CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE CAMPINAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E
DESENVOLVIMENTO DE SISTEMAS

MARIA CECILIA BAPTISTELLA TREVISANI

VALIDAÇÃO DA QUALIDADE DO SOFTWARE ATRAVÉS DE TESTES FUNCIONAIS.

CAMPINAS/SP
2022

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE CAMPINAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MARIA CECILIA BAPTISTELLA TREVISANI

VALIDAÇÃO DA QUALIDADE DO SOFTWARE ATRAVÉS DE TESTES FUNCIONAIS.

Trabalho de Graduação apresentado por **Maria Cecilia Baptistella Trevisani**, como pré-requisito para a conclusão do Curso Superior de Tecnologia em **Análise e Desenvolvimento de Sistemas**, da Faculdade de Tecnologia de Campinas, elaborado sob a orientação do Prof. Me. **Anderson Luiz Barbosa.**

CAMPINAS/SP
2022

FICHA CATALOGRÁFICA

CEETEPS - FATEC Campinas – Biblioteca

T814v

TREVISANI, Maria Cecilia Baptistella
Validação da qualidade do software através de testes funcionais. Maria Cecilia Baptistella Trevisani. Campinas, 2022.
83 p.; 30 cm.

Trabalho de Graduação do Curso de Análise e Desenvolvimento de Sistema – Faculdade de Tecnologia de Campinas, 2022.
Orientador: Prof. Me. Anderson Luiz Barbosa.

1. Testes. 2. Caixa-preta. 3. Qualidade de software. 4. Software.
I. Autor. II. Faculdade de Tecnologia de Campinas. III. Título.

CDD 005.3

Catalogação-na-fonte: Bibliotecária: Aparecida Stradiotto Mendes – CRB8/6553

TG ADS 22.2

Maria Cecilia Baptistella Trevisani

Validação da Qualidade do Software através de Testes Funcionais

Trabalho de Graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pelo CEETEPS / Faculdade de Tecnologia – Fatec Campinas.

Campinas, 05 de dezembro de 2022.

BANCA EXAMINADORA

Anderson Luiz Barbosa

Prof. Anderson Luiz Barbosa
Fatec Campinas

Anderson Luiz Coan

Prof. Anderson Luiz Coan
Fatec Campinas

Documento assinado digitalmente
gov.br
CLEBER CARVALHO PEREIRA
Data: 05/12/2022 20:22:36-0300
Verifique em https://verificador.iti.br

Prof. Cleber Carvalho Pereira
Fatec Campinas

RESUMO

O presente estudo pretende mostrar a importância dos Testes Funcionais de software para a qualidade dos sistemas de informação. Com o mercado demandando cada vez mais por soluções tecnológicas, cresceu também o número de empresas dedicadas à esta atividade. Com o aumento da competitividade, as empresas desenvolvedoras de software precisaram voltar sua atenção para a qualidade, como forma de se destacar no mercado. Os testes são parte importante para que o software atinja a qualidade desejada tanto pela empresa, quanto pelo usuário. São muitos os tipos de testes que podem ser executados em um sistema e entre eles estão os Testes Funcionais, também conhecidos como "Testes de Caixa Preta", que tem por objetivo validar as funcionalidades e os requisitos do software. O problema pesquisado foi identificar de que forma os testes funcionais contribuem a para o desenvolvimento de sistemas de informação de qualidade. No estudo foi realizado uma pesquisa bibliográfica sobre testes de software, com foco nos testes funcionais e como eles contribuem para aumentar a qualidade do sistema. Os conhecimentos adquiridos durante a elaboração do estudo foram aplicados no sistema web LevantAí, que foi desenvolvido em grupo, na Disciplina de Laboratório de Engenharia de Software. Foram criados casos de testes com base nos requisitos funcionais, executados testes manuais e implementado testes automatizados, resultando na melhorar qualidade do sistema.

Palavras-chave: testes; caixa-preta; qualidade de software; softwares.

ABSTRACT

The present study intends to show the importance of Software Functional Tests for the quality of information systems. With the market increasingly demanding for technological solutions, the number of companies dedicated to this activity also grew. With the increase in competitiveness, software development companies needed to turn their attention to quality, as a way to stand out in the market. Tests are an important part for the software to reach the quality desired by both the company and the user. There are many types of tests that can be performed on a system and among them are the Functional Tests, also known as “Black Box Tests”, which aim to validate the functionality and requirements of the software. The problem researched was to identify how functional tests contribute to the development of quality information systems. In the study, a bibliographic research was carried out on software testing, focusing on functional tests and how they contribute to increasing the quality of the system. The knowledge acquired during the elaboration of the study was applied in the LevantAí web system, which was developed in a group, in the Software Engineering Laboratory Discipline. Test cases were created based on functional requirements, manual tests were performed and automated tests were implemented, resulting in improved system quality.

Keywords: tests; black box; software quality; software.

LISTA DE QUADROS/TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Há uma grande demanda do mercado por softwares e, por consequência, um número cada vez maior de empresas desenvolvedoras, assim criar softwares de qualidade é fundamental neste cenário, que é cada vez mais competitivo. Os testes são parte importante para que o software atinja a qualidade desejada tanto pela empresa, quanto pelo usuário. São muitos os tipos de testes que podem ser executados em um sistema e entre eles estão os Testes Funcionais, também conhecidos como "Testes de Caixa Preta", que tem por objetivo validar as funcionalidades e os requisitos do software. Segundo Machado (2016) são nos Testes Funcionais que se encontram os maiores índices de falhas de um projeto. A execução dos testes podem ser feita de forma manual ou automatizada, sendo o mais comum que no planejamento dos testes, sejam incluídas as duas formas. Os conhecimentos adquiridos durante a elaboração do estudo foram aplicados no sistema Web LevantAí, que foi desenvolvido em grupo, na Disciplina de Laboratório de Engenharia de Software e foi submetido a testes funcionais, para validar sua qualidade.

1.2 JUSTIFICATIVA/PROBLEMÁTICA

Cada vez mais, se tem o entendimento de que para um software atingir seus objetivos com sucesso é preciso ter foco na qualidade. Portanto não basta desenvolver um sistema, é preciso que ele tenha aderência aos requisitos, que por sua vez, precisam estar em conformidade com o desejo do cliente.

A motivação para o desenvolvimento deste estudo é responder à pergunta: Como os testes funcionais auxiliam na qualidade de um sistema?

Além de responder esta questão, também foram aplicados os conhecimentos adquiridos neste estudo, executando testes manuais e automatizados, como forma de validar a qualidade da aplicação Web LevantAí, que foi desenvolvida em grupo na Disciplina de Laboratório de Engenharia de Software.

1.3 OBJETIVOS

O presente estudo abordou a importância dos Testes Funcionais de software para a qualidade de um sistema. As questões pesquisadas foram identificar de que forma os Testes Funcionais contribuem para o desenvolvimento de sistemas de informação com qualidade, se vale a pena automatizar testes em sistemas de baixa complexidade e se sistemas simples ganham mais qualidade com a aplicação de testes funcionais. Os testes funcionais foram aplicados no software LevantAí, como forma de verificar a qualidade da aplicação.

Foi realizada uma revisão bibliográfica sobre o tema, incluindo qualidade de software, tipos de testes, testes funcionais, casos de teste, execução de testes e automação de testes.

Sobre aplicação LevantAí, vale esclarecer que consiste em um site inspirado no método Pomodoro[1], porém incrementado com exercícios físicos laborais e que tem como foco as pessoas que trabalham ou estudam muitas horas em frente ao computador. O funcionamento dele é simples, o usuário ao começar o seu dia de trabalho – em frente a tela – acessa a aplicação, dispara o cronômetro e ao final do tempo, receba aviso para fazer pausa e também uma sugestão de atividade para ser realizada neste período. Nas pausas o usuário é incentivado a fazer exercícios laborais e atividades como beber água, ir ao banheiro ou até mesmo olhar para pontos distantes, o objetivo é minimizar os problemas causado ao corpo, quando o indivíduo passa muitas horas sentado na frente do computador, como dores musculares, problemas circulatórios, visão cansada, pouca hidratação, entre outros.

