UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FRANCICLÁUDIO DANTAS DA SILVA

UM ESTUDO DE CASO DE IMPLANTAÇÃO DE PROCESSO DE QUALIDADE NO DESENVOLVIMENTO DE UMA APLICAÇÃO PARA DISPOSITIVOS ANDROID

CAMPINA GRANDE - PB

2023

FRANCICLÁUDIO DANTAS DA SILVA

UM ESTUDO DE CASO DE IMPLANTAÇÃO DE PROCESSO DE QUALIDADE NO DESENVOLVIMENTO DE UMA APLICAÇÃO PARA DISPOSITIVOS ANDROID

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Professor Dr. Tiago Lima Massoni

CAMPINA GRANDE - PB

2023

FRANCICLÁUDIO DANTAS DA SILVA

UM ESTUDO DE CASO DE IMPLANTAÇÃO DE PROCESSO DE QUALIDADE NO DESENVOLVIMENTO DE UMA APLICAÇÃO PARA DISPOSITIVOS ANDROID

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Tiago Lima Massoni

Orientador – UASC/CEEI/UFCG

Professor Dr. Rohit Gheyi

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 28 de Junho de 2023.

CAMPINA GRANDE - PB

RESUMO

O desenvolvimento de aplicações Android é uma tarefa complexa. A variedade de dispositivos, e suas diferentes configurações, são um exemplo das razões que dificultam esse desenvolvimento. Por causa disso, um processo de desenvolvimento bem planejado é importante para garantir a qualidade do software. A falta desse processo pode aumentar a incidência de bugs e as chances de requisitos do sistema não serem satisfeitos. Esse trabalho tem como objetivo implantar um processo de qualidade de software no contexto do desenvolvimento de uma aplicação Android. Essa aplicação tem como propósito ser utilizada por agentes de saúde que a alimentarão com dados que serão processados e modelados a fim de emitir alertas antecipados sobre a incidência de populações de mosquitos transmissores de doenças. O processo de qualidade será definido a partir de uma pesquisa na literatura sobre boas práticas em desenvolvimento Android, um estudo sobre o domínio do problema e sobre regras de negócio da aplicação. A partir dessa pesquisa, será discutida uma estratégia para a criação de um processo que engloba o desenvolvimento de testes, versionamento de código, escolha dos requisitos não funcionais e procedimentos para a implantação da aplicação. Ao final do trabalho, será aplicado um questionário aos supervisores para que eles possam avaliar a eficácia do processo implantado. Espera-se que, depois de adotado o processo de qualidade, exista uma maior clareza e fluidez nas entregas da aplicação, e que seja minimizada a quantidade de problemas.

A CASE STUDY OF QUALITY PROCESS IMPLEMENTATION IN THE DEVELOPMENT OF AN APPLICATION FOR ANDROID DEVICES

ABSTRACT

The variety of devices and their different configurations make the development of Android applications difficult. Because of this, a well-defined and well-planned quality process is important. The lack of this process can increase the incidence of bugs and the chances of system requirements not being satisfied. This work aims to implement a software quality process in the context of Android application development. This application is intended to be used by health agents who will feed it with data that will be processed and modeled in order to issue early warnings about the incidence of disease-transmitting mosquito populations. The quality process is defined based on a literature search on good practices in Android development, a study on the problem domain and on the application's business rules. Based on this research, a strategy was discussed for the creation of a process that encompasses the development of tests, code versioning and procedures for the application deployment. At the end of the work, a questionnaire was applied to the project supervisors so that they could evaluate the effectiveness of the implemented process. The implantation of the process resulted in the identification of defects in the application, increasing the reliability from the implementation of system tests and helping the maintainability of the system with the documentation of the requirements and the standardization of the development process. Finally, it was noted that the process would be better evaluated if communication between team members was more efficient and if it were applied over a longer period of iterations.

Um Estudo De Caso De Implantação De Processo De Qualidade No Desenvolvimento de Uma Aplicação Para Dispositivos Android

Francicláudio Dantas da Silva
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
franciclaudio.silva@ccc.ufcg.edu.br

Tiago Lima Massoni
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
massoni@computacao.ufcg.edu.br

RESUMO

A variedade de dispositivos, e suas diferentes configurações, dificultam o desenvolvimento de aplicações Android. Por causa disso, um processo de qualidade bem definido e planejado é importante. A falta desse processo pode aumentar a incidência de bugs e as chances de requisitos do sistema não serem satisfeitos. Esse trabalho tem como objetivo implantar um processo de qualidade de software no contexto do desenvolvimento de uma aplicação Android. Essa aplicação tem como propósito ser utilizada por agentes de saúde que a alimentarão com dados que serão processados e modelados a fim de emitir alertas antecipados sobre a incidência de populações de mosquitos transmissores de doenças. O processo de qualidade é definido a partir de uma pesquisa na literatura sobre boas práticas em desenvolvimento Android, um estudo sobre o domínio do problema e sobre regras de negócio da aplicação. A partir dessa pesquisa, foi discutida uma estratégia para a criação de um processo que engloba o desenvolvimento de testes, versionamento de código e procedimentos para a implantação da aplicação. Ao final do trabalho, foi aplicado um questionário aos supervisores do projeto para que eles pudessem avaliar a eficácia do processo implantado. A implantação do processo resultou na identificação de defeitos na aplicação, aumentando a confiabilidade a partir da implementação de testes de sistema e ajudando a manutenibilidade do sistema com a documentação dos requisitos e a padronização do processo de desenvolvimento. Por fim, notou-se que o processo seria melhor avaliado se a comunicação entre a equipe fosse mais eficiente e se ele fosse aplicado em um período maior de iterações.

Palavras-chave

Qualidade de software, processo de software, testes de software, aplicação Android.

1. INTRODUÇÃO

Ao longo dos anos, o Android conseguiu se tornar o Sistema Operacional para dispositivos móveis mais popular do mundo [1]. Os motivos são muitos, mas pode-se destacar a sua compatibilidade com uma grande variedade de dispositivos [2]. Contudo, essa compatibilidade acaba tornando o processo de desenvolvimento de aplicações para esse sistema uma tarefa complexa. Isso porque com a grande quantidade de dispositivos, existem diversas configurações que podem atrapalhar o desenvolvimento da aplicação se um escopo não for bem definido.

