas que estamos usando. Além disso, procuro ferramentas que possam reduzir o esforço de manutenção.

3. A vantagem do Jest é sua familiaridade com o ecossistema JavaScript e sua configuração relativamente simples. Quanto ao Cypress, ele oferece uma simulação realista

das interações do usuário, mas pode ser mais lento e requer mais cuidados na manutenção.

RQ3: Avaliação da qualidade dos testes automatizados em cada nível

1. A avaliação da qualidade dos testes automatizados envolve principalmente garantir que eles cubram cenários críticos e evitem regressões. Revisões de código e análises regulares de cobertura também são realizadas para garantir a eficácia dos testes.

2. Os critérios que eu utilizo incluem a cobertura de cenários essenciais, a capacidade dos testes de detectar problemas e a estabilidade dos testes ao longo do tempo.

3. Para melhorar a qualidade dos testes, enfatizo a colaboração entre os desenvolvedores e os testadores. Também tento manter os testes o mais simples possível para reduzir a manutenção.

RQ4: Distribuição dos níveis de testes automatizados no front-end nas empresas

1. Nas empresas em que trabalhei, vi uma tendência a investir mais em testes unitários e de integração do que em testes end to end. Isso é especialmente verdadeiro em empresas que priorizam a agilidade no desenvolvimento.

2. Sim, há variações na distribuição dos níveis de testes automatizados. Empresas mais orientadas para a qualidade e regulamentação tendem a investir mais em testes end to end para garantir conformidade.

3. Um desafio é justamente manter um equilíbrio entre os diferentes níveis de testes. Às vezes, há pressões para aumentar a cobertura de testes end to end, mas isso pode afetar a velocidade do desenvolvimento e a manutenção a longo prazo.

APÊNDICE D – Transcrito Christian Hess

Nome: Christian Hess
Sexo: Masculino
Ensino superior: Bacharel Ciência da Computação - Universidade Farias Brito
Experiência profissional: 6 anos de experiência com desenvolvimento web e mobile.
Cargo atual: Engenheiro Front-end Sênior na Miratech
Tamanho da empresa atual: 500 funcionários
LinkedIn: https://www.linkedin.com/in/christian-wolf-a-105365116/

RQ1: Boas práticas para definir os níveis de testes automatizados no front-end

1. Na minha experiência, acredito que definir os níveis de testes automatizados no front-end deve começar com uma análise abrangente das funcionalidades e requisitos do projeto. A abordagem em pirâmide é um princípio sólido, onde priorizo testes unitários na base e, em seguida, testes de integração e testes de interface do usuário.

2. Para determinar os níveis de testes, considero a complexidade das funcionalidades, a criticidade dos fluxos de trabalho e a possível interação entre os componentes. Além disso, avalio a agilidade necessária para o projeto e como os diferentes níveis de testes podem impactar isso.

3. Um dos maiores desafios que enfrento é equilibrar a cobertura de testes e o tempo necessário para mantê-los. Às vezes, a pressão para lançar rapidamente pode levar a cortes nos testes, o que pode afetar a qualidade do produto a longo prazo.

RQ2: Ferramentas utilizadas para cada nível de teste automatizado no front-end

1. Para testes de unidade, costumo utilizar o Jest. Ele é amplamente adotado na comunidade JavaScript e oferece uma configuração amigável para testes unitários no front-end.

2. A escolha das ferramentas depende das necessidades do projeto e da familiaridade da equipe. Procuro ferramentas que tenham boa documentação e uma comunidade ativa para suporte.

3. A vantagem do Jest é sua facilidade de configuração e integração com frameworks de front-end. No entanto, ele pode ser mais focado em testes unitários e pode não ser tão adequado para testes de interface do usuário mais complexos.

RQ3: Avaliação da qualidade dos testes automatizados em cada nível

1. Avaliar a qualidade dos testes automatizados envolve revisar regularmente os cenários de teste e a cobertura de código. Além disso, realizo execuções regulares dos testes para garantir que eles ainda estejam funcionando conforme o esperado.

2. Os critérios de avaliação incluem a abrangência dos cenários de teste, a eficácia dos testes em identificar problemas e a estabilidade dos testes ao longo do tempo. Também considero a facilidade de manutenção dos testes.