[1] *O Método Pomodoro é uma forma de gerenciamento de tempo que consiste, inicialmente em intercalar períodos de foco de 25 minutos, com períodos de pausa de 5 minutos. O objetivo é auxiliar os indivíduos a se concentrarem melhor nas suas atividades (CONEXÃO PUC MINAS, 2022).*

2 REVISÃO BIBLIOGRÁFICA

Com o mercado demandando cada vez mais por soluções tecnológicas, cresceu também o número de empresas dedicadas à esta atividade. Com o aumento da competitividade, as empresas desenvolvedoras de software precisaram voltar sua atenção para a qualidade, como forma de se destacar no mercado. Entre as estratégias para desenvolver um software de qualidade está o processo de testes. Segundo Sommerville (2011), o objetivo do teste é demonstrar que o sistema faz o que foi proposto a fazer e também para descobrir defeitos do programa antes da entrega ao cliente. Quando um sistema é testado, o intuito é adicionar valor a ele, aumentando a qualidade e a confiabilidade do programa, através da localização e remoção de erros (MYERS, 2004).

Segundo Pressman (2011), é essencial para qualquer empresa de produtos que serão usados por terceiros, as atividades de qualidade e o controle de qualidade. A história da qualidade no desenvolvimento de software, caminha junto com a história da fabricação de hardware. Nas décadas de 1950 e 1960, a qualidade do sistema era de inteira responsabilidade do programador, mas a partir da década de 1970, elementos distintos passam a ter responsabilidade sobre a garantia de qualidade do software, engenheiros, gerentes de projeto, clientes, vendedores e os indivíduos que trabalham no grupo de Gestão da Qualidade de Software, onde estão incluídos os testers. O objetivo do teste é encontrar erros e um teste eficaz é o que tem grande probabilidade de encontrá-los. Os testes são um conjunto de atividades planejadas e executadas, com base em técnicas de projetos de casos de teste e métodos de teste.

Uma forma de definir a qualidade de um software é através do número de requisitos que foram efetivamente testados e em conformidade com o que foi especificado. Para obter maior qualidade do software e menor riscos de erros é necessária uma maior extensão dos testes, testando e validando adequadamente mais cenários (BARTIÊ, 2002).

Assim, é possível observar que qualidade é essencial para a comercialização de qualquer produto, e não seria diferente com os softwares. Os testes são parte importante na garantia de qualidade do software, e para serem relevantes, precisam ser adequadamente planejados e executados, encontrando a maior quantidade de erros possível.

Um grande desafio é controlar a qualidade de sistemas de software, sendo os principais motivos a alta complexidade dos produtos e as inúmeras dificuldades envolvidas no processo de desenvolvimento, que incluem questões humanas, técnicas, burocráticas e de negócio. (BERNARDO; KON, 2008).

> Na indústria brasileira de desenvolvimento de software para sistemas de informações, observa-se sensibilidade das empresas quanto à necessária melhoria contínua na qualidade do processo produtivo. Uma das formas de tratar a qualidade do desenvolvimento de software – na expectativa de alinhar os processos de negócio do cliente às rotinas computacionalmente programadas no sistema de informações – dá-se por meio do teste de software. O teste de software busca institucionalizar práticas de gestão de projetos e desenvolvimento de produtos de software, com o objetivo de localizar os problemas – e não garantir a sua inexistência (TOSETTO; BELLINI, 2008, p.326).

Os testes de software estão diretamente ligados à qualidade do sistema e a satisfação do usuário, porém são necessários um adequado planejamento e uma correta execução dos testes, para atingir o objetivo de encontrar o maior número possível de erros e corrigi-los. Durante a fase de definição de custo do projeto, uma parcela considerável dele, deve ser considerada para a fase de teste.

Segundo Veloso *et al* (2010), durante o processo de desenvolvimento de sistemas de informação, os testes são essenciais para garantir a qualidade dos resultados, contudo a atividade de testes demanda tempo e recursos, podendo chegar a ser responsável por 50% dos gastos do desenvolvimento. Assim as empresas buscam formas de otimizar essa atividade, sendo o uso de ferramentas de apoio a testes, a mais comumente utilizada.

De acordo com Myers (2004, p. 10),

> Ao testar um programa, você deseja adicionar algum valor a ele. Agregar valor por meio de testes significa aumentar a qualidade ou confiabilidade do programa. Aumentar a confiabilidade do programa significa encontrar e remover erros. Portanto, não teste um programa para mostrar que ele funciona; em vez disso, você deve começar com a suposição de que o programa contém erros (uma suposição válida para quase todos os programas) e, em seguida, testar o programa para encontrar o maior número possível de erros.[2]

Os testes são atividades executadas em um produto, sistema ou componente, sob condições específicas e com a avaliação de resultados. O objetivo do teste é um grupo identificado de características de software que, em condições específicas, precisa ser medido, comparando-se o comportamento encontrado com o comportamento desejado, tal como descrito na documentação do software (PAULA FILHO, 2015).

Segundo Paula Filho (2015), um teste é um conjunto de procedimentos e casos de testes. Sendo:

---

[2] *When you test a program, you want to add some value to it. Adding value through testing means raising the quality or reliability of the program. Raising the reliability of the program means finding and removing errors. Therefore, don't test a program to show that it works; rather, you should start with the assumption that the program contains errors (a valid assumption for almost any program) and then test the program to find as many of the errors as possible*.

- Procedimento de teste: é uma estrutura detalhada de instruções para a execução do teste.
- Caso de teste: é a definição das entradas, resultados pretendidos e dos cenários para execução. O objetivo de um bom caso de teste é detectar defeitos e não mostrar que o sistema funciona corretamente. Deve obrigatoriamente ter uma descrição das saídas desejadas, que serão utilizadas para avaliar os resultados das saídas reais, encontradas em cada execução. Os casos de teste devem comportar tanto as entradas válidas quanto as inválidas. Precisam ser relevantes para garantir uma cobertura adequada. É importante escrever casos de testes para execução de procedimentos errados, quando forem possíveis. É relevante que os casos de teste tenham uma ordem definida de execução, pois a execução de um caso de testes, muitas vezes depende de uma base de dados, que é alimentada pela execução correta de casos anteriores.

Segundo Delamaro, Maldonado e Jino (2016), é complexa a atividade de testes. A ocorrência de erros num software pode ser decorrente de diversos fatores, como algoritmos incorretos, não utilização de regras de segurança e muitos outros. Assim a atividade de teste deve ser dividida em fases, com objetivos diferentes. Estas fases podem ser:

- Teste de Unidade: foca nas menores unidades de um sistema, como funções, métodos e classes, buscando identificar erros relacionados a algoritmos mal implementados ou incorretos, erros na estrutura de dados ou na programação. As unidades são testadas separadamente, durante a implementação e pelo próprio desenvolvedor.
- Teste de Integração: é um teste posterior ao unitário e o foco é na construção da estrutura do sistema. À medida que as partes do software são desenvolvidas e integradas, é necessário verificar se estão funcionando corretamente e sem erros.
- Teste de Sistema: tem foco na verificação se as funcionalidades especificadas nos requisitos estão corretamente implementadas. Inclui também, testes de segurança, performance e robustez. Muitas empresas designam estes testes para equipes independentes em relação a equipe de desenvolvimento.

Há ainda os chamados teste de regressão, que são executados durante a manutenção do software. Quando uma modificação é feita no sistema, pode ocorrer a introdução de defeitos, assim é necessário testar o que foi modificado e verificar se está funcionando corretamente,

assim como testar as demais funcionalidades para verificar se continuam válidas. (DELAMARO; MALDONADO; JINO, 2016)

Paula Filho (2015), faz uma classificação dos principais tipos de testes, listados quanto a sua função nas diversas etapas do desenvolvimento e manutenção de um sistema. As funções não são excludentes, o mesmo teste pode desempenhar vários papéis. O Quadro 1, mostra esta classificação. Entre os testes classificados no quadro, está o Teste Funcional, que é o foco deste trabalho e que avalia a conformidade do sistema, com base nos requisitos funcionais.

Quadro 1 – Classificação dos testes quanto ao papel nos processos.