Com o passar dos anos, o tamanho das aplicações Android, em termos de armazenamento e de tamanho do código, também têm aumentado. Esse aumento, consequentemente, pode acabar tornando ainda mais complexo o processo de desenvolvimento e afetar a manutenção do software [3]. Isso acaba sendo uma questão importante principalmente porque a grande maioria dos bugs e problemas encontrados em aplicações Android provém de erros no processo de implementação, ao invés de problemas próprios do Sistema Operacional [4]. Por esse motivo, para assegurar a qualidade do software, um processo bem definido e organizado se faz necessário.

Um projeto em parceria com diversas instituições (entre elas a Universidade Federal de Campina Grande, a Universidade Federal de Pernambuco, a prefeitura de Recife, a Universidade de São Paulo, a University College London e a Boğaziçi University), chamado MEWAR, visa desenvolver modelos preditivos sobre a incidência de mosquitos transmissores de doenças, como Zika e dengue, com base em dados socioeconômicos, clima, água e saneamento básico, por exemplo [5]. Um dos meios para conseguir esses dados é a partir do desenvolvimento de uma aplicação mobile. Essa aplicação, desenvolvida para Android, tem o intuito de permitir que agentes de saúde alimentem a base de dados dos modelos preditivos com informações dos locais suscetíveis à reprodução de mosquitos. Também espera-se que esse aplicativo sirva como um sistema de alerta em tempo real às autoridades sobre a reprodução dos mosquitos, a fim de facilitar o desenvolvimento de políticas públicas.

Por estar em uma fase inicial, o desenvolvimento desse aplicativo ainda não tinha processos definidos quanto à implementação do código, ou mesmo um escopo definido para a implementação dos testes. Também faltava documentar os requisitos e as regras para o versionamento do código. A falta de padronização para o versionamento do código, e o compartilhamento desorganizado de diferentes versões do aplicativo foi responsável pela ocorrência de muitos defeitos identificados.

Como mencionado anteriormente, ter as etapas da implementação da aplicação bem definidas é muito importante. Dessa forma, é possível minimizar as chances de

ocorrência de bugs e aumentar as chances de requisitos do sistema, funcionais ou não funcionais, serem satisfeitos.

Este trabalho tem como objetivo propor e implantar um processo de qualidade de software no contexto do desenvolvimento desse aplicativo, determinando o escopo da realização dos testes do software, políticas de entregas, e de avaliação e de validação do aplicativo. Foram definidas ferramentas para a realização de testes de unidade e testes de sistema, tendo sido estes últimos priorizados em decorrência do tempo disponível. Também foi padronizada a documentação dos requisitos, com *User Stories*, a dinâmica de iterações, adotando *sprints*, e os critérios de aceitação para cada nova versão do *software* como sendo estabilidade, implementação dos testes, validação com os requisitos, atualização da documentação e revisões.

A organização deste documento se dá da seguinte forma: A seção 2 mostra os principais conceitos discutidos e fala sobre o projeto onde foi realizado o estudo de caso, a seção 3 fala sobre a metodologia utilizada para a realização deste trabalho, a seção 4 demonstra os resultados, a seção 5 fala sobre as principais limitações encontradas e a seção 6 mostra as conclusões e sugestões para trabalhos futuros.

2. CONTEXTO

Esta seção tem como objetivo explicitar os principais conceitos que serviram como base para o desenvolvimento deste trabalho e o contexto do projeto em que a solução proposta foi inserida.

2.1 Processo de Software

Um processo de software pode ser definido como a formalização de um conjunto de práticas que serão realizadas para cumprir o desenvolvimento de um software. Preferencialmente, essas atividades devem incluir ações que envolvam a especificação de requisitos, um projeto para implementação do código, a implementação propriamente dita e formas de validação e evolução do software implementado. Porém, apesar de serem atividades preferíveis de estarem no processo de software, não existe um processo considerado ideal. As características do software e da equipe de desenvolvimento podem ser fatores determinantes para a tomada de decisões no que diz respeito à aplicação de certas atividades ou não, tornando o procedimento para a definição de um processo de software uma tarefa flexível [6].

2.1.1 Metodologias Ágeis

O Manifesto para Desenvolvimento Ágil de Software [7] afirma que no processo de desenvolvimento de Software devemos dar prioridade às interações com os indivíduos, ao software em funcionamento, à colaboração com o cliente e à resposta rápida à mudanças. Esses aspectos seriam mais importantes do que ter ferramentas e planos criteriosamente definidos com antecedência, com documentações abrangentes, ou com as interações com os clientes se limitando à negociação de contratos.

O processo de desenvolvimento em metodologias ágeis se baseia em refatoramento, desenvolvimento minimalista, revisões de código e integração contínua [17]. O refatoramento consiste em melhorias no código para aprimorar sua performance, manutenibilidade e legibilidade, sem alterar seu comportamento. O desenvolvimento minimalista foca em criar apenas o código necessário para a funcionalidade a ser entregue no momento. As revisões de código são realizadas por outros desenvolvedores para identificar possíveis melhorias. A integração contínua envolve a entrega contínua de funcionalidades, testando-as assim que são entregues e repetindo os testes das funcionalidades anteriores.

O Scrum, por exemplo, destaca-se como uma das metodologias ágeis mais utilizadas para desenvolvimento de software [8]. O Scrum possui uma abordagem iterativa e incremental onde cada iteração, ou *sprint*, pode variar de 2 a 4 semanas. Nessa metodologia, é mantido um *backlog* com as próximas atividades a serem feitas e que é atualizado conforme as sprints avançam. A comunicação entre o time ocorre por meio de reuniões diárias e revisões das atividades realizadas. Usualmente, equipes que usam Scrum possuem um *Product Owner* que é responsável por gerenciar os objetivos do time de desenvolvimento em cada *sprint*, um *Scrum master*, que fica com a função de eliminar possíveis impedimentos no progresso do desenvolvimento e garantir que as práticas da metodologia estão sendo seguidas, além do time de desenvolvimento que trabalha para atingir os objetivos de cada sprint [9].

2.2 Testes de Software

Testes são feitos com o objetivo de identificar possíveis problemas no software antes de ele ser disponibilizado para o usuário final. Eles são parte do processo de verificação e validação do software, onde a verificação confere se o software está de acordo com os requisitos do sistema e a validação checa se o software está de acordo com o esperado pelo cliente. Embora seja a etapa do processo de desenvolvimento onde problemas no software são identificados, os testes não conseguem, uma vez feitos, garantir que o software está livre de problemas, mas são importantes para aumentar a confiabilidade [8].

