UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

ARTUR DE CASTRO ROCHA SAMAPAIO

CRIAÇÃO DE UM GUIA DE TESTE DE SOFTWARE A PARTIR DA ANÁLISE DE REPOSITÓRIOS DO GITHUB

RUSSAS

2021

ARTUR DE CASTRO ROCHA SAMAPAIO

CRIAÇÃO DE UM GUIA DE TESTE DE SOFTWARE A PARTIR DA ANÁLISE DE REPOSITÓRIOS DO GITHUB

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Prof. Dra. Jacilane de Holanda Rabelo

RUSSAS

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S181c Sampaio, Artur de Castro Rocha.
Criação de um Guia de Teste de Software a Partir da Análise de Repositórios do GitHub / Artur de Castro Rocha Sampaio. – 2021.
32 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2021.
Orientação: Profa. Dra. Jacilane de Holanda Rabelo.

1. Mineração de Repositórios de Software. 2. Teste de Software. 3. GitHub. I. Título.

CDD 005.1

ARTUR DE CASTRO ROCHA SAMAPAIO

CRIAÇÃO DE UM GUIA DE TESTE DE SOFTWARE A PARTIR DA ANÁLISE DE REPOSITÓRIOS DO GITHUB

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em:

BANCA EXAMINADORA

Prof. Dra. Jacilane de Holanda Rabelo (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Ms. José Osvaldo Mesquita Chaves
Universidade Federal do Ceará (UFC)

Prof. Dra. Patrícia Freitas Campos de Vasconcelos
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

Aos meus pais, irmãos, tios, avós, primos por toda a confiança e amor e por terem me ajudado a chegar onde estou hoje.

A minha irmã Isabelle de Castro Rocha Sampaio, por ter me ajudado a tomar sempre as decisões corretas quando elas surgiram.

A minha orientadora Profa. Jacilane de Holanda Rabelo, por ter acreditado e aceitado meu tema de pesquisa, e ter me guiado em momentos de dificuldades.

Aos amigos que a Universidade Federal do Ceará me proporcionou e que sempre estiveram ao meu lado nos altos e baixos, em especial: Cibele Rodrigues, Cleiton Monteiro, Herverson de Sousa, Marlo Henrique, Mateus Franco, Susana Moreira, nossas noites em claro não foram em vão, obrigado por toda a ajuda e motivação, sempre lembrarei de vocês!

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

A todos os professores, servidores e colaboradores da Universidade Federal do Ceará - Campus Russas, por terem proporcionado uma experiência incrível no curso de engenharia de software e me preparar para a vida.

Por fim, a todos que participaram diretamente ou indiretamente da minha formação nesses quatro anos de graduação.

Muito obrigado a todos.

RESUMO

A Mineração de Repositórios de Software(MSR) é uma área de pesquisa que tem como objetivo coletar e analisar dados de repositórios de software. Esses dados podem ajudar na tomada de decisões em projetos de software. Estudos recentes nesta área mostram como usar a MSR para analisar os efeitos de uma linguagem, um processo ou a evolução do software ao longo do ciclo de vida do sistema. O objetivo central deste trabalho foi realizar uma análise em repositórios abertos que estejam hospedados no GitHub para identificar práticas, processos e frameworks que visam auxiliar a aplicação de testes de software. Durante a analise dos repositórios, foram coletadas informações sobre aplicação de boas práticas em códigos de teste de software, além de identificar quais os frameworks mais usados nos projetos. Os resultados obtidos foram usados para a escrita do guia que tem como objetivo ajudar a programadores iniciantes a aprenderem a criar teste eficientes para os seus projetos.

Palavras-chave: Mineração de repositórios de software. Teste de software. GitHub.

ABSTRACT

Software Repository Mining (MSR) is a research area that aims to collect and analyze data from software repositories. This data can help decision making on software projects. Recent studies in this area show how to use MSR to analyze the effects of a language, a process or the evolution of software throughout the system's lifecycle. The main objective of this work was to perform an analysis on open repositories that are hosted on GitHub to identify practices, processes and frameworks that aim to help the application of software tests. During the analysis of the repositories, information was collected on the application of best practices in software testing codes, in addition to identifying which frameworks were most used in the projects. The results obtained were used to write the guide that aims to help beginning programmers to learn how to create efficient tests for their projects.

Keywords: Mining of software repositories. Software Testing. GitHub.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MSR	Mining Software Repository
TDD	Test Driven Development
DSL	Domain-specific language
CI	Continuous Integration

SUMÁRIO

1 INTRODUÇÃO

Repositórios de software são armazenamentos que contêm artefatos de software, como código, metadados (isto é documentação, métricas de código) e informações adicionais relacionadas (isto é histórico de mudanças, rastreador de problemas) (LUZGIN; KHOLOD, 2020). Todos os artefatos de software podem estar disponíveis para mineração de dados. O conhecimento obtido através da análise desses artefatos podem afetar o processo de tomada de decisão em um projeto de software e melhorar a qualidade do sistema de software que está sendo desenvolvido (MUNAIAH *et al.*, 2017).

Tendo em mente a quantidade de informação que pode-se retirar de um repositório de software, foi criado um campo de pesquisa: Mineração de Repositórios de Software, ou Mining Software Repository (MSR). De acordo com FARIAS *et al.* 2016, a MSR se concentra em descobrir informações interessantes e úteis sobre programas e projetos, extraindo e analisando dados disponíveis em diferentes repositórios.

Com a popularização de aplicativos e sistemas de usos gerais, as atividades de garantia de qualidade tornaram-se uma tarefa complexa que requererem conhecimento especializado para alcançar sua implementação. Portanto tarefas como: verificação, validação e teste de software tornam-se cada vez mais comuns e vitais para o sucesso em projetos de software (IBARRA; MUñOZ, 2018). Para garantir a qualidade desses sistemas são adotadas práticas e ferramentas que auxiliam no desenvolvimento de testes, porém um problema que pode ocorrer é saber quais ferramentas, práticas e processos usar no projeto.

Com o intuito de popularizar e ajudar no uso de técnicas, ferramentas e frameworks de teste, este trabalho analisou repositórios hospedados no GitHub com o objetivo de identificar algumas práticas e processos que são adotadas na aplicação de testes de software em repositórios públicos. A partir da análise foi criado um guia que busca auxiliar na adoção de testes de software, voltado para desenvolvedores iniciantes.

Este trabalho está dividido da seguinte forma: O Capítulo 2 apresenta os objetivos gerais e específicos. O Capítulo 3 apresenta a fundamentação teórica necessária. O Capítulo 4 mostra os trabalhos relacionados. O Capítulo 5 mostra os procedimentos metodológicos utilizados. O Capítulo 6 apresenta a sintese dos dados e a forma como o guia foi escrito. No Capítulo 7 será encontrado os resultados da análise dos dados. Por fim, as conclusões da pesquisa e os trabalhos futuros são descritos no Capítulo 8.