Termo	**Versão em inglês**	**Definição**
Teste de Aceitação	*Acceptance testing*	Teste formal realizado para determinar se um sistema satisfaz a seus critérios de aceitação e capacitar um usuário, cliente ou outra entidade autorizada a determinar se aceita ou não o sistema.
Teste de Desenvolvimento	*Development testing*	Teste formal ou informal realizado durante o desenvolvimento de um sistema ou componente, geralmente pelo desenvolvedor.
Teste de Integração	*Integration testing*	Teste no qual os componentes são combinados e avaliados para testar a integração entre eles.
Teste de Qualidade	*Qualification testing*	Teste realizado para determinar se um sistema ou componente é adequado para uso operacional.
Teste de Regressão	*Regression testing*	Teste seletivamente repetido de um sistema ou componente para verificar se alterações não causaram efeitos indesejáveis e se o sistema ou componente mantem a conformidade com os requisitos especificado.
Teste de Sistema	*System testing*	Testes conduzidos em um sistema completo integrado, para avaliar a sua conformidade com os requisitos especificados.
Teste de Unidade	*Unit testing*	Teste individual de unidades ou grupos de unidades.
Teste Funcional	*Functional testing*	Teste realizado para avaliar a conformidade de um sistema ou componente com os requisitos funcionais especificados.
Teste Operacional	*Operational testing*	Teste realizado para avaliar um sistema ou componente em seu ambiente operacional.

Fonte: Pádua Filho (2015), pág. 352

O software pode ser testado a partir de duas linhas diferentes: técnicas de projeto de caso de teste “caixa branca”, onde é testada a lógica interna do sistema e técnicas de casos de teste “caixa preta”, onde são testados os requisitos do software. Em qualquer dos casos, a intenção o objetivo é encontrar o máximo de erros com o menor esforço e tempo (PRESSMAN, 2011).

Segundo Machado (2016), os requisitos são o princípio para a completa definição do sistema, ou seja, são os pontos fundamentais para o desenvolvimento do produto. São de natureza variável e podem ser uma descrição de funcionalidade, uma especificação detalhada, uma restrição técnica ou de processo. Os Requisitos podem ser classificados em dois grupos:

- Requisitos Funcionais: especificam como o sistema interage com o contexto ao seu redor. É onde se encontram as maiores dificuldades e maiores índices de falhas nos projetos. São responsáveis por descrever o comportamento do software, ou seja, descrevem as funcionalidades, que se espera que o sistema forneça.
- Requisitos Não Funcionais: demonstram atributos de qualidade da solução, se preocupando com confiabilidade, desempenho, robustez, segurança, usabilidade, manutenibilidade, entre outros.

Sendo os testes de “caixa preta” também chamados de testes funcionais, segundo Paula Filho (2015), os testes funcionais são descritos, para checar a solidez entre o produto desenvolvido e os respectivos requisitos funcionais. As três principais técnicas para descrever os testes funcionais são:

- Partição de equivalência: divide as entradas em categorias de dados. Cada categoria mostra uma classe de defeitos, sendo possível que casos de teste da mesma categoria, sejam excluídos, sem comprometer a cobertura dos testes. Para cada entrada do software, são selecionados conjuntos de valores válidos e inválidos que definem as equivalências para cada entrada.
- Análise do valor limite: seleciona os casos de teste que avaliam os limites. Erros nos limites dos domínios são mais comuns que os das regiões centrais.
- Testes de comparação: há situações em que é preciso comparar as saídas de diversas versões de uma aplicação, quando sujeitas as mesmas entradas. Aplicam-se a situações como, uso de softwares redundantes, para sistemas complexos ou verificação de saídas de produtos em evolução.

2.1 TESTES AUTOMATIZADOS

Os testes funcionais podem ser executados manualmente ou de forma automatizados. Segundo Quintino; Motta e Vargas (2019), a automação de testes de software tem por objetivo agilizar o processo de testes, aumentando a assertividade e a produtividade, assim visando reduzir a atuação humana na realização dos testes manuais repetitivos, testando aplicações e funcionalidades inúmeras vezes, buscando encontrar erros que impactem nos requisitos especificados ou na qualidade do Software. Ainda segundo estes autores as vantagens são: reutilização das informações, ganhos de produtividade, volume, redução de custos, redução de erros.

O mercado disponibiliza várias ferramentas para automação de testes de Software, entre elas o Cypress, que é um framework de testes, de código aberto, de fácil configuração, com curva de aprendizado menor que outras ferramentas para o mesmo fim. O servidor e interpretador da Linguagem Java Script utilizado é o Node JS. Possui também um painel, onde é possível acompanhar a execução dos testes, auxiliando o profissional a visualizar quais partes precisam de ajuste. O Cypress possui uma documentação completa em cypress.io, com dicas e exemplos, o que auxilia em muito a escrita dos testes. (ATECH, 2021).

Através do levantamento bibliográfica foi possível concluir que as atividades de teste são essenciais para garantir a qualidade do sistema de informação que está em desenvolvimento. Como os testes consomem considerável tempo e recursos do projeto, é fundamental que as etapas de planejamento e execução dos testes sejam corretamente definidas. Além disso a automação de testes, é uma importante estratégia para ganhos de produtividades e redução de custos. O tester deve sempre supor que o software contém erros e atuar para encontrar o maior número deles, contribuindo assim o desenvolvimento de software de alta qualidade.

3 MATERIAIS E MÉTODOS

Após a definição dos objetivos, o estudo foi iniciado pela revisão bibliográfica sobre teste de software, com ênfase nos testes funcionais, para ampliação do repertório teórico sobre o tema. A fase seguinte, envolveu a aplicação LevantAí, com levantamento dos requisitos funcionais, escrita dos casos de teste, execução de testes manuais e automação dos casos de testes. A seguir os materiais e métodos são apresentados de forma mais detalhada.

3.1 MATERIAIS

No estudo foi realizada uma revisão bibliográfica sobre testes de software e como eles contribuem para aumentar a qualidade do sistema, com foco nos testes funcionais ou testes de caixa preta.

A revisão foi realizada através de levantamento e seleção de livros, artigos de revistas e sites sobre o tema, disponibilizados fisicamente ou através dos meios eletrônicos, em língua portuguesa ou inglesa.

O sistema LevantAí é uma aplicação Web que foi desenvolvida utilizando:

- React
- Node
- MySQL
- Git

Para a execução dos testes funcionais foram escritos os casos de testes no formato BDD[3] e foi disponibilizada uma versão executável da aplicação LevantAí.

Para os testes automatizados foram utilizados o editor Visual Studio Code e o Cypress, que é um framework que usa a linguagem de programação JavaScript.

[3] *BDD (Behavior Driven Development) é uma técnica de desenvolvimento ágil que associa as regras de negócio com linguagem de programação.*

3.2 MÉTODOS

Inicialmente foi realizada uma revisão bibliográfica, para aprofundamento sobre tema, com foco em testes funcionais e qualidade de software. Os testes funcionais foram executados na aplicação Web LevantAí, que foi desenvolvida em grupo na disciplina de Laboratório Engenharia de Software.

Os requisitos funcionais que foram utilizados para verificar a aderência da aplicação, estão listados no Quadro 2, abaixo.

Quadro 2 – Requisitos Funcionais da Aplicação LevantAí.

Código	Requisitos
RF01	O sistema deverá permitir que o usuário faça login (integração com Gmail).
RF02	O sistema deverá permitir que o usuário inicie seu tempo de foco (duração fixa: 25 minutos).
RF03	O sistema deverá permitir que o usuário abandone seu tempo de foco, reiniciando o tempo.
RF04	O sistema deverá permitir que o usuário inicie seu tempo de pausa após finalizar tempo de foco (duração fixa: 5 minutos).
RF05	O sistema deverá permitir que o usuário aborte a pausa, reiniciando o tempo de foco.
RF06	O sistema deverá exibir uma sugestão de desafio para o período de pausa.
RF07	O sistema deverá permitir que o usuário informe se o desafio do período de pausa foi realizado ou não (Botões “Completei” e “Falhei”).
RF08	O sistema deverá exibir uma barra com a experiência (”xp”) dos desafios completados (cada desafio possui um valor de “xp” de acordo com a dificuldade).
RF09	O sistema deverá exibir um relatório com os desafios completados/falhados e o tempo focado.
RF10	O sistema deverá permitir que o usuário faça logout, podendo trocar de conta.
RF11	O sistema deverá voltar a tela do tempo de foco quando finalizar o tempo de pausa.
RF12	O sistema deverá sempre redirecionar para tela Login, caso não o tenha feito e tente acessar o site diretamente.
RF13	O sistema deverá emitir um efeito sonoro para sinalizar final do tempo de foco.

RF14	O sistema exibirá para Administradores relatórios de “Total de usuários”, “Usuários online”, “Rank Máximo alcançado” e um gráfico do “Total de tempo focado por dia”.

Fonte: Autoria Própria (2022).