Por ser possível avaliar com os testes se os requisitos do sistema estão sendo cumpridos ou não, Paul Ammann e Jeff Offutt [10] afirmam que eles funcionam como uma espécie de guardiões do software. Fazer essa avaliação por casos de testes é efetivo porque é mais simples avaliar o comportamento individual de cada caso de teste do que avaliar o comportamento do sistema completo. Assim, cada caso de teste funcionaria como uma forma de aumentar a confiança no funcionamento do software como um todo. Eles continuam dizendo que em processos ágeis, por exemplo, a suíte de testes acaba funcionando como a documentação de requisitos do sistema. Se houver o apontamento de que algum requisito ou comportamento não está sendo cumprido, bastaria modificar ou criar um novo caso de teste correspondente. Dessa forma, os testes teriam um papel central no processo de desenvolvimento de metodologias ágeis, pois uma suíte de testes feita com qualidade aumentaria as chances de sucesso do desenvolvimento.

Existem vários tipos de testes e é possível associá-los a cada fase do desenvolvimento de software. Na fase de análise de requisitos estão associados os testes de aceitação, que verificam a conformidade do software com as necessidades do cliente. Na fase do design da arquitetura dos componentes, os testes de sistema verificam se os componentes do sistema, quando juntos, estão de acordo com a especificação do software. Os testes de integração são úteis para verificar se o software está de acordo com o que foi definido no design dos componentes mais internos, enquanto que os testes de módulos avaliam aspectos mais específicos desse design, como a conformidade na comunicação entre as interfaces dos módulos. Por fim, os testes de unidade estão associados à fase de implementação de software e avaliam cada unidade implementada, como os métodos em Java, por exemplo [10].

A figura 1 ilustra a relação entre o aparecimento de faltas, a detecção delas, e o quanto essa detecção custa quando ocorre em cada etapa do desenvolvimento, desde a especificação dos requisitos, passando pelo design, a implementação com os testes de unidade, a fase de testes de integração, testes de sistema e, por fim, a fase de implantação. É possível notar que quanto mais tardiamente as faltas são identificadas, maior é o custo para resolvê-las. Na fase de implantação, por exemplo, em comparação com as outras fases, foram detectadas menos faltas, mas o custo foi de longe o maior. Isso ocorre porque o trabalho investido para consertar essas faltas depende do trabalho original que as ocasionou, o que é muitas vezes difícil de identificar. Além disso, essa fase também precisa ser revisada.

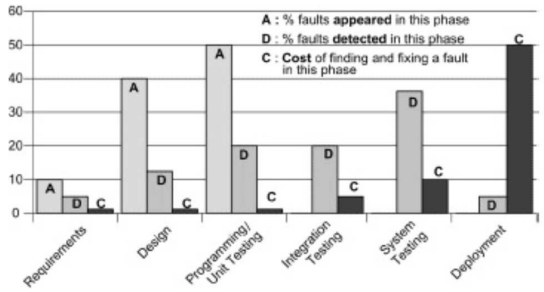

Figura 1. Custo de testar o software tardiamente.
Fonte: [10].

2.2.1 Testes para aplicações Android

Testar aplicações feitas para dispositivos móveis traz algumas preocupações que softwares comuns para desktop ou web, por exemplo, não trazem. A diversidade de aparelhos e seus componentes de hardware fazem com que o processo de testes, assim como o processo de desenvolvimento, seja um pouco mais complexo [11]. A figura 2 ilustra a média do crescimento do tamanho dos aplicativos e a quantidade de classes presentes em cada um deles ao longo dos anos.

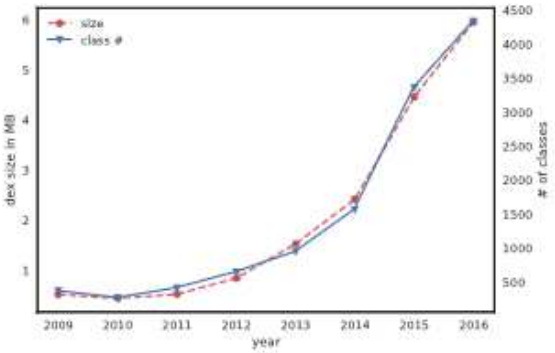

Figura 2. Tamanho dos aplicativos e a quantidade de classes presentes ao longo do tempo.
Fonte: [11].

A existência de sensores como GPS e acelerômetro, e processadores com as mais diversas características resulta que a construção do código da aplicação e os testes para aplicações móveis devem ser *cross-platform*, isto é, não se limitar a emulações [12]. Ao contrário de softwares para computadores, por exemplo, é mais difícil emular certas condições para a realização dos testes. Akour et al. [13] afirmam que o uso de dispositivos reais é essencial no processo de testagem de *software mobile* se o objetivo for analisar a experiência completa do usuário. Eles constataram que emuladores não dão uma indicação próxima da realidade no que se refere a testes que avaliam aspectos como a performance, por exemplo. Eles são ideais, no entanto, para testes de comportamento e de funcionalidades. Portanto, se o objetivo é assegurar o máximo de aspectos de qualidade do software a partir dos testes, é importante a realização de testes em dispositivos reais. Esse fato acaba tornando o processo de testagem mais caro, porque existe uma enorme variedade de dispositivos Android no mercado. Para mitigar esse gasto, um escopo de dispositivos pode ser delimitado.

Li Li et al. [14] realizaram uma revisão bibliográfica onde identificaram que os tipos de testes mais comuns adotados para aplicações *Android* são, respectivamente, testes de sistema, testes de unidade e testes de integração. Os testes de sistema, do tipo caixa-preta, possuíram uma recorrência consideravelmente maior em comparação com os outros tipos de testes. Porém, embora menos recorrentes, os testes de unidade e testes de integração permaneceram estáveis ao longo do tempo. Dentre os objetivos principais dos testes de software em aplicações *Android* estão a detecção de *bugs*, aspectos de segurança e de performance. O alvo principal dos testes usualmente é a interface gráfica de usuário.

2.3 Processo de Qualidade de Software

Software está cada vez mais inserido no nosso cotidiano, fazendo com que sejamos cada vez mais dependentes deles, desde soluções simples, como softwares de mensagens instantâneas, até soluções mais críticas, como as que realizam o controle de pousos em aeroportos. Em 2022, o setor de *softwares* tradicionais representou 13% da participação no macro setor de Tecnologia da Informação e Comunicação, e tem uma estimativa de crescimento de 15,1% para 2023 [15].

Dada a importância desse setor, é necessário que exista uma forma de garantir que os softwares produzidos sejam de qualidade.