2 OBJETIVOS

2.1 Objetivo geral

O objetivo geral deste trabalho consiste em criar um guia que ajude, tanto a usuários novos quanto a usuários experientes, na adoção de práticas de teste de software em seus projetos.

2.2 Objetivo específicos

- Investigar repositórios abertos no github.
- Analisar práticas, processos e frameworks adotados para testes de software.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção é apresentado a fundamentação teórica que foi usada para essa pesquisa.

3.1 Mineração de repositórios de software

Repositórios de software contêm informação histórica e valiosa sobre o desenvolvimento de software. De acordo com SPADINI *et al.* 2018, a MSR foca em extrair e analisar dados disponíveis em repositórios de software, com o intuito de descobrir informações interessantes, vantajosas e contestáveis sobre o sistema.

De acordo com MUNAIAH *et al.* 2017, a análise de repositórios de software produz conhecimentos valiosos sobre a evolução e o crescimento do produto de software. Além disso, eles também afirmaram que o conhecimento ganho através da análise pode afetar as decisões do projeto e aumentar a qualidade do software sendo desenvolvido. Para FARIAS *et al.* 2016, repositórios de software contêm uma grande quantidade de histórico de dados, que pode incluir informações interessantes sobre o código fonte, defeitos e outras questões como novas funcionalidades.

3.2 Teste de software

De acordo com a norma ANSI/IEEE-1059, teste de software é um processo de análise de um item de software para detectar as diferenças entre as condições existentes e necessárias (ou seja, defeitos/erros/bugs) e para avaliar as características do item de software.

Para SPADINI 2021, o teste automatizado é um processo essencial para a qualidade do software. Os testes podem ajudar a assegurar que o código é robusto, ou seja, as funcionalidades do sistema funcionam sob diferentes cenários de uso, e que o código atende às necessidades de performance e segurança.

Para JAMIL *et al.* 2016, o teste de software refere-se a encontrar bugs, erros ou requisitos ausentes no sistema de software desenvolvido. Para os autores, esta é uma investigação que fornece às partes interessadas o conhecimento exato sobre a qualidade do produto.

3.3 Git e GitHub

De acordo com o GIT, ele é um sistema de controle de versão gratuito e open source destinado a lidar com tudo, desde projetos pequenos até projetos grandes, com rapidez e eficiência.

Para KALLIAMVAKOU *et al.* 2014, GitHub é um site de hospedagem de código colaborativo com base no sistema de controle de versão Git. GitHub usa um modelo “fork e pull”, onde desenvolvedores podem criar suas próprias cópias de repositórios e enviar pull requests quando eles querem que o dono do projeto puxe as mudanças para a o ramo (branch) principal do projeto, dessa forma é providenciado um ambiente onde pessoas podem facilmente revisar códigos. Além disso, GitHub também providencia um ambiente onde é possível relatar e discutir erros e bugs do projeto. GitHub também integrou recursos sociais, usuários são capazes de se inscreverem para receber informações por “observar” projetos e “seguir” usuários, resultando em um feed de informação desses projetos e usuários de interesse.

Para COSENTINO *et al.* 2017, Git é um sistema de versionamento de código que gerencia e armazena revisões baseado na replicação ponto a ponto sem mestre, onde qualquer cópia do projeto pode enviar ou receber qualquer informação de ou para qualquer outra réplica. Os autores também discorrem sobre, apesar da relação com o Git, as funcionalidades próprias do GitHub, sendo elas voltadas especialmente para facilitar a colaboração e a interação social em projetos.

4 TRABALHOS RELACIONADOS

Nesta seção serão apresentados os trabalhos relacionados a esta pesquisa.

4.1 Mineração no GitHub

BORLE *et al.* 2018 coletaram repositórios no GitHub com o intuito de analisar os efeitos que a aplicação de Desenvolvimento Orientado a Teste, do inglês Test Driven Development (TDD), tem sobre um projeto. Os autores utilizaram uma linguagem de domínio específico, do inglês Domain-specific language (DSL), e infraestrutura, conhecida como Boa, para coletar os repositórios analisados no trabalho. Eles realizaram a análise com o objetivo de identificar se a aplicação do TDD afeta a velocidade de commits, ou seja a velocidade de registro de alterações, e a quantidade de *bugs* abertos e corrigidos, além de analisar se ele afeta a colaboração no projeto e se integração contínua, do inglês Continuous Integration (CI), é mais prevalente em projetos de TDD. Com base no resultado da análise, os autores concluíram que não existe uma diferença notável entre repositórios que utilizam TDD e aqueles que não utilizam.

KALLIAMVAKOU *et al.* 2014 analisaram os benefícios e os perigos envolvidos na mineração no GitHub. Os autores realizaram uma enquete e entrevistas com usuários do GitHub, com o objetivo de examinar como a plataforma é usada para colaboração, partindo dos dados coletados foi notado que que o GitHub além de ser usado como uma plataforma de colaboração, também é usado para outros fins, como armazenagem de dados, projetos pessoais e projetos escolares. Com isso, os autores fizeram uma análise mais a fundo dos repositórios no GitHub e identificaram alguns perigos relacionados ao uso dessa plataforma para mineração, por exemplo o perigo I: Um repositório não é necessariamente um projeto, além disso eles também recomendaram algumas estratégias para prevenir esses perigos.

Uma análise de dois repositórios abertos foi realizada no trabalho de ZAIDMAN *et al.* 2008 com o objetivo de analisar como testes são feitos em sistemas de software open-source. A partir da análise dos resultados, os autores introduziram três formas de visualização, sendo elas o histórico de mudanças, o histórico de crescimento e a visão da evolução da qualidade do teste, com a união dessas visões foi investigado como os códigos de teste evoluíram ao decorrer do tempo.

4.2 Ferramentas para mineração e análise de dados

A seguir serão apresentados trabalhos que propoem ferramentas e plataformas para auxiliar na mineração de repositórios de software.

GOUSIOS 2013 propôs o GHTorrent, uma ferramenta que coleta dados do GitHub e disponibiliza para pesquisadores, esses dados são disponibilizados através de data dumps, ou seja, uma grande quantidade de dados que pode ser armazenada ou duplicada. Dessa forma, pesquisadores podem ter acesso a uma base de dados de forma mais fácil e rápida.