Os casos de teste foram escritos em conformidade com os requisitos funcionais, com o objetivo de fazer a maior cobertura possível das funcionalidades, com o objetivo de encontrar o maior número de bugs.

A escrita dos casos de testes foi baseada no conceito do BDD (Behavior Driven Development), que segundo Rodrigues (2017), é uma técnica de desenvolvimento ágil que associa as regras de negócio com linguagem de programação, enfatizando o teste do software e com o objetivo de garantir que a funcionalidade esteja de acordo com o esperado.

Como a entrega das funcionalidades é incremental, os casos de teste das funcionalidades ainda não entregues (Implantado: Não) estão com status “Adiado”. Para fins didáticos os casos de testes foram colocados em tabelas e agrupados a cada quatro Requisitos Funcionais.

Quadro 3 – Casos de Teste e Registro dos Testes da Aplicação LevantAí (RF01 e RF04).

Nº Teste	Caso de Teste	Tipo	Testado por	Data	Status
T01	**Requisito: RF01** **Cenário:** Logar no Sistema. **Dado que**: Estou no navegador Chrome. **E:** Entro na página de Login da aplicação LevantAí. **Quando:** Seleciono o botão “Login com Google” com e-mail válido. **Então:** Acesso a página inicial da aplicação. **Implantado:** Não	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T02	**Requisito: RF02** **Cenário:** Iniciar ciclo. **Dado que**: Estou na página inicial. **E:** Visualizo o botão “Iniciar um ciclo”. **Quando:** Clico no botão “Iniciar um ciclo”. **Então:** Inicia o tempo do ciclo de 25 minutos. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T03	**Requisito: RF03** **Cenário:** Abandonar ciclo. **Dado que**: Estou com um ciclo em andamento. **E:** Resolvo abandonar este ciclo. **Quando:** Clico no botão “Abandonar”. **Então:** O ciclo se encerra, retornando ao início. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T04	**Requisito: RF04** **Cenário:** Iniciar pausa. **Dado que**: Finalizei o tempo do ciclo. **E:** Exibiu o botão “Iniciar” para a Pausa. **Quando:** Clico no botão “Iniciar” **Então:** Inicia o tempo da pausa de 5 minutos.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou

	Implantado: Sim.				

Fonte: Autoria Própria (2022).

Quadro 4 – Casos de Teste e Registro dos Testes da Aplicação LevantAí (RF05 e RF08).

Nº Teste	Caso de Teste	Tipo	Testado por	Data	Status
T05	**Requisito: RF05** **Cenário:** Abortar pausa. **Dado que**: Estou com a pausa em andamento. **E:** Resolvo abortar o ciclo. **Quando:** Clico no botão "Abortar". **Então:** A pausa se encerra, retornando ao início do ciclo. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T06	**Requisito: RF06** **Cenário:** Exibir desafio da pausa. **Dado que**: Finalizei o tempo do ciclo. **E:** Cliquei no botão "Iniciar" da pausa. **Quando:** Começa a correr o tempo de pausa. **Então:** O sistema exibe um desafio aleatório. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T07	**Requisito: RF07** **Cenário:** Completar desafio da pausa. **Dado que**: Iniciei o desafio da pausa. **E:** Realizei o desafio. **Quando:** Clico no botão "Completei" do desafio. **Então:** É exibido um modal com "Parabéns". **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T08	**Requisito: RF07** **Cenário:** Falhar desafio da pausa. **Dado que**: Iniciei o desafio da pausa. **E:** Não finalizei o desafio. **Quando:** Clico no botão "Falhei" do desafio. **Então:** O sistema retorna para a tela de Foco. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T09	**Requisito: RF08** **Cenário:** Exibir "xp" dos desafios. **Dado que**: Estou na página inicial do LevantAi. **E:** Completo desafios dos ciclos. **Quando:** Visualizo a barra de "xp". **Então:** O sistema exibirá a sua experiência (xp) de acordo com os desafios completados (xp diferente por atividade). **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T10	**Requisito: RF08** **Cenário:** Verificar a soma dos "xp" após a execução da atividade. **Dado que**: Finalizei o tempo de pausa. **E:** Exibe o Botão "Completei". **Quando:** Cliquei no botão "Completei". **Então:** O sistema deverá somar a quantidade de experiências (xp) conquistadas, na barra de experiências. **Implantado:** Sim	Manual Manual Manual	Cecilia Anderson Cecilia	18/05/22 19/05/22 02/06/22	Falhou Passou Passou

T11	**Requisito: RF08** **Cenário:** Verificar se ao atingir a quantidade de experiências necessárias, avança de nível. **Dado que**: Finalizei o tempo de pausa. **E:** Exibe o Botão "Completei". **Quando:** Cliquei no botão "Completei". **Então:** Ao atingir a quantidade de experiências, o sistema deve avançar de nível. **Implantado:** Sim	Manual Manual Manual	Cecilia Anderson Cecilia	18/05/22 19/05/22 02/06/22	Falhou Passou Passou
T12	**Requisito: RF08** **Cenário:** Verificar a exibição do Modal. **Dado que**: Finalizei o tempo de pausa. **E:** Exibe o Botão "Completei". **Quando:** Cliquei no botão "Completei". **Então:** O sistema deve exibir o Modal com Parabéns pela Mudança de nível. **Implantado:** Sim	Manual Manual Manual	Cecilia Anderson Cecilia	18/05/22 19/05/22 02/06/22	Passou Passou Passou
T13	**Requisito: RF08** **Cenário:** Verificar se o "Nível" fica salvo **Dado que**: Finalizei o uso do LevantAí hoje. **E:** Tenho um Nível conquistado. **Quando:** Fecho a aplicação e saio do navegador. **Então:** Ao retornar a aplicação (no mesmo computador e no mesmo navegador) o sistema deve ter salvo o nível. **Implantado:** Sim	Manual Manual Manual	Cecilia Anderson Cecilia	18/05/22 19/05/22 02/06/22	Falhou Passou Passou

Fonte: Autoria Própria (2022).

Quadro 5 – Casos de Teste e Registro dos Testes da Aplicação LevantAí (RF09 a RF12).

Nº Teste	**Caso de Teste**	**Tipo**	**Testado por**	**Data**	**Status**
T14	**Requisito: RF09** **Cenário:** Exibir relatório do usuário. **Dado que**: Estou na página inicial do LevantAi. **E:** Já utilizei o site em outros momentos. **Quando:** Clico no menu e em Relatório. **Então:** O sistema exibirá os desafios completados/ falhados e o tempo focado. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T15	**Requisito: RF09** **Cenário:** Validar "Tempo Focado" no histórico. **Dado que**: Estou na tela inicial da aplicação. **E:** Completo um ciclo. **Quando:** Entro na tela de histórico. **Então:** Visualizo esse tempo acrescentado em "Tempo Focado" **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T16	**Requisito: RF09** **Cenário:** Validar "Desafios Completados" no histórico. **Dado que**: Estou na tela inicial do site. **E:** Completei um desafio. **Quando:** Entro na tela de histórico. **Então:** Visualizo esse desafio acrescentado em "Desafios Completados". **Implantado:** Sim.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T17	**Requisito: RF09** **Cenário:** Validar "Desafios falhados" no histórico. **Dado que**: Estou na tela inicial do site. **E:** Falhei um desafio.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado

	Quando: Entro na tela de histórico. **Então:** Visualizo esse desafio acrescentado em "Desafios Falhados". **Implantado:** Não.				
T18	**Requisito: RF10** **Cenário:** Sair da conta logada. **Dado que**: Estou na página inicial do LevantAi. **E:** Quero sair da conta logada. **Quando:** Clico no ícone de sair. **Então:** O sistema voltará para a página de Login. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T19	**Requisito: RF11** **Cenário:** Retornar a tela de foco. **Dado que**: Finalizei o tempo de pausa **E:** Quero iniciar novo ciclo. **Quando:** Clico no botão "Completei" ou "Falhei". **Então:** O sistema exibe "Iniciar um ciclo" novamente. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T20	**Requisito: RF12** **Cenário:** Redirecionar para Tela de login, quando não logado. **Dado que**: Estou na tela inicial. **E:** Não faço login. **Quando:** Tento acessar diretamente a URL da página inicial ou do histórico. **Então:** O site redireciona para a tela de Login. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado

Fonte: Autoria Própria (2022).

Quadro 6 – Casos de Teste e Registro dos Testes da Aplicação LevantAí (RF13 e RF14).