Lewis [16] acredita que a qualidade do software está diretamente relacionada com a conformidade com os requisitos do sistema. Para verificar a qualidade, os requisitos precisam ser mensuráveis de modo que seja possível extrair uma resposta binária dessa verificação: o sistema está de acordo com os requisitos do sistema ou não está de acordo. Ele continua dizendo que existem três componentes principais para garantir a qualidade do software: testes, controle de qualidade e as configurações de gerenciamento do software. A figura 3 ilustra esses três componentes principais que são complementados por uma coleção de padrões, convenções, procedimentos e especificações bem definidas.

Figura 3. Componentes de qualidade de software.
Fonte: [16].

A implementação dos testes é um importante componente do processo de qualidade porque a partir dele é possível realizar a verificação e validação do software, desde o design da solução, até o código e a documentação. Também a partir dos testes, é possível conferir se a inserção de novas funcionalidades não alterou o comportamento de funcionalidades previamente adicionadas.

O controle de qualidade está relacionado aos mecanismos usados para assegurar continuamente a conformidade com os requisitos do sistema, seja por meio de revisões dos artefatos ou a garantia de remoção de defeitos antes da entrega do produto. É no controle de qualidade que os defeitos são identificados e corrigidos.

Já as configurações de gerenciamento do software definem como se dá o controle das diferentes versões dos componentes do sistema e como eles se relacionam durante as diferentes fases do processo de desenvolvimento. É nesse processo em que são definidas as estratégias para o versionamento do código, configurações de execução conjunta dos diferentes componentes do sistema e onde as modificações para o sistema são propostas e avaliadas.

2.4 Projeto MEWAR

Mosquito population modelling for early warning system and rapid health authority response (MEWAR) é um projeto interdisciplinar que une áreas como a ciência da computação, meteorologia, modelagem climática e saúde pública com o intuito de avaliar como as mudanças climáticas impactam na incidência de mosquitos transmissores de arboviroses. Com base nessa avaliação, o objetivo do projeto é criar modelos capazes de prever a ocorrência da proliferação desses mosquitos e estar apto a alertar as autoridades competentes para que elas possam tomar providências para mitigar o impacto dessa proliferação.

Esse projeto é uma parceria entre a Universidade Federal de Campina Grande, a Universidade Federal de Pernambuco, a prefeitura de Recife, a Universidade de São Paulo, a University College London (Reino Unido) e a Boğaziçi University (Turquia) e tem seu escopo de atuação atualmente centrada na região Nordeste do Brasil, mais especificamente nas cidades de Campina Grande (PB) e Recife (PE) que já foram marcadas por epidemias de arboviroses.

A fase atual do projeto tem como intuito a criação de um sistema composto por uma aplicação mobile e uma aplicação web de onde serão cadastrados e gerenciados dados, sobre infestações ou potenciais focos de contaminação. A ideia é que os dados sejam coletados nas visitas de rotina às residências da população. A aplicação mobile é desenvolvida para dispositivos Android e deve ser utilizada por agentes de saúde ambiental que farão a coleta dos dados em visitas às residências e irão cadastrar essas informações por meio do aplicativo. Já a aplicação web será utilizada como um painel de controle para a gerência dos agentes e dos dados coletados.

2.4.1 Aplicativo Android

O aplicativo para dispositivos Android possui uma arquitetura no modelo cliente-servidor, composta pelos componentes do *backend*, *frontend* e o banco de dados. O *backend* possui uma API que é responsável por permitir a requisição de todas as funcionalidades da aplicação. O *frontend* é a representação visual da interface do usuário que se comunica e realiza as requisições à API do componente do *backend*. Tanto o *backend* quanto o *frontend* são desenvolvidos em Typescript e o banco de dados utilizado foi o MongoDB.

3. METODOLOGIA

Esta seção explicita os passos adotados para a compreensão do problema e a identificação dos pontos a serem considerados e discutidos para definir e aplicar a proposta do processo de qualidade no desenvolvimento do aplicativo *Android* do projeto MEWAR.

3.1 O Problema

No estágio em que o desenvolvimento do aplicativo estava quando o primeiro autor deste artigo foi inserido no projeto, não havia um processo de software bem definido. Não havia documentação para os requisitos do sistema, ou para o que já havia sido implementado, como a API do *backend*, por exemplo. A equipe de desenvolvimento contava com apenas uma integrante, o que fazia com que todas as etapas do processo de desenvolvimento fossem alocadas para uma única pessoa, tanto a implementação quanto a verificação e validação do software. Esse acúmulo de funções fez com que o processo de verificação e validação não fosse contínuo.

Outro problema que ocorria no desenvolvimento era a falta do uso de uma ferramenta adequada de versionamento de código. Sempre que uma nova versão do software era

implementada, gerava-se um *Android Package* que era disponibilizado no Google Drive para os supervisores do projeto. No entanto, não havia critérios bem definidos para a geração dessa nova versão, exceto a vontade subjetiva da integrante da equipe de desenvolvimento. Além disso, a desenvolvedora não conseguia dedicar-se exclusivamente à implementação do software, porque também atuava em outras áreas do projeto.

Somado ao descrito até então, o projeto tinha prazos curtos a serem cumpridos e existia pouco tempo para colocar o aplicativo em produção, para que pudesse ser testado pelos agentes de saúde ambiental no dia a dia de trabalho. A expectativa era de que fossem disponibilizados aparelhos *Android* para serem distribuídos entre esses agentes, e isso demandava um planejamento rigoroso e fazia-se necessário um processo de desenvolvimento ágil.

3.2 Definição do Processo de Qualidade

A partir de estudos e discussões com a equipe do projeto, foram definidas as estratégias que melhor se adequavam ao contexto do aplicativo quanto ao desenvolvimento dos testes, de versionamento de código, documentação dos requisitos, e os procedimentos para colocar o software em produção. A seguir, estão descritas as etapas realizadas.

3.2.1 Revisão na literatura

Inicialmente foi realizada uma pesquisa na literatura sobre boas práticas em desenvolvimento para aplicativos móveis, em especial aqueles desenvolvidos para o Android, com o objetivo de identificar os principais padrões e processos de desenvolvimento nessa plataforma, focando nos tipos de testes e ferramentas mais utilizadas. A partir dessa pesquisa, ficou definido que o foco deveria ser nos testes de unidade para a avaliação estrutural do código, e nos testes de sistema para a avaliação funcional, por terem sido identificados como os tipos de testes mais relevantes para o contexto do desenvolvimento *Android*.