O *framework* PyDriller foi proposto por SPADINI *et al.* 2018, ele é um *framework* em python que permite extrair informações como commits, pessoas, modificações e códigos fonte a partir do link do repositório no GitHub, não necessitando uma cópia local do repositório para fazer a análise. A vantagem dessa ferramenta é que ela possibilita uma investigação a fundo do repositório, facilitando a análise dos dados.

O Google BigQuery foi utilizado no trabalho de BALTES *et al.* 2019 para armazenar histórico de versão e blocos de códigos em postagens no Stack Overflow. O Google BigQuery permite executar consultas complexas sem a necessidade de importar toda a base de dados, facilitando a coleta e análise de dados.

O Boa foi utilizado no trabalho de HUNG; DYER 2020 para execução de queries, o Boa é uma DSL e infraestrutura que facilita a mineração de repositórios de software. O Boa, assim como o GHTorrent, armazena uma cópia de alguns repositórios abertos do GitHub para facilitar a coleta de dados.

A Tabela 1 contém um resumo dos trabalhos relacionados, sendo possível fazer uma comparação entre todos, incluindo este trabalho.

Tabela 1 – Resumo dos trabalhos relacionados

Trabalho	Uso de Ferramenta	Fonte dos dados	Conteúdo abordado	Objetivo
Dyer e Hung (2020)	Não	GitHub	Mineração de repositórios e Desenvolvimento de ferramenta	Apresentar uma ferramenta que ajuda pesquisadores a minerar repositórios de software.
Borle, et al. (2018)	Sim	Boa	Mineração de repositórios e Desenvolvimento Orientado a Teste	Identificar os efeitos do uso do TDD em projetos de software
Spadini, et al. (2018)	Sim	Git	Desenvolvimento de framework	Apresentar um framework que ajuda pesquisadores a minerar repositórios de software
Kalliamvakou, et al. (2014)	Não	GitHub	Perigos de minerar no GitHub	Disponibilizar recomendações para pesquisadores sobre como melhor utilizar os dados disponíveis e destacar alguns riscos a serem evitados
Gousios (2013)	Não	GitHub	Desenvolvimento de ferramenta	Apresentar uma ferramenta que ajuda pesquisadores a minerar repositórios de software
Zaidman, et al. (2008)	Sim	Não especificado	Teste de software	Identificar a forma como testes ocorrem em sistemas de software open-source
Este trabalho	Sim	GitHub	Mineração de repositórios e Teste de software	Escrever um guia com boas práticas para o teste de software

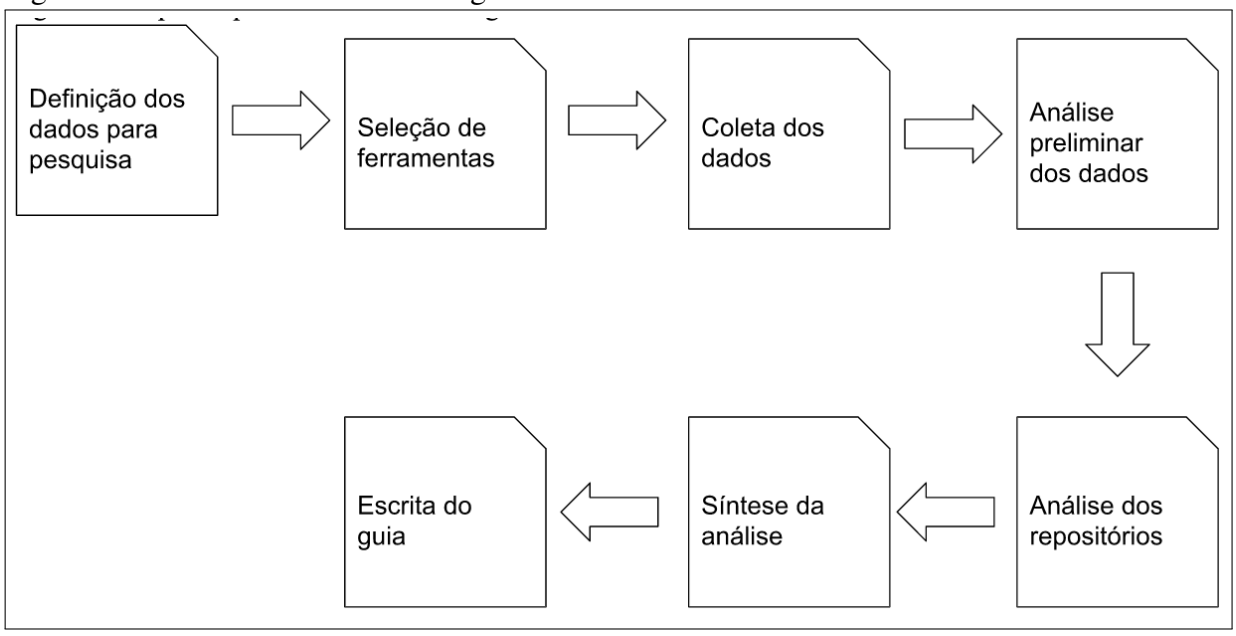
Fonte: Elaborada pelo autor (2021)

5 PROCEDIMENTOS METODOLÓGICOS

Para minerar os repositórios foi feito primeiramente um estudo da literatura para identificar metodologias, práticas e ferramentas para serem usadas durante a coleta e análise deste trabalho. Em seguida foi definido qual o objetivo da mineração, ou seja, quais seriam as informações que se espera obter com a análise dos repositórios. Em conjunto com a definição do objetivo, foi feita a seleção das ferramentas que foram usadas no decorrer do projeto. Uma vez que os objetivos foram definidos e as ferramentas foram escolhidas, foi feita a coleta dos repositórios que irão servir como base de dados para este trabalho, completada a coleta foi feita a separação dos dados, de forma a separar os repositórios que são relevantes à pesquisa dos que não são relevantes.

A seguir é explicado de forma mais detalhada como cada etapa foi executada. A Figura 1 as atividades que foram realizadas ao decorrer desta trabalho.

Figura 1 – Procedimentos Metodológicos



Fonte: elaborado pelo autor (2021).

5.1 Definição dos dados para pesquisa

Esta etapa teve como objetivo definir quais informações serão analisadas no repositório. As informações selecionadas tem o intuito de, ao final da pesquisa, servir como base para a criação de um guia que visa apresentar boas práticas na adoção de teste em sistemas de software e boas práticas de colaboração em projetos com teste de software. Abaixo são apresentadas as

informações que serão analisadas.

- O repositório contém código de teste?
- O repositório utiliza algum framework ou biblioteca para a realização dos testes?
- O projeto utiliza algum processo para testes?
- Como é feita a colaboração nesse repositório?