Nº Teste	**Caso de Teste**	**Tipo**	**Testado por**	**Data**	**Status**
T21	**Requisito: RF13** **Cenário:** Validar efeito sonoro no final do Foco. **Dado que**: Estou na tela de Foco. **E:** Iniciei o tempo de foco. **Quando:** O tempo acaba. **Então:** O sistema irá emitir um efeito sonoro para sinalizar que o tempo de Foco acabou. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T22	**Requisito: RF13** **Cenário:** Validar a notificação no final do Foco. **Dado que**: Iniciei o tempo de foco **E:** Permito as notificações do navegador. **Quando:** O tempo acaba. **Então:** O sistema irá exibir uma notificação do navegador. **Implantado:** Sim.	Manual Manual Manual	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Passou Passou Passou
T23	**Requisito: RF14** **Cenário:** Validar Campo "Total de Usuários" no relatório do Adm. **Dado que**: Estou na página inicial. **E:** Acesso o sistema com um novo usuário. **Quando:** Acesso o relatório de Adm **Então:** Verifico acrescentar em "Total de usuários" o novo usuário. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado

T24	**Requisito: RF14** **Cenário:** Validar campo “Usuários Online” no relatório do Adm, quando online. **Dado que**: Acesso o sistema com usuário. **E:** Permaneço online. **Quando:** Acesso o relatório de Adm. **Então:** Verifico acrescentar em “Usuários Online” esse usuário, enquanto permanecer no site. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T25	**Requisito: RF14** **Cenário:** Validar campo “Usuários Online” no relatório do Adm, quando sai do sistema. **Dado que**: Há um usuário conectado no site. **E:** Visualizo adicionar no campo “Usuário Online”. **Quando:** Este usuário se desconecta. **Então:** Ao atualizar o relatório de Adm, deve subtraí-lo do campo “Usuários Online”. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T26	**Requisito: RF14** **Cenário:** Validar Campo “Rank Máx Alcançado” no relatório do Adm. **Dado que**: Há um usuário com um nível de Rank maior que 0. **E:** Outro usuário com um Rank maior que o usuário anterior. **Quando:** Acesso o relatório de Adm. **Então:** O campo “Rank Máx Alcançado” exibirá o maior Rank dos usuários. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado
T27	**Requisito: RF14** **Cenário:** Validar gráfico “Total de Tempo Focado por dia” no relatório do Adm. **Dado que**: Acesso site com um usuário. **E:** Realizo um tempo de foco. **Quando:** Acesso o relatório de Adm. **Então:** Visualizo o valor e data corretos desse foco acrescentados no gráfico “Total de Tempo Focado por dia”. **Implantado:** Não.	N/A N/A N/A	Alexandre Anderson Cecilia	12/05/22 19/05/22 02/06/22	Adiado. Adiado Adiado

Fonte: Autoria Própria (2022).

3.2.1 TESTES AUTOMATIZADOS

A automação dos testes foi realizada no editor Visual Studio Code e o framework usado foi o Cypress. As funcionalidades entregues da Aplicação LevantAí, permitiram a execução de 15 testes, destes foi possível automatizar 11, pois o resultado da execução destes testes puderam ser previstos e identificados na tela. Porém 4 testes não puderam ser automatizados, foram eles: os testes T10 e T11 que são relacionados a quantidade de “xps” (pontos) dos desafios de pausa e como os desafios têm quantidades de xps diferentes e são apresentados de forma aleatório, não foi possível prever o resultado; teste T21 é relacionado ao sinal sonoro e o T22 a uma notificação que não aparece na execução do Cypress.

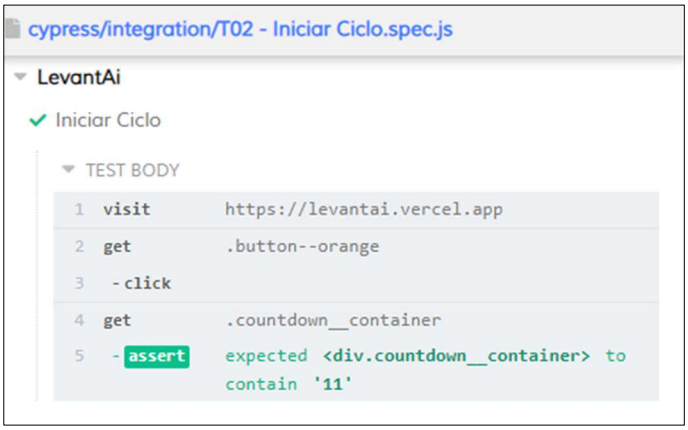
Os testes automatizados serão executados novamente e os resultados inseridos na tabela de registro dos testes, quando houver entrega de novas funcionalidades, a fim de verificar rapidamente e com pouca intervenção humana, se os códigos incrementados, quebraram algum código que estava funcionado corretamente, conforme já referido por Quintino; Motta e Vargas (2019),

As Tabelas de 1a até 11b a seguir, mostram os códigos (tabelas a) e a execução (tabelas b) dos 11 testes que foram automatizados. Nas tabelas "b" que mostram a execução dos testes pelo Cypress, é possível identificar que o teste passou através do retorno "assert" em verde.

Tabela 1a – Código Cypress: T02 – Iniciar Ciclo

Fonte: Autoria Própria (2022).

Tabela 1b – Execução Cypress: T02 – Iniciar Ciclo



Fonte: Autoria Própria (2022).

Tabela 2a – Código Cypress: T03 – Abortar Ciclo

T03 - Abortar Ciclo.spec.js

cypress > integration > T03 - Abortar Ciclo.spec.js > ...

```
/// <reference types="cypress" />

describe('LevantAi', () => {
    it('Abortar Ciclo', () => {
        cy.visit('https://levantai.vercel.app')
        cy.get('.button--orange').click()
        cy.get('.button--red').click()
        cy.get('.button--orange').should('contain','Iniciar um ciclo')
    })
})
```

Fonte: Autoria Própria (2022).

Tabela 2b – Execução Cypress: T03 – Abortar Ciclo

Fonte: Autoria Própria (2022).

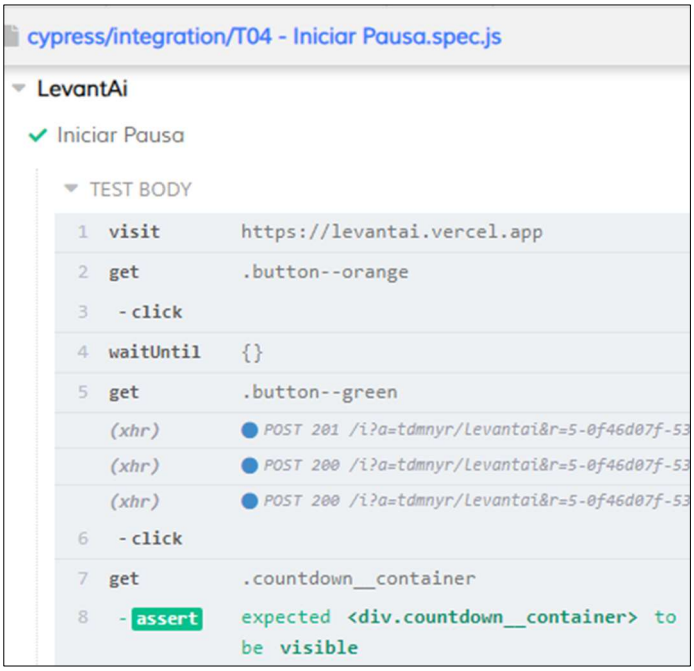
Tabela 3a – Código Cypress: T04 – Iniciar Pausa

```
/// <reference types="cypress" />

describe('LevantAi', () => {
    it('Iniciar Pausa', () => {
        cy.visit('https://levantai.vercel.app')
        cy.get('.button--orange').click()
        cy.waitUntil(() =>
            cy.get('.button--green', {timeout: 13000})
                .click()
            )
        cy.get('.countdown__container').should('be.visible','5')
    })
})
```

Fonte: Autoria Própria (2022).