3.2.2 Avaliação da aplicação MEWAR

Uma vez conhecidas as principais técnicas do desenvolvimento de aplicações Android, foram feitos estudos com a documentação do projeto MEWAR com o objetivo de contextualizar a proposta do aplicativo móvel. A primeira ação foi a de realizar reuniões com a principal representante dos requisitos do aplicativo sob o ponto de vista dos agentes de saúde ambiental, os usuários alvo do aplicativo. A partir dessas reuniões, foram definidas as *User Stories*, que seriam a base para o restante do desenvolvimento do aplicativo.

Quanto à implementação do código, identificou-se que até então era seguido um processo semelhante ao modelo em cascata, isto é, as funcionalidades foram implementadas e só depois foram criados os testes. O único tipo de testes implementado eram os testes de unidade, mas acabou que no estágio em que o desenvolvimento estava havia poucos testes implementados. Apenas o *backend* possuía uma boa cobertura de testes de unidade. Por esse motivo, foi decidido que seguiríamos um modelo de desenvolvimento ágil, com os testes sendo implementados concomitantemente com o código da aplicação.

Ainda nessa etapa de avaliação do código, aproveitou-se para realizar um relatório com problemas identificados na aplicação, quanto a faltas ou falhas presentes no sistema. Assim que foi escrito, esse relatório foi imediatamente disponibilizado para a desenvolvedora.

Com base nas *User Stories* e no que já estava implementado da aplicação, também foi construído um documento com os principais cenários de testes, para futuramente ser usado na confecção dos testes de sistema. Um cenário de testes, nesse contexto, descreve as etapas e as condições para a realização de uma funcionalidade do sistema, seja com condições válidas ou inválidas, e seus resultados esperados. Uma *User Story* poderia gerar mais de um cenário de teste, e um cenário poderia gerar mais de um caso de teste. Cada cenário de teste consistia de pré-condições e passos necessários para atingi-lo, como também dos dados de testes utilizados nesse processo e dos resultados esperados. A ideia é que esse documento com os cenários de testes tenha atualizações constantes, conforme o software evolua.

3.2.3 Definição do processo

Dada a conjuntura do projeto, foram definidos alguns pontos de ação para trazerem melhorias para o andamento do desenvolvimento do aplicativo. A seguir estão listados os pontos que foram considerados:

- Suprir a falta de desenvolvedores da aplicação;
- Criar artefatos para documentar os requisitos e o código da aplicação, para facilitar a inserção de novos integrantes à equipe de desenvolvimento e a manutenibilidade do sistema;
- Realizar a implementação dos tipos de testes mais relevantes para o contexto da aplicação;
- Definir o versionamento do código e as entregas contínuas;
- Adotar o formato de Sprints, com reuniões diárias, para organizar as entregas do software e propiciar o desenvolvimento ágil.

Ficou decidido que o *GitHub* seria usado como o repositório principal do projeto e onde também aconteceria o versionamento do código. As regras de versionamento foram seguidas com base na documentação do *SemVer*[1], onde cada versão poderia ter uma alteração *MAJOR*, *MINOR* ou *PATCH*.

Também foi adotado o uso de *Sprints* de 15 dias para a entrega contínua do *software*. Em todo início de Sprint seria discutido quais tarefas seriam desenvolvidas e quem seriam os desenvolvedores responsáveis por cada uma. Cada tarefa deveria gerar um *Pull Request* no *GitHub*, e o código implementado só poderia ser enviado para a branch principal do projeto, isto é, a branch que gerará ao fim da *Sprint* uma nova versão, se os testes correspondentes estivessem implementados. Se passasse nos testes de sistema e quando pelo menos um dos integrantes fizesse a revisão e aprovasse

[1] https://semver.org/

as alterações, ele poderia ser enviado para a branch principal e estaria apto a gerar uma nova versão. Haveria reuniões diárias para comunicar o progresso individual das atividades e, ao final de cada *Sprint*, um novo *Android Package* seria gerado contendo as alterações mais recentes do software.

Para ser possível gerar uma nova versão, o código deveria ter as seguintes características:

- Estabilidade: todos os comandos devem estar funcionando sem quebrar;
- Testado: ter os testes de unidade correspondentes implementados;
- Validado: passar em todos os cenários de testes de sistema;
- Documentado: documentação atualizada e de acordo com as alterações realizadas;
- Revisado: ter sido revisado e aprovado por pelo menos um dos integrantes do projeto;

Os documentos de apresentação dos repositórios não eram atualizados há um tempo e possuíam instruções para rodar que não correspondiam com o então estado do desenvolvimento. Outras faltas foram notadas na documentação dos requisitos do sistema. Por esses motivos, priorizou-se realizar a atualização dessa documentação, com o objetivo de facilitar a contextualização do processo de desenvolvimento para futuros novos integrantes e melhorar a manutenibilidade do sistema.

Os tipos de testes identificados como os mais relevantes para o desenvolvimento do aplicativo nesse primeiro momento, como mencionado anteriormente, foram os testes de unidade e os testes de sistema. Ficou definido que os testes de unidade deveriam ser criados por cada desenvolvedor para cada implementação ou alteração feita no código por ele. Os testes de sistema seriam feitos com base no documento de cenários de testes e deveria servir como um verificador antes de cada nova versão do aplicativo ser gerada.

Todas as orientações descritas nessa seção foram mantidas em um documento de diretrizes[2] a serem seguidas. Esse documento foi previamente discutido e aprovado pelos integrantes do projeto.

3.3 Avaliação do processo de qualidade

Ao final, foi aplicado um formulário online com os envolvidos no projeto com o objetivo de avaliar a eficácia das ações tomadas para melhorar a qualidade do processo de desenvolvimento. Foram feitas afirmações, onde cada afirmação poderia ser avaliada em uma faixa de 0 a 5 em que 0 significa discordância total e 5 significa que a pessoa concorda completamente com a afirmação. Ao todo foram feitas as 6 afirmações listadas abaixo:

- O processo de qualidade foi efetivo em detectar defeitos na aplicação;
- O processo de qualidade contribuiu para a redução de defeitos na aplicação;
- O processo de qualidade melhorou a confiabilidade da aplicação;
- As diretrizes e instruções do processo de qualidade eram claros e concisos;
- O processo de qualidade foi apropriadamente adaptado às características e necessidades do projeto;
- A comunicação sobre as etapas e os objetivos do processo de qualidade com a equipe do projeto foi eficiente.

4. RESULTADOS

Esta seção explicita os resultados dos passos definidos para a aplicação do processo de qualidade de software, desde a conformidade com o que havia sido discutido, até as alterações ocorridas no decorrer do processo.