5.2 Seleção de ferramentas

Esta etapa teve como objetivo selecionar as ferramentas que serão usadas para a realização da pesquisa. Para realizar a coleta foram analisados a linguagem Boa e as ferramentas GHTorrent e Google BigQuery. O Boa, apesar de fornecer outros benefícios além da base de dados, não será usado neste trabalho, pois ele limita a somente para projetos Java disponibiliza o histórico do repositório e o código fonte. Após a eliminação do Boa, teve-se escolher entre utilizar o GHTorrent ou o Google BigQuery, no fim foi escolhido usar o Google BigQuery pois, para utilizar o GHTorrent é necessário acesso ao MySQL, enquanto o Google BigQuery pode-se ser usado pelo navegador Web, dessa forma facilitando a replicação das etapas realizadas no projeto.

5.3 Coleta dos dados

Esta etapa teve como objetivo coletar a base de dados que será usada para a pesquisa. Para coletar esses dados foi feita uma consulta com o Google BigQuery, para a coleta foram considerados repositórios que utilizam a linguagem Javascript. A query usada para a coleta dos repositórios está disposta na Figura 2;

O banco de dados usado para a query foi o “bigquery-public-data.github_repos”, um banco de dados que armazena uma cópia do histórico de repositórios públicos do GitHub, até o dia atual. Porém devido ao tamanho do banco original e a limitação do sistema, onde um usuário só pode consumir 1 Terabyte de dados por mês para queries, foi usado o “sample_contents”, que é uma versão reduzida do banco.

Essa query verifica se dentro do repositório existe um arquivo chamado “package.json”, um arquivo usado na linguagem Javascript para armazenar metadata relevante para o projeto, os repositórios que contêm esse arquivo são coletados e retornados como resposta da query. Após a execução dessa query foi gerado um arquivo Json com 1829 repositórios.

Figura 2 – Query de pesquisa

```
SELECT *
FROM `bigquery-public-data.github_repos.sample_contents`
WHERE id IN (SELECT id FROM (
  SELECT *
  FROM `bigquery-public-data.github_repos.files`
  WHERE path = 'package.json'
  AND ref = 'refs/heads/master'
));
```

Fonte: elaborado pelo autor (2021).

5.4 Análise preliminar dos dados

Esta etapa teve como objetivo fazer uma breve análise dos dados coletados, com o intuito de separar os repositórios que têm relevância para o trabalho daqueles que não são relevantes.

Essa separação foi feita levando em consideração os perigos discutidos por KALLI-AMVAKOU *et al.*, porém como o objetivo dessa pesquisa é analisar a maneira como os testes são feitos e mantidos no GitHub alguns perigos foram desconsiderados, os seguintes perigos foram considerados nesta etapa:

- Perigo I: Um repositório não é necessariamente um projeto.
- Perigo III: A maior parte dos projetos são inativos.
- Perigo V: 2/3 dos projetos são projetos pessoais.
- Perigo IX: Muitos projetos não conduzem todo o seu desenvolvimento de software no GitHub

Levando em conta esses perigos, foi analisado se o repositório é um projeto de software, além disso projetos que são cópias de outros projetos ou que foram criados para teste ou aprendizado foram considerados como não relevantes para a pesquisa, sanando dessa forma os perigos I e V. Considerando o perigo III, foram considerados somente os repositórios que foram atualizados nos últimos 2 anos, desta forma além de diminuir a probabilidade de pegar projetos inativos também é possível garantir que as informações que serão utilizadas no guia estejam o mais atualizadas possível.

Além da separação dos dados, também foi analisado se os repositórios apresentam

algum código de teste e se eles utilizam algum framework de teste.

5.5 Análise dos repositórios

Esta etapa teve como objetivo realizar uma análise mais a fundo dos repositórios, sendo verificado como o código é estruturado, padrões de nomenclatura, como são feitas as correções de bugs e como são introduzidos novos testes.

Para a análise dos códigos de teste foi levado em consideração a forma como os arquivos e os códigos são estruturados, como as funções de testes são feitas e quais seus objetivos, ou seja, quais as responsabilidades dos testes escritos.

Para identificar padrões foi analisado como as funções de teste estão estruturadas, também sendo levado em consideração a definição das funções e de variáveis. A partir dessa análise é possível inferir algumas boas práticas, como por exemplo boas práticas para declaração de funções.

5.6 Síntese da análise

Esta etapa teve como objetivo realizar uma análise de todos os dados coletados dos repositórios, com o propósito de identificar processos, padrões e boas práticas mais usadas em projetos. Para isso foram usados os dados previamente coletados, sendo feita uma análise quantitativa dos padrões encontrados nos vários repositórios. A partir dessa análise foram considerados os padrões mais utilizados, sendo esses padrões considerados como boas práticas para teste de software.

5.7 Escrita do Guia

Esta etapa teve como objetivo utilizar as descobertas encontradas na etapa anterior para a escrita do Guia, que está disposto na plataforma Medium e através do Google Drive.

O Guia será dividido em 5 capítulos. O capítulo 1 introduz o guia, explicando brevimente o conteudo encontrado nele. No capítulo 2 terá uma breve introdução a teste de software. No capítulo 3 estará disposto como fazer arquivos de teste, sendo recomendadas algumas boas práticas encontradas nesta pesquisa para a construção dos arquivos de teste. No capítulo 4 será apresentado um resumo de todas as boas práticas encontradas neste trabalho. O capítulo 5 apresenta a conclusão do guia.

6 SÍNTESE DOS DADOS E ESCRITA DO GUIA

Este capítulo apresenta como ocorreu a sintese dos dados para a escrita do guia.

Para a escrita do guia foi feita uma análise dos repositórios, e com os padrões encontrados, foi feito uma análise quantitativa dos dados encontrados, sendo identificados os seguintes padrões que foram considerados como boas práticas:

- Padrão para declaração de funções.
- Padrão para declaração de variáveis.
- Padrão para a estrutura do código.
- Padrão para nomenclatura de arquivos.
- Responsabilidade do teste

O padrão para declaração de funções coniste na forma como as funções de teste são estruturadas, sendo considerado palavras chaves utilizadas nas declarações das funções, sendo essas palavras usadas para ajudar na definição da responsabilidade do teste. Foi identificado que as palavras chaves mais usadas são "should"e "when", no início da função. Tanto o "should"quanto o "when"são usados em conjunto ao nome do teste para definir o que está sendo testado ou um resultado esperado do teste.

O padrão para declaração de variáveis foi usado para identificar a forma como as váriaveis são declaradas e nomeadas em códigos de teste, para isso foi considerado dois padrões muito utilizados para nomenclatura, sendo eles o camel case, define que as váriaveis devem seguir o seguinte padrão: "nomeDaVariavel", e o snake case, que define o seguinte padrão: "nome_da_variavel". Além da nomenclatura das variáveis, também foi identificado a forma como elas são nomeadas. As variáveis em códigos de teste usam nomes que deixam claro o que essa váriavel representa, no caso de uma resposta de uma requisição ela é geralmente nomeada como "response"ou "data", ou caso seja usada algum atributo especifico da resposta a variável recebbe o nome do atributo.