Tabela 3b – Execução Cypress: T04 – Iniciar Pausa



cypress/integration/T04 - Iniciar Pausa.spec.js

LevantAi

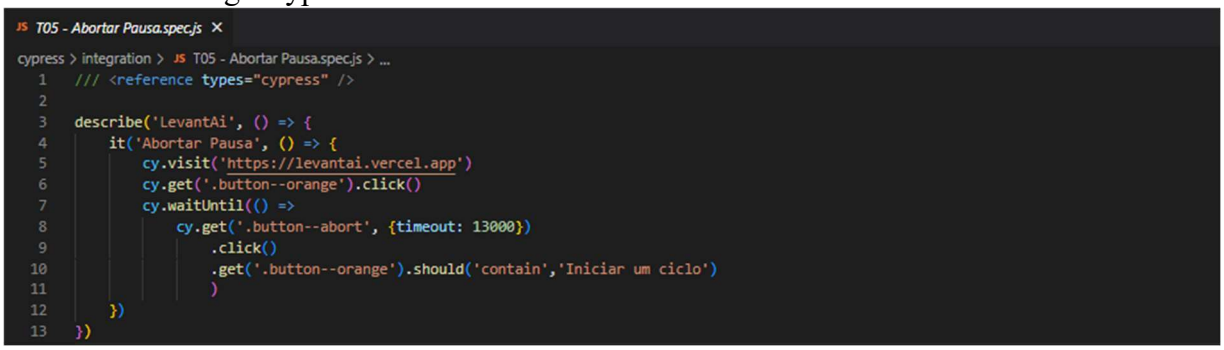
Iniciar Pausa

TEST BODY

1	visit	https://levantai.vercel.app
2	get	.button--orange
3	- click	
4	waitUntil	{}
5	get	.button--green
	(xhr)	POST 201 /i?a=tdmnyr/levantai&r=5-0f46d07f-53
	(xhr)	POST 200 /i?a=tdmnyr/levantai&r=5-0f46d07f-53
	(xhr)	POST 200 /i?a=tdmnyr/levantai&r=5-0f46d07f-53
6	- click	
7	get	.countdown__container
8	- assert	expected <div.countdown__container> to be visible

Fonte: Autoria Própria (2022).

Tabela 4a – Código Cypress: T05 – Abortar Pausa

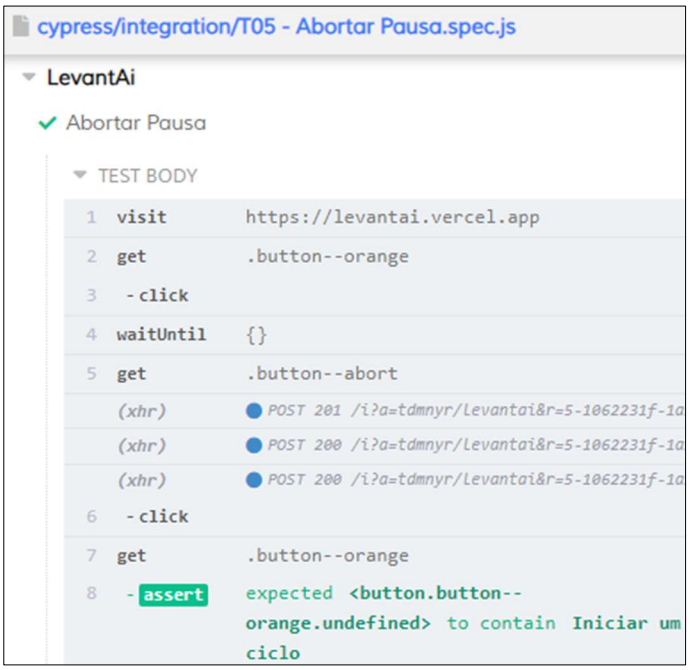


```
/// <reference types="cypress" />

describe('LevantAi', () => {
    it('Abortar Pausa', () => {
        cy.visit('https://levantai.vercel.app')
        cy.get('.button--orange').click()
        cy.waitUntil(() =>
            cy.get('.button--abort', {timeout: 13000})
                .click()
                .get('.button--orange').should('contain','Iniciar um ciclo')
                )
    })
})
```

Fonte: Autoria Própria (2022).

Tabela 4b – Execução Cypress: T05 – Abortar Pausa



```
cypress/integration/T05 - Abortar Pausa.spec.js
LevantAi
Abortar Pausa
TEST BODY
1  visit      https://levantai.vercel.app
2  get        .button--orange
3   -click
4  waitUntil  {}
5  get        .button--abort
   (xhr)      POST 201 /i?a=tdmnyr/levantai&r=5-1062231f-1a
   (xhr)      POST 200 /i?a=tdmnyr/levantai&r=5-1062231f-1a
   (xhr)      POST 200 /i?a=tdmnyr/levantai&r=5-1062231f-1a
6   -click
7  get        .button--orange
8   -assert   expected <button.button--
              orange.undefined> to contain Iniciar um
              ciclo
```

Fonte: Autoria Própria (2022).

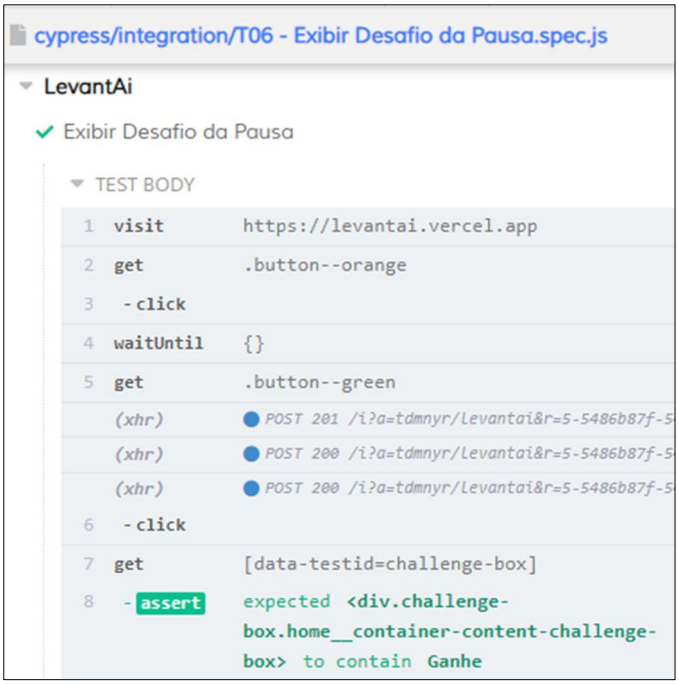
Tabela 5a – Código Cypress: T06 – Exibir Desafio de Pausa

```
/// <reference types="cypress" />

describe('LevantAi', () => {
    it('Exibir Desafio da Pausa', () => {
        cy.visit('https://levantai.vercel.app')
        cy.get('.button--orange').click()
        cy.waitUntil(() =>
            cy.get('.button--green', {timeout: 13000})
                .click()
            )
        cy.get('[data-testid=challenge-box]').should('contain','Ganhe')
    })
})
```

Fonte: Autoria Própria (2022).

Tabela 5b – Execução Cypress: T06 – Exibir Desafio de Pausa



Fonte: Autoria Própria (2022).

Tabela 6a – Código Cypress: T07 – Completar Desafio da Pausa

T07 - Completar Desafio da Pausa.spec.js

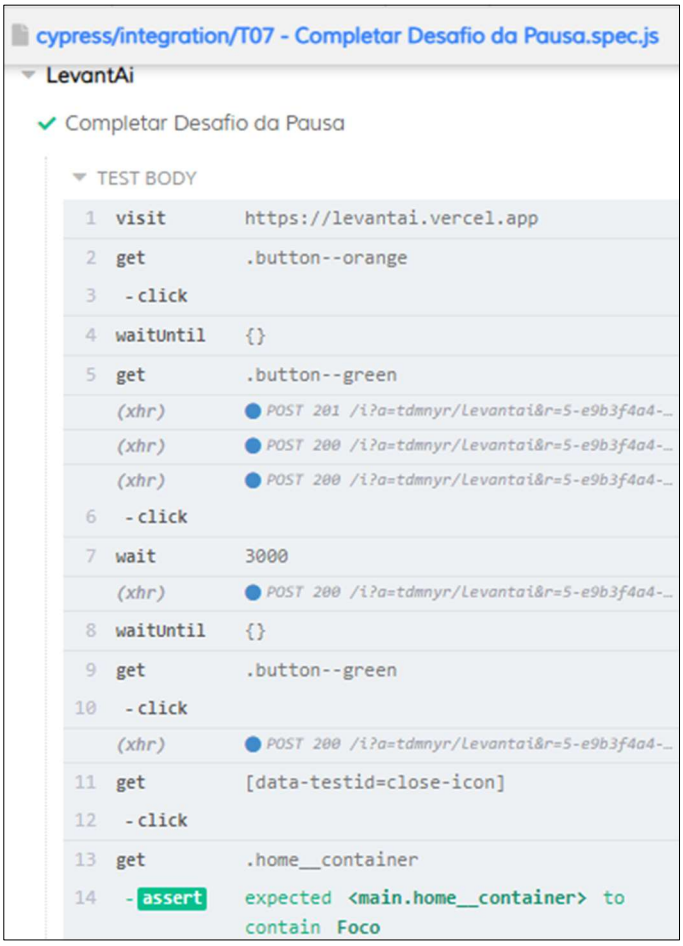
cypress > integration > T07 - Completar Desafio da Pausa.spec.js > ...

```
/// <reference types="cypress" />

describe('LevantAi', () => {
    it('Completar Desafio da Pausa', () => {
        cy.visit('https://levantai.vercel.app')
        cy.get('.button--orange').click()
        cy.waitUntil(() =>
            cy.get('.button--green', {timeout: 13000})
              .click()
            )
        cy.wait(3000)
        cy.waitUntil(() =>
            cy.get('.button--green', {timeout: 7000})
              .click()
            )
        cy.get('[data-testid=close-icon]').click()
        cy.get('.home__container').should('contain','Foco')
    })
})
```