4.1 Documentação

A primeira ação foi a de adicionar mais uma pessoa ao time de desenvolvimento, a fim de suprir a carência de desenvolvedores e desafogar todas as obrigações da única integrante até então. O integrante adicionado foi um aluno da graduação de Ciência da Computação da UFCG que aceitou participar do projeto como voluntário. Baseado na experiência do primeiro autor deste artigo e do integrante recém chegado a equipe de desenvolvimento, notou-se que os repositórios careciam de um documento de apresentação atualizado com os comandos para inicialização da aplicação e para a execução dos testes. Portanto, foi designado ao novo desenvolvedor a tarefa de atualizar e padronizar essas documentações com o objetivo de familiarizá-lo com o código implementado até então. Também foi pedido que fosse feita a documentação da API do *backend* com a descrição das rotas disponíveis e instruções de como realizar as requisições e o que significava cada resposta retornada.

Em paralelo, para guiar o restante do processo de desenvolvimento, foram realizadas reuniões com a representante dos usuários alvo da aplicação, os agentes de saúde ambiental, para definir um documento de *User Stories*. A partir desse documento foi possível manter um *backlog* com as funcionalidades que ainda não estavam implementadas. A tabela 1 descreve as seis *User Stories* que foram definidas:

ID	*User Story*
1	Enquanto agente, quero realizar login no sistema, para que eu possa utilizar as funcionalidades disponíveis para agentes.
2	Enquanto agente, quero editar as informações do meu perfil, para que elas possam ser atualizadas.
3	Enquanto agente, quero pesquisar por minhas visitas passadas, para que eu possa recuperar essas informações
4	Enquanto agente, quero visualizar as visitas que me foram designadas, para que eu possa saber quais

[2] Guidelines For Development Process

	localidades realizar a visitação.
5	Enquanto agente, quero criar um formulário de visita, para cadastrar as informações obtidas durante cada visitação, como a identificação da existência de infestações e focos de contaminação.
6	Enquanto agente, quero cadastrar as informações obtidas nas visitas mesmo sem internet, para que não haja impedimentos em localidades onde o acesso à internet é limitado

Tabela 1. *User Stories* do aplicativo *Android* do projeto MEWAR.
Fonte: Autoria própria.

Com base no documento de *User Stories* foi possível construir um documento com os principais cenários de testes da aplicação que seriam usados posteriormente para facilitar a implementação dos testes de sistema. Esses cenários descreviam o passo a passo para realizar as funcionalidades do sistema a partir da interface do usuário no *frontend*. Para reproduzí-los, era necessário que o sistema estivesse sendo executado por completo, tanto o *frontend* como o *backend* e a instância do banco de dados. Ao todo foram escritos 6 cenários de testes, que geraram 26 casos de testes. Nesses testes estavam contemplados as *User Stories* de 1 à 4. As *User Stories* 5 e 6 ainda não haviam sido implementadas, o que fez com que não fosse possível nesse momento simular o comportamento do usuário no *frontend* da aplicação para simulá-los. Mas a intenção é que o documento de cenário de testes seja constantemente atualizado, conforme novas funcionalidades apareçam ou as funcionalidades existentes sejam alteradas.

4.2 Identificação de defeitos

Durante a confecção do documento com os cenários de testes, aproveitou-se para registrar os defeitos encontrados na aplicação. Estes defeitos variam desde condições não tratadas no *backend* até problemas com a interface do usuário no *frontend*. Ao todo, foram identificados 12 defeitos.

Um dos defeitos encontrados, por exemplo, foi a da expiração do token de autenticação que, quando ocorria, impedia o usuário de realizar qualquer operação que precisasse de autenticação, mas não havia indicativo visual ou qualquer tipo de redirecionamento. A tabela 2 mostra um exemplo de como esse defeito foi relatado, com a descrição do problema, o componente alvo, a prioridade para resolução e os passos para reproduzi-lo (entrada, saída atual e saída esperada).

Name	Token expiration with no redirection
Description	When the token expires and I do some action that needs authentication, it is not redirecting to the login screen. I only know that he is logged out because of the backend request that comes with unauthorized
	401 status.
Target	Frontend
Priority	Medium
Input	Remove token or wait until it expire
Actual output	No visual cues or redirection.
Expected output	Redirection to login screen

Tabela 2. Descrição do problema de expiração do *token*.
Fonte: Autoria própria.

4.3 Iterações

Ao todo, o processo de qualidade foi aplicado no período de uma iteração de 15 dias. Por questões de conflitos de horários disponíveis dos membros da equipe, optou-se por substituir as reuniões diárias pelo preenchimento de um formulário online com as atualizações do processo de desenvolvimento. A ideia era que o preenchimento deste formulário fosse a primeira atividade do dia, onde os desenvolvedores poderiam reportar o que foi feito no dia anterior, possíveis imprevistos ou impedimentos e o que estava planejado para ser feito no dia atual. Também neste formulário existia um campo para preencher com a quantidade de horas gastas nas atividades reportadas, com a finalidade de facilitar estimativas para as atividades futuras. O formulário acabou tendo uma boa adesão no início da *Sprint*, mas posteriormente seu uso diminuiu e a comunicação informal por outros meios prevaleceu.

Embora não tivesse sido possível ter as reuniões diárias, foi mantido uma reunião com frequência quinzenal, pelo *Skype*, onde seriam discutidas as tarefas finalizadas durante a *Sprint* e as tarefas a serem feitas na iteração seguinte.

4.4 Implementação dos testes

A maioria dos testes de unidade que estavam implementados antes da aplicação do processo de qualidade eram do *backend* e correspondiam a pouco mais de 70% de cobertura. Como o tempo era curto, optou-se por dar prioridade aos testes de sistema, do tipo *end-to-end*, por serem os testes mais feitos no contexto de aplicações *Android* e por terem sido julgados como mais relevantes para o usuário final. O *framework* escolhido foi o *Cypress* [18], e foram implementados 21 casos de testes dos 26 planejados no documento de cenários de testes. Ainda não foi possível alcançar o número total de casos de testes porque o *Cypress* era uma ferramenta nova para os desenvolvedores e algumas funcionalidades estão sendo estudadas. Nesse processo foram descobertos mais defeitos na aplicação, que foram colocados como *backlog* das atividades de desenvolvimento.

Para os testes de unidade, ficou decidido que iríamos usar as bibliotecas que já estavam presentes no código, *Mocha* [19] e *Chai* [20], e que elas iriam começar a ser utilizadas na *Sprint* seguinte, conforme as atividades de desenvolvimento fossem sendo feitas. Portanto, para essa *Sprint*, a pré-condição de o código estar com os testes de

unidade implementados para ser aprovado na revisão não foi adotada.