3. Para melhorar a qualidade dos testes automatizados, busco manter uma colaboração estreita com a equipe de QA e desenvolvedores. Também realizo revisões regulares de código dos testes para identificar oportunidades de melhorias.

RQ4: Distribuição dos níveis de testes automatizados no front-end nas empresas

1. Nas empresas em que trabalhei, vi uma tendência a enfatizar mais os testes de unidade e de integração. Isso ajuda a garantir a funcionalidade dos componentes individuais e a interação entre eles.

2. Sim, a distribuição dos níveis de testes varia dependendo da cultura da empresa e dos requisitos do projeto. Empresas que valorizam a qualidade podem investir mais em testes end to end, enquanto outras podem focar mais em testes unitários para agilidade.

3. Manter a distribuição equilibrada pode ser um desafio, especialmente quando há pressões para lançar rapidamente. O desafio é convencer a equipe sobre a importância dos testes em diferentes níveis para garantir a qualidade e a estabilidade do produto final.

APÊNDICE E – Transcrito Guilherme Bruzzi

Nome: Guilherme Bruzzi
Sexo: Masculino
Ensino superior: Bacharel Ciência da Computação - UFRJ
Experiência profissional: 13 anos de experiência como full-stack com ênfase em front-end com React JS e React Native.
Cargo atual: Engenheiro Front-end Sênior na Coinbase
Tamanho da empresa atual: 4000 funcionários
LinkedIn: https://www.linkedin.com/in/guilhermebruzzi/

RQ1: Boas práticas para definir os níveis de testes automatizados no front-end

1. Com base na minha experiência, acredito que definir os níveis de testes automatizados no front-end deve começar com uma abordagem replicável. Avalio as áreas de maior impacto para o usuário e para o negócio da empresa e priorizo os testes nesses pontos, sejam eles testes unitários ou de interface do usuário.

2. Ao determinar os níveis de testes automatizados, levo em consideração a complexidade dos fluxos de trabalho, as funcionalidades críticas e o escopo do projeto. Quanto mais crucial a funcionalidade, maior a cobertura de testes, independente do nível.

3. Um dos principais desafios é encontrar um equilíbrio entre os níveis de testes. Às vezes, focar demais em testes de unidade pode negligenciar a visão geral do front-end, enquanto testes end to end podem ser demorados e difíceis de manter.

RQ2: Ferramentas utilizadas para cada nível de teste automatizado no front-end

1. Nos testes de unidade, costumo usar o Jest pela sua ampla utilização no mercado, e as vezes utilizo o Mocha pela sua sintaxe mais explicita principalmente ao lidar com dublês de teste, por exemplo no Jest só temos o *mock* e no Mocha temos o *spy*, *stub* e *mock*.

2. A seleção das ferramentas é influenciada pela experiência da equipe e pela simplicidade de configuração. Escolho ferramentas que não causem sobrecarga e se integrem bem ao fluxo de desenvolvimento.

3. A vantagem do Jest é sua simplicidade e a familiaridade da equipe com a sintaxe, com ele conseguimos escrever teste unitário e de integração. Agora para os testes e2e utilizamos o Detox para o React Native.

RQ3: Avaliação da qualidade dos testes automatizados em cada nível

1. A avaliação da qualidade dos testes automatizados envolve verificar se eles estão alinhados com as funcionalidades críticas e se estão fornecendo uma boa cobertura geral do aplicativo, além de terem uma boa organização e legibilidade para facilitar futuras manutenções.

2. Para avaliar a eficácia dos testes, observo a frequência de falhas em produção relacionadas às áreas cobertas pelos testes. Também verifico se os testes estão acompanhando o ritmo das mudanças no código.

3. Para melhorar a qualidade dos testes automatizados, incentivo a revisão de código dos testes entre os desenvolvedores e a realização de testes exploratórios manuais para identificar lacunas nos testes automatizados.

RQ4: Distribuição dos níveis de testes automatizados no front-end nas empresas

1. Nas empresas em que trabalhei, percebi uma distribuição variada dos níveis de testes. Algumas empresas priorizam testes de unidade, mais que de integração, mas a maioria não possuem teste e2e.