Fonte: Autoria Própria (2022).

Tabela 6b – Execução Cypress: T07 – Completar Desafio da Pausa



Fonte: Autoria Própria (2022).

Tabela 7a – Código Cypress: T08 – Falhar Desafio de Pausa

T08 - Falhar Desafio de Pausa.spec.js

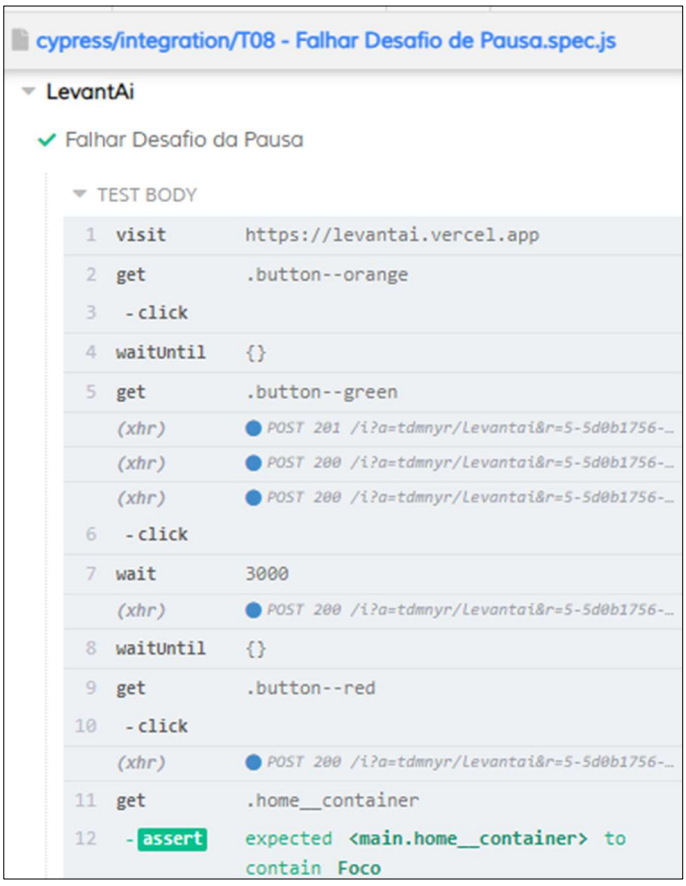
cypress > integration > T08 - Falhar Desafio de Pausa.spec.js > ...

```
/// <reference types="cypress" />

describe('LevantAi', () => {
    it('Falhar Desafio da Pausa', () => {
        cy.visit('https://levantai.vercel.app')
        cy.get('.button--orange').click()
        cy.waitUntil(() =>
            cy.get('.button--green', {timeout: 13000})
                .click()
            )
        cy.wait(3000)
        cy.waitUntil(() =>
            cy.get('.button--red', {timeout: 7000})
              .click()
            )
        cy.get('.home__container').should('contain','Foco')
    })
})
```

Fonte: Autoria Própria (2022).

Tabela 7b – Execução Cypress: T08 – Falhar Desafio de Pausa



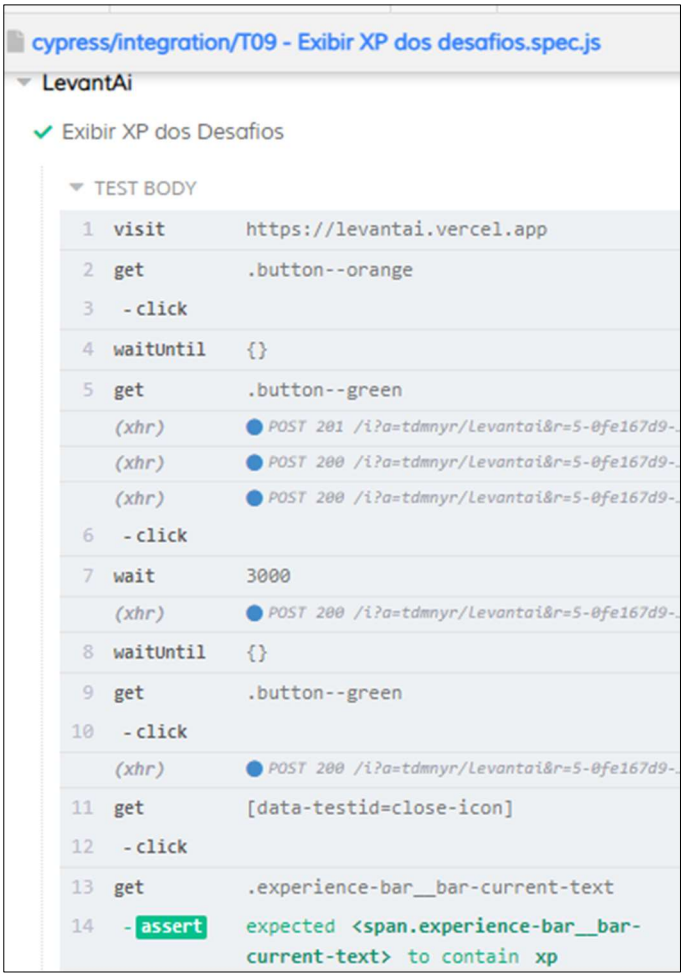
Fonte: Autoria Própria (2022).

Tabela 8a – Código Cypress: T09 – Exibir xp dos desafios

```
T09 - Exibir XP dos desafios.spec.js
cypress > integration > T09 - Exibir XP dos desafios.spec.js > ...
1   /// <reference types="cypress" />
2
3   describe('LevantAi', () => {
4       it('Exibir XP dos Desafios', () => {
5           cy.visit('https://levantai.vercel.app')
6           cy.get('.button--orange').click()
7           cy.waitUntil(() =>
8               cy.get('.button--green', {timeout: 13000})
9                   .click()
10                  )
11          cy.wait(3000)
12          cy.waitUntil(() =>
13              cy.get('.button--green', {timeout: 7000})
14                  .click()
15                  )
16          cy.get('[data-testid=close-icon]').click()
17          cy.get('.experience-bar__bar-current-text').should('contain', 'xp')
18      })
19  })
```

Fonte: Autoria Própria (2022).

Tabela 8b – Execução Cypress: T09 – Exibir xp dos desafios



```
cypress/integration/T09 - Exibir XP dos desafios.spec.js
LevantAi
✓ Exibir XP dos Desafios
TEST BODY
1   visit       https://levantai.vercel.app
2   get         .button--orange
3    -click
4   waitUntil   {}
5   get         .button--green
    (xhr)       POST 201 /i?a=tdmnyr/levantai&r=5-0fe167d9-
    (xhr)       POST 200 /i?a=tdmnyr/levantai&r=5-0fe167d9-
    (xhr)       POST 200 /i?a=tdmnyr/levantai&r=5-0fe167d9-
6    -click
7   wait        3000
    (xhr)       POST 200 /i?a=tdmnyr/levantai&r=5-0fe167d9-
8   waitUntil   {}
9   get         .button--green
10   -click
    (xhr)       POST 200 /i?a=tdmnyr/levantai&r=5-0fe167d9-
11  get         [data-testid=close-icon]
12   -click
13  get         .experience-bar__bar-current-text
14  -assert     expected <span.experience-bar__bar-
                current-text> to contain xp
```

Fonte: Autoria Própria (2022).