O padrão para a estrutura do código consistiu na análise da estrtura dos projetos, para isso foi estudado como os repositórios armazenam arquivos de testes. Com esse estudo foi identificado que as estruturas usadas nos repositórios são bem paracidas quando são usados frameworks de teste no projeto, com diferenças em arquivos específicos de framework e alternando o nome do diretório dos arquivos, sendo usado o nome "test"para projetos que utilizam o framework Mocha e o nome "spec"para o framework Jasmine.

O padrão de nomenclatura de arquivos foi usado para identificar padrões nos nomes

dos arquivos de teste. Feita a análise foi identificado que uma boa prática para a nomeação é usar o nome do arquivo, módulo ou componente que os testes estão validando.

A responsabilidade do teste consistiu em analisar o que está sendo testado e como o teste está sendo feito. Foi identificado que os testes são divididos em várias funções, cada uma delas responsável por validar uma funcionalidade da aplicação.

Além dos padrões identificados, também foi analisado quais frameworks de teste são mais utilizados nos projetos, dessa forma foi possível identificar qual framework é mais popular para a aplicação de testes.

6.1 Escrita do Guia

Após a sintese dos dados foi iniciado a escrita do guia, nesta etapa foi definido a estrutura do guia e foi definido um projeto que foi usado para demonstrar e explicar como criar testes.

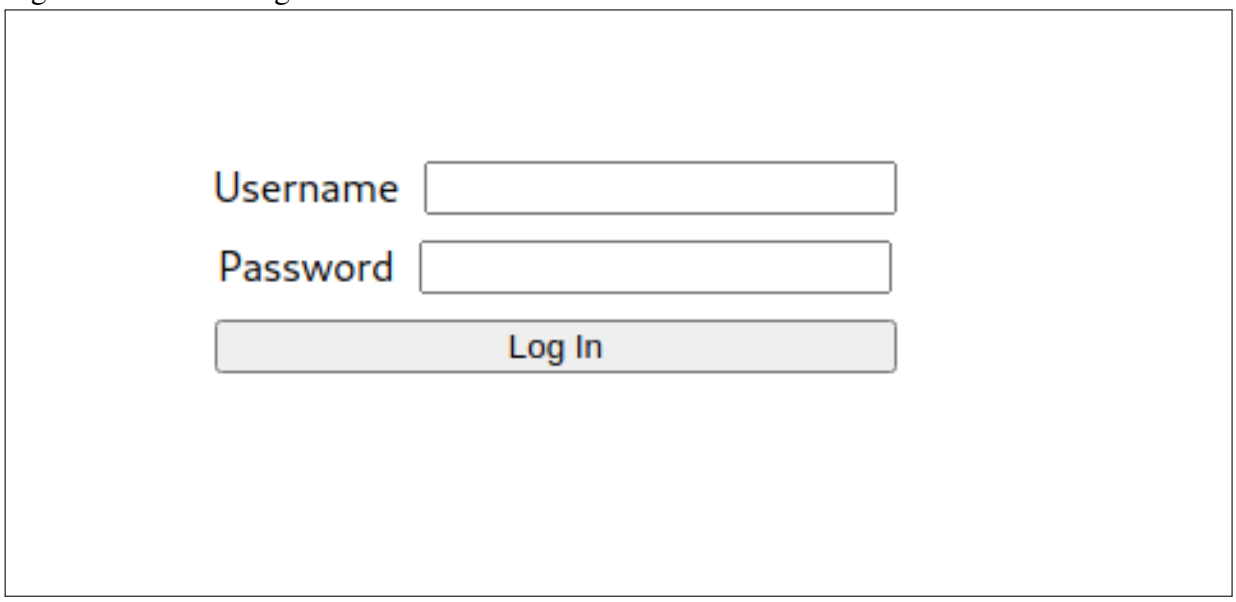
Antes de iniciar a escrita do guia foi feito um estudo de como guias são estruturados e a forma como o conteúdo abordado é apresentado para guias da área de programação. A partir desse estudo foi definido a estrutura do guia e quais são os principais pontos que devem estar presentes nele.

O guia foi estruturado da seguinte forma:

- Introdução, onde foi apresentado o objetivo do guia.
- A importância dos testes de software, onde foi explicado brevemente a importância de aplicar testes em projetos de software.
- Criação dos casos de teste, onde foi apresentado o projeto que será usado no guia e a partir dele foi definido o que será testado.
- Criação do código de teste, onde foi apresentado como criar e codificar testes no projeto, além de explicar como aplicar algumas das boas práticas identificadas neste trabalho.
- Outros projetos para praticar, onde foi apresentado links para outros projetos para o leitor do guiar poder práticar.
- Boas práticas, onde foi decorrido sobre as boas práticas que podem ser aplicadas em projetos, sendo elas identificadas a partir da analise feita neste trabalho.

Sendo definido como o guia deveria ser estruturado foi criado um projeto simples para ser usado como referência no guia para a criação de testes. O projeto usado para a criação dos casos de teste e do código de teste foi uma simples tela de login, como mostrado na Figura 3.

Figura 3 – Tela de login



Fonte: elaborado pelo autor (2021).

A partir do projeto definido, foram criados os seguintes testes:

- Logar com usuário e senha válidos.
- Logar com usuário válido e senha inválida.
- Logar com usuário inválido.

Após a demonstração de como codificar os testes criados, foi recomendado alguns outros projetos para o leitor praticar o conteúdo abordado.

Por fim foram apresentadas as boas práticas identificadas neste trabalho como recomendações para o leitor adotar em seus projetos de software.