2. Sim, existem variações na distribuição dos níveis de testes, principalmente devido à cultura da empresa e aos requisitos específicos do projeto. Empresas que buscam rapidez podem optar por menos testes de interface do usuário. Atualmente na Coinbase, estamos testando remover os testes e2e pois muitos deles falhavam não pegavam uma regressão, isso acontecia pela dificuldade em ter um back-end estável para testes de ponta a ponta.

3. Manter uma distribuição equilibrada é desafiador, especialmente quando há mudanças frequentes nas prioridades do projeto. O desafio é adaptar a distribuição de acordo com as necessidades em constante evolução do desenvolvimento.

Referências

AXELROD, A. *Complete Guide to Test Automation*. [S.l.]: Springer, 2018. Citado 2 vezes nas páginas 9 e 13.

BECK, K. *Test-driven development: by example*. [S.l.]: Addison-Wesley Professional, 2003. Citado 2 vezes nas páginas 9 e 10.

CATELANI, M. et al. Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use. *Computer Standards & Interfaces*, Elsevier, v. 33, n. 2, p. 152–158, 2011. Citado 2 vezes nas páginas 7 e 9.

CHARMAZ, K. *Constructing grounded theory: A practical guide through qualitative analysis*. [S.l.]: sage, 2006. Citado 3 vezes nas páginas 7, 16 e 17.

COHN, M. *Succeeding with agile: software development using Scrum*. [S.l.]: Pearson Education, 2010. Citado 2 vezes nas páginas 3 e 10.

EASTERBROOK, S. et al. Selecting empirical methods for software engineering research. In: *Guide to advanced empirical software engineering*. [S.l.]: Springer, 2008. p. 285–311. Citado na página 16.

FEATHERS, M. Working effectively with legacy code. *Object Mentor, Inc. Available online at http://www. objectmentor. com*, 2002. Citado na página 9.

FOWLER, M. *Refactoring: improving the design of existing code*. [S.l.]: Addison-Wesley Professional, 2018. Citado 3 vezes nas páginas 9, 10 e 11.

FROST, B. *Atomic design*. [S.l.]: Brad Frost Pittsburgh, 2016. Citado 4 vezes nas páginas 3, 11, 12 e 13.

GLASER, B. G.; STRAUSS, A. L. *Grounded theory: strategien qualitativer forschung*. [S.l.]: Huber, 2010. Citado na página 16.

HOVE, S. E.; ANDA, B. Experiences from conducting semi-structured interviews in empirical software engineering research. In: IEEE. *11th IEEE International Software Metrics Symposium (METRICS'05)*. [S.l.], 2005. p. 10–pp. Citado na página 15.

JORGENSEN, P. C. *Software testing: a craftsman's approach*. [S.l.]: CRC press, 2018. Citado na página 11.

KALLIO, H. et al. Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. *Journal of advanced nursing*, Wiley Online Library, v. 72, n. 12, p. 2954–2965, 2016. Citado na página 15.

LE, N. Creating software component using atomic design and test-driven development. Laurea-ammattikorkeakoulu, 2017. Citado 2 vezes nas páginas 12 e 13.

LETHBRIDGE, T. C.; SIM, S. E.; SINGER, J. Studying software engineers: Data collection techniques for software field studies. *Empirical software engineering*, Springer, v. 10, n. 3, p. 311–341, 2005. Citado 2 vezes nas páginas 7 e 15.

LÓPEZ-FERNÁNDEZ, D. et al. Devops team structures: Characterization and implications. *arXiv preprint arXiv:2101.02361*, 2021. Citado na página 15.

Martin Fowler. *GivenWhenThen*. 2013. [Online; accessed 26-January-2023]. Disponível em: <https://martinfowler.com/bliki/GivenWhenThen.html>. Citado na página 9.

MESZAROS, G. *xUnit test patterns: Refactoring test code*. [S.l.]: Pearson Education, 2007. Citado na página 13.

MUSTHAFA, F. N. et al. A scoping review on automated software testing with special reference to android based mobile application testing. *International Journal on Advances in ICT for Emerging Regions*, v. 14, n. 3, p. 3, 2021. Citado na página 9.