Em concomitância com a implementação, uma ferramenta de análise estática do código era constantemente executada pelos desenvolvedores, a fim de garantir a qualidade da escrita do código. Por já ter sido utilizada anteriormente, a ferramenta escolhida foi o *Eslint* [21], que realiza a análise estática de código em *Typescript*. Entre as métricas escolhidas para o *Eslint* estava a verificação de variáveis não utilizadas, importações duplicadas, uso desnecessário de *logs* no *console*, dentre outros.

4.5 Versionamento do código

O período de uma iteração, ao qual o processo de qualidade foi aplicado, foi considerado curto e não o suficiente para termos segurança de lançar a primeira versão oficial do software. A ideia é que a implementação dos testes e a resolução dos principais defeitos seja priorizada nas próximas *sprints* para que a primeira versão cumpra os requisitos propostos de estabilidade, testabilidade e a validação com os testes de sistema.

4.6 Formulário de avaliação

A seguir estão sumarizadas as afirmações presentes no formulário e abaixo está o gráfico 1 mostrando os resultados para cada afirmação na escala *Likert* de 1 à 5.

Q1:*O processo de qualidade foi efetivo em detectar defeitos na aplicação*
Q2*: O processo de qualidade contribuiu para a redução de defeitos na aplicação*
Q3*: O processo de qualidade melhorou a confiabilidade da aplicação*
Q4*: As diretrizes e instruções do processo de qualidade eram claros e concisos*
Q5*: O processo de qualidade foi apropriadamente adaptado às características e necessidades do projeto*
Q6*: A comunicação sobre as etapas e os objetivos do processo de qualidade com a equipe do projeto foi eficiente*

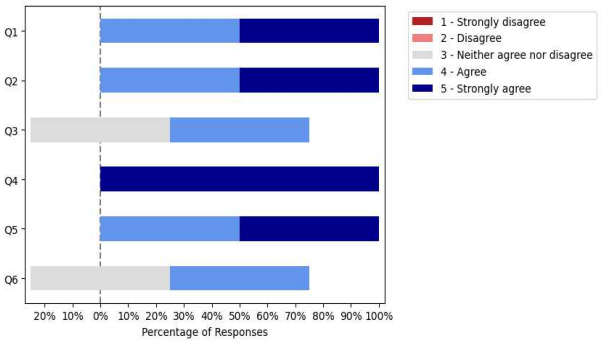

Gráfico 1. Respostas ao formulário de avaliação do processo de qualidade.
Fonte: Autoria própria.

O formulário evidencia que os membros do projeto concordam com a afirmação de que as diretrizes e instruções do processo de qualidade foram claras e concisas, e também há concordância quanto às afirmações de que o processo foi efetivo em detectar defeitos na aplicação, em reduzir esses defeitos e que foi adequadamente adaptado às características e necessidades do projeto. As afirmações onde a concordância não foi tão contundente foram aquelas relacionadas à melhoria da confiabilidade da aplicação e à eficiência da comunicação entre os membros da equipe do projeto.

5. LIMITAÇÕES

A validade do processo de qualidade de software definido neste trabalho sofreu com algumas limitações não previstas na aplicação do processo. Estes imprevistos fizeram com que o processo só pudesse ser aplicado pelo período de uma *sprint* de 15 dias. Esse curto período de tempo foi visto como um ponto que dificultou enxergar os reais benefícios da aplicação do processo, assim como também para adicionar as sugestões ou adaptações propostas no decorrer da aplicação.

A concentração do trabalho do desenvolvimento em uma pessoa da equipe era um problema que deveria ter sido suprido com a adição do aluno voluntário da UFCG, mas acabou não sendo tão eficaz porque a pessoa que trabalhava no desenvolvimento até então precisou se ausentar do projeto, deixando o aluno da UFCG como único responsável momentaneamente pelo processo de desenvolvimento. Assim o problema da carência de desenvolvedores se manteve, e a fase de revisão de código foi prejudicada porque não haveria outro desenvolvedor para realizar esta tarefa. O novo desenvolvedor também não tinha conhecimento sobre o desenvolvimento antes da aplicação do processo de qualidade, para conseguir opinar com mais segurança sobre os benefícios obtidos após a aplicação.

O processo teve uma priorização na implementação dos testes, e o fato de a maioria das funcionalidades já ter sido implementada e os responsáveis por implementá-las não estarem disponíveis para ajudar na implementação dos testes foi um fator que atrapalhou o andamento desta fase, principalmente na implementação dos testes de unidade. Foi por esse motivo que optamos por priorizar os testes *end-to-end*.

Como não foi possível avançar na implementação dos tipos de testes escolhidos, o que ocasionou o adiamento do lançamento da primeira versão estável do *software*, também não foi possível realizar testes em dispositivos reais ou ter a validação por meio de testes de usabilidade. O escopo dos dispositivos reais onde serão feitos os testes ainda está para ser decidido pelo departamento responsável do MEWAR que disponibilizará os aparelhos para os testes.

Houveram também algumas falhas de comunicação que dificultaram o progresso do desenvolvimento. A comunicação era predominantemente feita por trocas de e-mails, e a equipe principal de desenvolvimento era do Reino Unido, o que já poderia ser um problema por causa da barreira da língua. Mas, mesmo quando a comunicação ocorria, havia um grande atraso nas respostas, o que prolongou o tempo para a realização de ações como a de dar permissão de acesso aos repositórios da aplicação ao novo desenvolvedor.

6. CONCLUSÕES E CONSIDERAÇÕES FUTURAS

Este trabalho teve como objetivo realizar um estudo de caso sobre a implantação de um processo de qualidade de software no contexto do desenvolvimento do aplicativo para dispositivos *Android* do projeto MEWAR. A elaboração do processo de qualidade foi feita com base em pesquisas na literatura e discussões com a equipe de desenvolvimento. O foco principal da aplicação do processo foi a identificação de defeitos da aplicação. Para isso, utilizou-se de mecanismos de verificação e validação, além da padronização do processo de desenvolvimento e o gerenciamento do código e suas versões.

A implantação do processo de qualidade representou um importante avanço para o desenvolvimento do aplicativo MEWAR porque evidenciou falhas do processo de desenvolvimento e contribuiu para contorná-las. A partir da concepção de artefatos de documentação e da implementação de testes, em especial os testes de sistema, foi possível identificar defeitos na aplicação que por muito tempo passaram despercebidos.