7 RESULTADOS

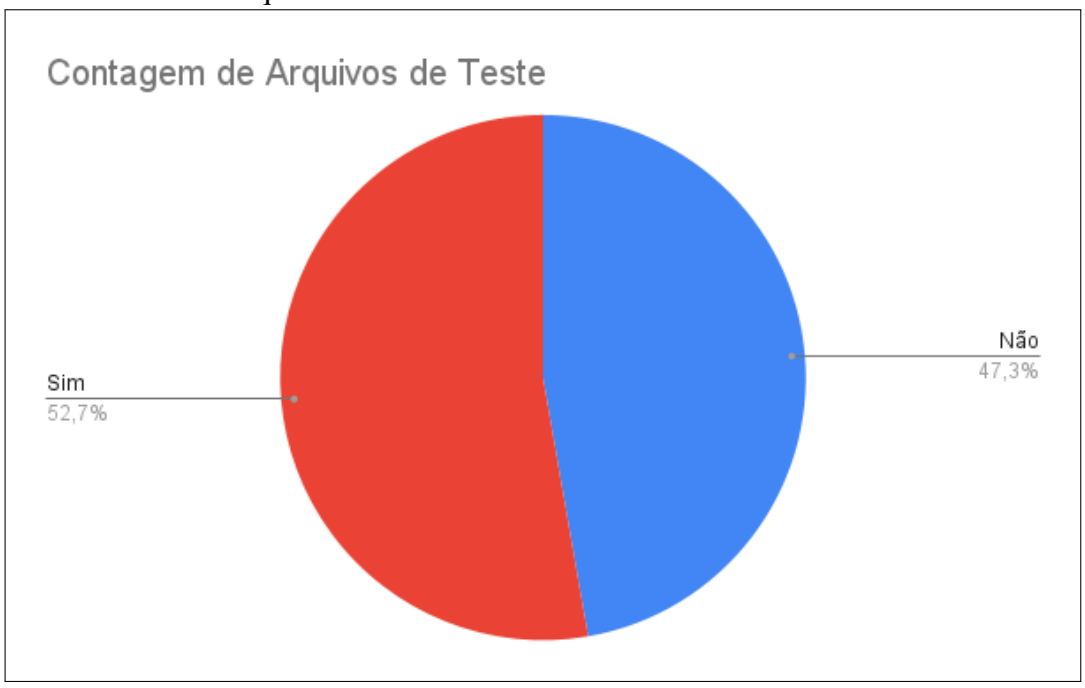
Neste capítulo serão apresentados os resultados obtidos após a análise dos dados.

7.1 Resultado da análise dos repositórios

Nesta subseção será exposto por meio de gráficos, os resultados da análise preliminar dos repositórios. Para a geração dos gráficos foram usados somente os repositórios que ainda estão disponíveis no GitHub, ou seja, dos 1829 repositórios encontrados inicialmente, foi possível encontrar 1513 ainda abertos na plataforma.

Na Figura 4 é possível observar que 52,7% dos repositórios encontrados apresentam algum arquivo de teste, o que equivale a 797 repositórios, enquanto 47,3% não apresentam arquivos de teste ou não seguem o padrão que foi usadado na análise.

Figura 4 – Gráfico de arquivo de teste

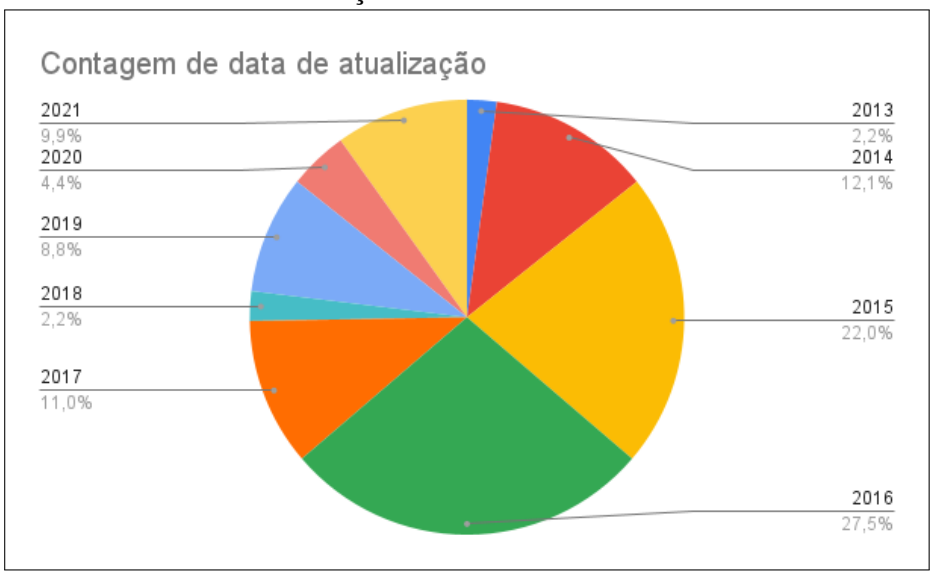


Fonte: elaborado pelo autor (2021).

A Figura 5 mostra a divisão dos 797 repositórios de acordo com o ano em que o repositório recebeu a última atualização. Com base nessa separação foram escolhidos repositórios com atualização entre os anos de 2019 e 2021, sendo eliminados projetos pessoais, restando 183 repositórios.

Desses 183 repositórios, 78 tiveram atualização no ano de 2021, 35 foram atualizados no ano de 2020 e 70 no ano de 2019.

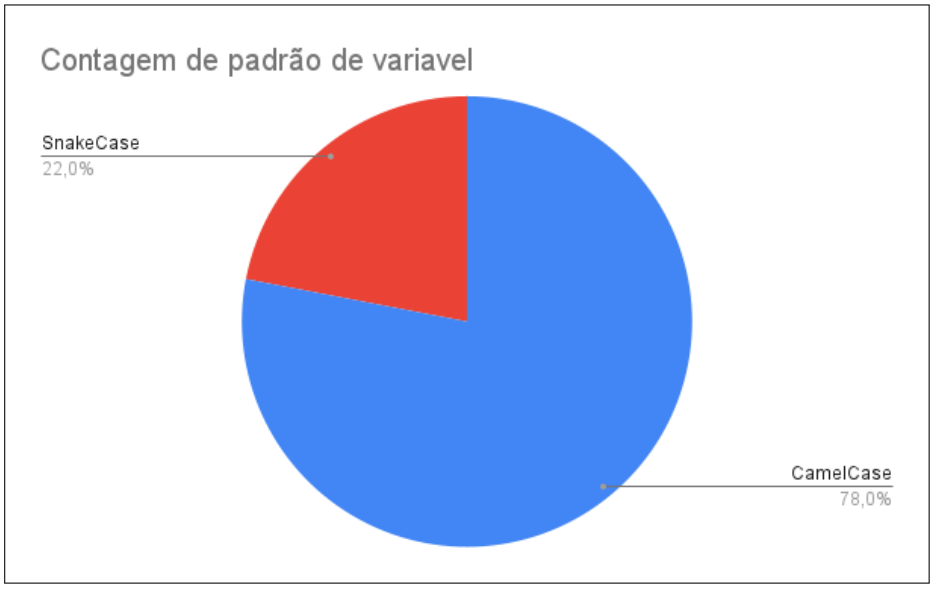
Figura 5 – Gráfico de data de atualização



Fonte: elaborado pelo autor (2021).

A Figura 6 mostra o estilo usado pelos repositórios para a declaração de variáveis e parâmetros, sendo identificados duas formas de nomear, sendo elas o Snake Case e o Camel Case, esses dois são padrões usados nas linguagens de programação para definir a forma como variáveis e funções são nomeadas, o Snake Case segue o seguinte padrão: "nome_da_variavel", enquanto o Camel case segue este padrão: "nomeDaVariavel". Dos 183 repositórios analisados, 143 deles utilizam o Camel Case, enquanto 40 utilizam o Camel Case.

Figura 6 – Gráfico de padrão de formatação das variáveis

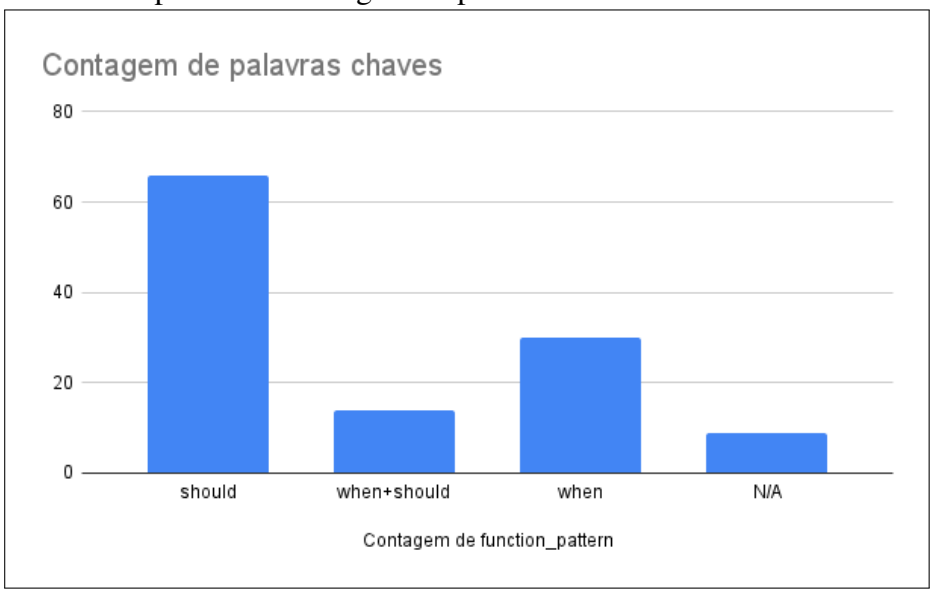


Fonte: elaborado pelo autor (2021).

A Figura 7 mostra as principais palavras encontradas nas declarações de funções. Foi descoberto que as palavras "should"e "when"são as mais usadas no início da declaração de

uma função de teste, um exemplo do uso dessas palavras são as seguintes: "should fetch data"e "when file is deleted". Também foi notado que alguns repositórios utilizam as duas palavras em conjunto, isto é, eles utilizam essas duas palavras no início da declaração da função, enquanto existem outros repositórios que não utilizam nenhum padrão para a nomenclatura, ou seja, um exemplo desse caso é o seguinte: "delete invoice".

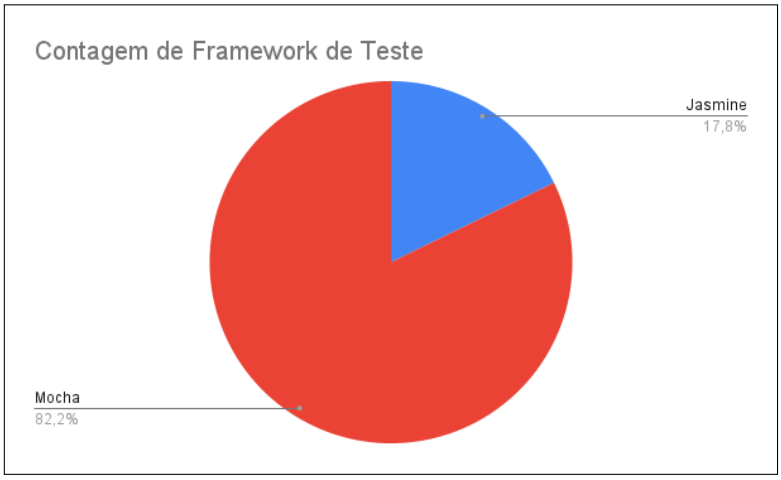
Figura 7 – Gráfico de padrão de contagem de palavras chaves



Fonte: elaborado pelo autor (2021).

A Figura 8 apresenta os *frameworks* encontrados nos repositórios estudados, sendo identificados dois frameworks mais utilizados, o Mocha e o Jasmine. A partir dessa análise é possível concluir que o framework Mocha é o *framework* mais popular para a realização de testes em projetos de software na linguagem Javascript.

Figura 8 – Gráfico de Frameworks de teste



Fonte: elaborado pelo autor (2021).

7.2 Resultado da síntese

Nesta subseção será exposto as informações retiradas da análise dos repositórios, sendo apresentado os padrões e boas práticas encontradas.

Após a análise foi possível notar alguns padrões e boas práticas que são adotados, sendo eles:

- Na maioria dos casos, os códigos de teste ficam em uma pasta com o nome “test” separados do resto do código, neste caso os arquivos têm o mesmo nome que o arquivo o qual eles são responsáveis por testar ou o nome do arquivo seguido da palavra "test", como por exemplo "NomeDoArquivoTest".
- Foram encontrados alguns casos em que os arquivos de teste ficam junto com o resto do código, nesse caso os os arquivos de teste tem uma nomenclatura com o seguinte padrão: NomeDoModuloTest, onde o “NomeDoModulo” é o arquivo onde se encontra o código da funcionalidade.
- Caso o módulo que está sendo testado faça alguma requisição a uma api ou ao banco de dados, é feito um teste para esta requisição.
- As funções nos arquivos de teste tem nomes intuitivos, ou seja, é possível saber o que está sendo testado a partir do nome da função
- Foram encontradas duas palavras que são muito usadas nas funções de teste, sendo elas “should” (deve) e “when” (quando), Ambas são usadas no início do nome da função de teste.
- Os testes usam injeção de dependência, ou seja, os testes não importam os arquivos que eles estão testando, ao invés disso os testes utilizam os parâmetros da função para pegar os dados necessários para a sua execução.
- Devido a injeção de dependência, na maioria dos casos as variaveis nao sao declaradas na função de teste, ao invés são usados os parâmetros passados para o teste, neste caso o nome dos parâmetros é ou mesmo nome que a variável a qual ele está referenciando ou é denominada com nomes genéricos como "value"ou "data".
- Foram encontrados alguns casos onde variáveis são criadas no corpo da função, isso ocorre comumente em funções responsáveis por testar respostas de requisições ao bancos de dados ou API’s, nesse caso a variável é nomeada ou como "response"ou "data", sendo palavras genéricas, ou ela é nomeada levando em consideração o que está sendo requisitado, como por exemplo "amountToPay".