A adoção do *GitHub* como ferramenta para versionamento de código ajudou no processo de desenvolvimento porque facilitou a realização de revisões de código, proporcionou uma maior colaboração entre o time de desenvolvimento, além de permitir uma maior rastreabilidade para os problemas da aplicação.

Apesar de ter sido possível observar benefícios na implantação do processo de qualidade, aspectos como o tamanho da equipe, tempo de implantação do processo e a comunicação entre o time foram vistos como pontos negativos. Notamos que o período de uma iteração foi pouco para garantir confiabilidade na avaliação das diretrizes adotadas, visto que, mesmo nesse curto período de tempo, houveram mudanças que fizeram com que o processo não pudesse atingir um ciclo completo até o lançamento da primeira versão estável do *software*. O tamanho da equipe de desenvolvimento, composta por dois desenvolvedores, também se mostrou como um ponto negativo, porque não havia margem para imprevistos. Se um dos desenvolvedores precisasse se ausentar, por exemplo, o processo de revisão do código e de implementação dos testes seria prejudicado. Problemas de comunicação também foram um dos fatores que atrapalharam algumas decisões do processo, dado que a comunicação ocorria predominantemente por email e havia atrasos nas respostas.

Portanto, para trabalhos futuros, recomenda-se a aplicação do processo de qualidade por um período de tempo maior. Desta forma, a avaliação do processo seria mais abrangente, possibilitando a inserção de sugestões de melhorias e permitindo que essas sugestões também possam ser avaliadas e consolidadas. Também enxergou-se que com uma equipe maior que 2 desenvolvedores e uma comunicação mais direta e constante a aplicação e a avaliação do processo seriam beneficiadas.

Por um momento, cogitamos utilizar algum processo de desenvolvimento consolidado, a exemplo do Scrum. No fim, optamos por priorizar um processo mais customizável, pegando elementos de vários processos existentes e adaptando-os à realidade do projeto MEWAR. Para o futuro, talvez seja interessante aplicar um processo consolidado e comparar a aceitação por parte da equipe do projeto.

As etapas específicas do processo de qualidade para o desenvolvimento de aplicações *Android* serão feitas no futuro por meio de testes de usabilidade e validação de requisitos não funcionais em dispositivos reais. Como não conseguimos concluir um ciclo de desenvolvimento a tempo, as etapas referentes ao *Android* não puderam ser demonstradas. Embora tenha sido idealizado para o caso específico dessa aplicação, entendemos que as etapas do processo que foram aplicadas até o momento não as deixam limitadas a processos de desenvolvimento para este sistema operacional.

7. REFERÊNCIAS

[1] Laricchia, Federica. *Mobile operating systems' market share worldwide from 1st quarter 2009 to 4th quarter 2022*. Disponível em <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009>.

[2] *Why Android OS is More Popular Among Developers?*. 2022. Disponível em: <https://www.appsquadzeducation.com/blog/why-android-os-is-more-popular-among-developers>.

[3] Jun Gao, Li Li, Tegawendé F. Bissyandé, Jacques Klein. *On the Evolution of Mobile App Complexity*. 2019. 24th International Conference on Engineering of Complex Computer Systems (ICECCS).

[4] Amiya Kumar Maji, Kangli Hao, Salmin Sultana, and Saurabh Bagch. *Characterizing Failures in Mobile OSes: A Case Study with Android and Symbian*. Disponível em <https://engineering.purdue.edu/dcsl/publications/papers/2010/android_issre10_submit.pdf>.

[5] *Mosquito population modelling for early warning system and rapid health authority response (MEWAR)*. Disponível em <https://www.ucl.ac.uk/risk-disaster-reduction/research-projects/2020/sep/mosquito-population-modelling-early-warning-system-and-rapid-health>.

[6] Sommerville, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

[7] *Manifesto for Agile Software Development*. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>.

[8] Al-Saqqa, Samar. *Agile Software Development: Methodologies and Trends*. 2020. Disponível em: <https://online-journals.org/index.php/i-jim/article/view/13269/7405>.

[9] Anand, R. Vijay; Dinakaran, Dr. M. *Popular Agile Methods in Software Development: Review and Analysis*. In:

International Journal of Scientific and Technical Advancements, Volume 2, Issue 4, pp. 147-150, 2016.

[10] Ammann, Paul; Offutt, Jeff. *Introduction to software testing*. 2. ed. New York: Cambridge University Press, 2017.

[11] Li Li, Pingfan Kong; Bissyandé, Tegawendé F.; Klein, Jacques. *On the Evolution of Mobile App Complexity*. In: *International Conference on Engineering of Complex Computer Systems (ICECCS)*, 24, 2019.

[12] Amalfitano, Domenico; Fasolino, Anna Rita; Tramontana, Porfirio; Robbins, Bryan. *Testing Android Mobile Applications: Challenges, Strategies, and Approaches*. In: Advances in Computers, Volume 89, 2013.

[13] Akour, Mohammed; Falah, Bouchaib; Al-Zyoud, Ahmad A.; Bouriat, Salwa; Alemerien, Khalid. *Mobile Software Testing: Thoughts, Strategies, Challenges, and Experimental Study*. *International Journal of Advanced Computer Science and Applications*, Vol. 7, No. 6, 2016.

[14] Li Li, Pingfan Kong; Kui Liu, Jun Gao; Bissyandé, Tegawendé F.; Klein, Jacques. *Automated Testing of Android Apps: A Systematic Literature Review*. 2018.

[15] Brasccom. Relatório Setorial 2022: Macrossetor de TIC. 2023. Disponível em <https://brasscom.org.br/wp-content/uploads/2023/05/BRI2-2023-008-001-Relatorio-Setorial-v32-versao-resumida-SITE_compressed.pdf>.

[16] Lewis, William E. *Software Testing and Continuous Quality Improvement*, 3. ed. Boca Raton: Taylor & Francis, 2008.

[17] Coram, Michael; Bohner, Shawn. *The Impact of Agile Methods on Software Project Management*. In: *IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 12, 2005.

[18] *Cypress: Frontend testing tool*. Versão 12.13.0. 2023. Disponível em <https://www.cypress.io>.

[19] *Mocha: JavaScript test framework*. Versão 7.1.1. 2020. Disponível em <https://mochajs.org>.

[20] *Chai: JavaScript BDD/TDD assertion library*. Versão 7.1.1. 2020. Disponível em <https://www.chaijs.com>.

[21] *Eslint: Static code analyzer*. Versão 6.6.0. 2019. Disponível em <https://eslint.org>.