- A nomenclatura de variáveis são no estilo Camel Case, ou seja, elas seguem o seguinte padrão "nomeDaVariavel"
- As funções são responsáveis por testar somente uma funcionalidade, em alguns casos existem várias funções de teste para a mesma funcionalidade, cada uma sendo responsável por testar um cenário diferente.

7.3 Escrita do Guia

O guia criado a partir deste trablho pode ser encontrado no seguinte link https://drive.google.com/file/d/1VZFoFyNRRouRm29d4ONeQH6_Hz14QwXc/view?usp=sharing

8 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho teve como objetivo investigar repositórios públicos hospedados no GitHub. Para isso foi necessário um estudo sobre mineração de repositórios de software, além de uma pesquisa bibliográfica para entender melhor sobre a aplicação de teste de software e o conceito de Git e GitHub.

A coleta dos dados necessários para essa pesquisa foi feita através do Google BigQuery, onde inicialmente foram coletados 1829 repositórios, que após um processo de filtração foram reduzidos para 183 repositórios. Sendo feita essa divisão, foi feito uma análise mais a fundo nesses repositórios para identificar padrões utilizados, sendo coletadas as palavras mais usadas, padrões de declaração de variáveis e funções, a estrutura dos códigos e como eles são organizados dentro do projeto.

Os resultados obtidos desta análise apontam que o framework mais utilizado para o desenvolvimento de testes de software na linguagem Javascript é o Mocha, além disso, também foi descoberto que as variáveis e funções nesses projetos utilizam em sua maioria o padrão Camel Case. Outra descoberta feita foi que os arquivos de teste ficam guardadas em pastas separadas do código do software, esta pasta sendo nomeada "Test", além disso os arquivos de teste tem o mesmo nome que o arquivo que está sendo testado, ou com o nome do arquivo seguido da palavra "test". Também foram coletadas as palavras mais usadas para as declarações de funções, sendo encontradas as palavras "should"e "when".

Com a análise feita, este trabalho dispõe os padrões e boas práticas encontradas em repositórios abertos na plataforma GitHub, a fim de ajudar a novas desenvolvedores a desenvolverem testes em seu projetos, para isso será escrito um guia com tutoriais de como iniciar, recomendações de estrutura e boas práticas, exemplos de códigos e desafios para que esses desenvolvedores possam praticar.

8.1 Trabalhos Futuros

Em trabalhos futuros espera-se a validação do guia com programadores com diferentes níveis de experiência em teste de software, para que dessa forma seja possível identificar as partes que geraram mais dificuldades para desenvolvedores iniciantes e com isso encontrar possíveis melhorias para o guia. Uma outra contribuição seria a análise do ciclo de vida dos testes de software, ou seja, analisar como o teste é criado, atualizado e removido, de forma a

ajudar na aplicação de testes em projetos de software.

REFERÊNCIAS

ANSI/IEEE-1059. Ieee guide for software verification and validation plans. **IEEE Std 1059-1993**, p. 1–87, 1994.

BALTES, S.; TREUDE, C.; DIEHL, S. **SOTorrent: Studying the Origin, Evolution, and Usage of Stack Overflow Code Snippets**. IEEE Press, 2019. 191–194 p. (MSR '19). Disponível em: https://doi.org/10.1109/MSR.2019.00038.

BORLE, N. C.; FEGHHI, M.; STROULIA, E.; GREINER, R.; HINDLE, A. Analyzing the effects of test driven development in github. **Empirical Software Engineering Journal**, v. 23, p. 1931–1958, 2018.

COSENTINO, V.; IZQUIERDO, J. L. C.; CABOT, J. A systematic mapping study of software development with github. **IEEE Access**, v. 5, p. 7173–7192, 2017.

FARIAS, M. A. de F.; NOVAIS, R.; JúNIOR, M. C.; CARVALHO, L. P. da S.; MENDONçA, M.; SPÍNOLA, R. O. A systematic mapping study on mining software repositories. Association for Computing Machinery, New York, NY, USA, p. 1472–1479, 2016. Disponível em: https://doi.org/10.1145/2851613.2851786.

GIT. **git-scm**. 2021. Disponível em: https://git-scm.com/.

GOUSIOS, G. The ghtorent dataset and tool suite. **2013 10th Working Conference on Mining Software Repositories (MSR)**, p. 233–236, 2013.

HUNG, C. S.; DYER, R. Boa views: Easy modularization and sharing of msr analyses. **Proceedings of the 17th International Conference on Mining Software Repositories**, Association for Computing Machinery, New York, NY, USA, p. 147–157, 2020. Disponível em: https://doi.org/10.1145/3379597.3387480.

IBARRA, S.; MUñOZ, M. Support tool for software quality assurance in software development. **2018 7th International Conference On Software Process Improvement (CIMPS)**, p. 13–19, 2018.

JAMIL, M. A.; ARIF, M.; ABUBAKAR, N. S. A.; AHMAD, A. Software testing techniques: A literature review. **2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)**, p. 177–182, 2016.

KALLIAMVAKOU, E.; GOUSIOS, G.; BLINCOE, K.; SINGER, L.; GERMAN, D. M.; DAMIAN, D. The promises and perils of mining github. **Proceedings of the 11th Working Conference on Mining Software Repositories**, Association for Computing Machinery, New York, NY, USA, p. 92–101, 2014. Disponível em: https://doi.org/10.1145/2597073.2597074.

LUZGIN, V. A.; KHOLOD, I. I. Overview of mining software repositories. **2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)**, p. 400–404, 2020.

MUNAIAH, N.; KROH, S.; CABREY, C.; NAGAPPAN, M. Curating github for engineered software projects. **Empirical Software Engineering Journal**, v. 22, p. 3219–3253, 2017.

SPADINI, D. **Supporting Quality In Test Code For Higher Quality Software Systems**. Tese (Doutorado) – Delft University of Technology, The address of the publisher, 3 2021. An optional note.

SPADINI, D.; ANICHE, M.; BACCHELLI, A. Pydriller: Python framework for mining software repositories. **Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering**, Association for Computing Machinery, New York, NY, USA, p. 908–911, 2018. Disponível em: https://doi.org/10.1145/3236024.3264598.

ZAIDMAN, A.; ROMPAEY, B. V.; DEMEYER, S.; DEURSEN, A. van. Mining software repositories to study co-evolution of production amp; test code. **2008 1st International Conference on Software Testing, Verification, and Validation**, p. 220–229, 2008.