Tabela 9a – Código Cypress: T12 – Verificar a Exibição

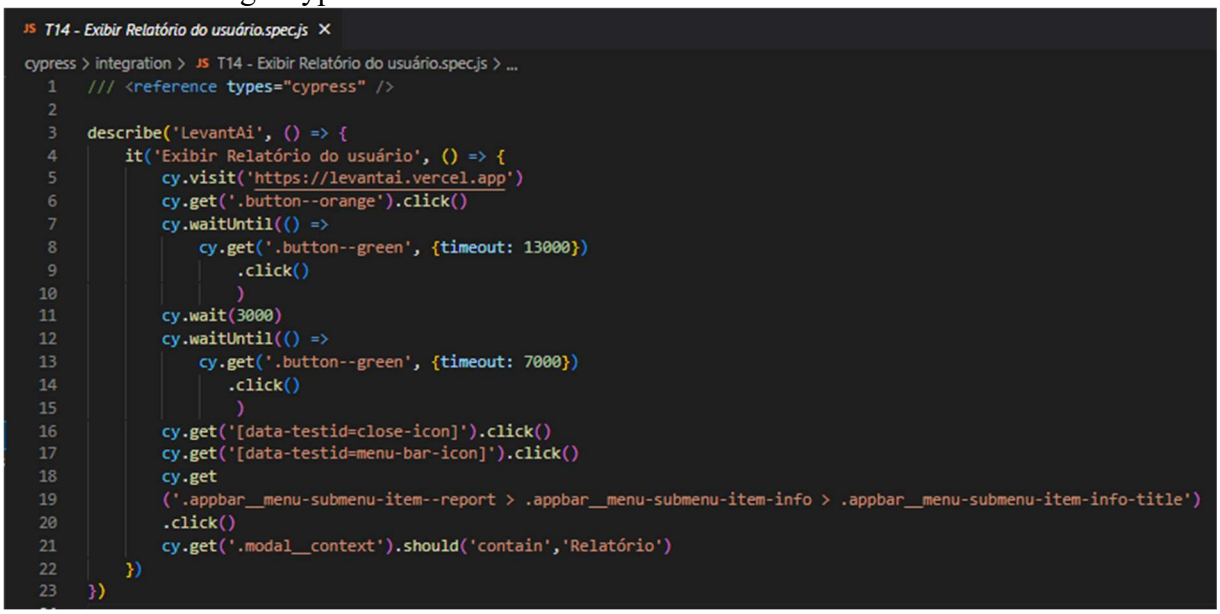
```
JS T12 - Verificar a Exibição do Modal.spec.js ×
cypress > integration > JS T12 - Verificar a Exibição do Modal.spec.js > ...
1   /// <reference types="cypress" />
2
3   describe('LevantAi', () => {
4       it('Completar Desafio da Pausa', () => {
5           cy.visit('https://levantai.vercel.app')
6           cy.get('.button--orange').click()
7           cy.waitUntil(() =>
8               cy.get('.button--green', {timeout: 13000})
9                   .click()
10                  )
11          cy.wait(3000)
12          cy.waitUntil(() =>
13              cy.get('.button--green', {timeout: 7000})
14                  .click()
15                  )
16          cy.get('.level-up-modal__container-title').should('contain', 'Parabéns')
17      })
18  })
```

Fonte: Autoria Própria (2022).

Tabela 9b – Execução Cypress: T12 – Verificar a Exibição

Fonte: Autoria Própria (2022).

Tabela 10a – Código Cypress: T14 – Exibir Relatório do usuário

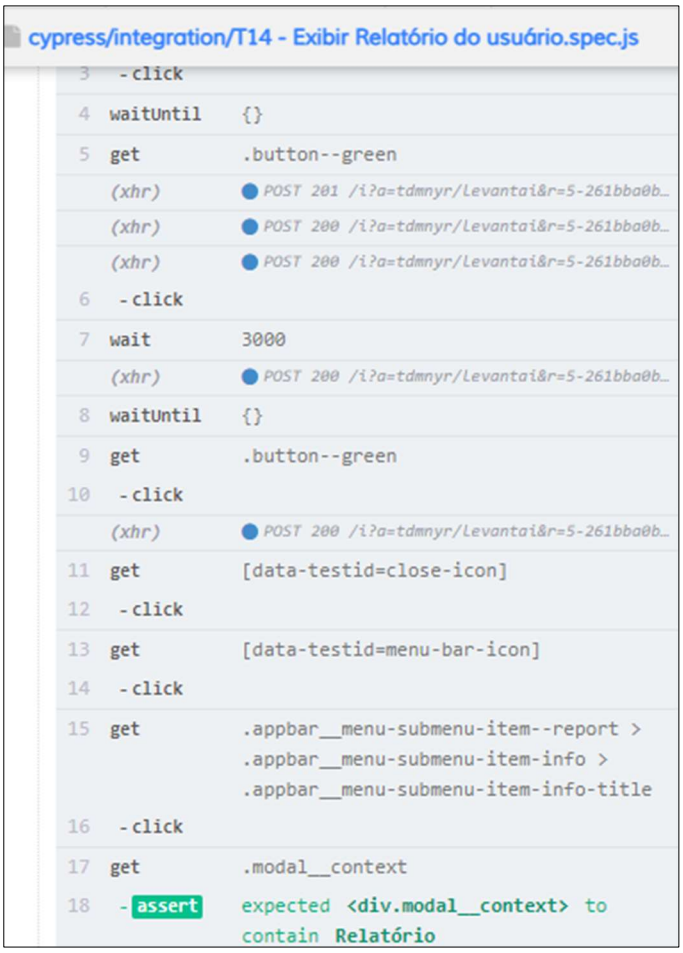


```
/// <reference types="cypress" />

describe('LevantAi', () => {
    it('Exibir Relatório do usuário', () => {
        cy.visit('https://levantai.vercel.app')
        cy.get('.button--orange').click()
        cy.waitUntil(() =>
            cy.get('.button--green', {timeout: 13000})
              .click()
            )
        cy.wait(3000)
        cy.waitUntil(() =>
            cy.get('.button--green', {timeout: 7000})
              .click()
            )
        cy.get('[data-testid=close-icon]').click()
        cy.get('[data-testid=menu-bar-icon]').click()
        cy.get
        ('.appbar__menu-submenu-item--report > .appbar__menu-submenu-item-info > .appbar__menu-submenu-item-info-title')
        .click()
        cy.get('.modal__context').should('contain','Relatório')
    })
})
```

Fonte: Autoria Própria (2022).

Tabela 10b – Execução Cypress: T14 – Exibir Relatório do usuário



cypress/integration/T14 - Exibir Relatório do usuário.spec.js

3	-click	
4	waitUntil	{}
5	get	.button--green
	(xhr)	POST 201 /i?a=tdmnyr/Levantai&r=5-261bba0b…
	(xhr)	POST 200 /i?a=tdmnyr/Levantai&r=5-261bba0b…
	(xhr)	POST 200 /i?a=tdmnyr/Levantai&r=5-261bba0b…
6	-click	
7	wait	3000
	(xhr)	POST 200 /i?a=tdmnyr/Levantai&r=5-261bba0b…
8	waitUntil	{}
9	get	.button--green
10	-click	
	(xhr)	POST 200 /i?a=tdmnyr/Levantai&r=5-261bba0b…
11	get	[data-testid=close-icon]
12	-click	
13	get	[data-testid=menu-bar-icon]
14	-click	
15	get	.appbar__menu-submenu-item--report > .appbar__menu-submenu-item-info > .appbar__menu-submenu-item-info-title
16	-click	
17	get	.modal__context
18	-assert	expected <div.modal__context> to contain Relatório

Fonte: Autoria Própria (2022).

Tabela 11a – Código Cypress: T19 – Retornar a tela de Foco

Fonte: Autoria Própria (2022).

Tabela 11b – Execução Cypress: T19 – Retornar a tela de Foco

Fonte: Autoria Própria (2022).

Os resultados dos testes funcionais foram adequadamente registrados, nos quadros 3 a 6. Na primeira rodada de testes, ocorrida entre os dias 12 e 18/05/22 (teste Alpha), foram executados 15 testes e deste 3 falharam. Os erros foram corrigidos e no dia 19/05/22 (teste Beta) foram executados novamente os testes e nenhum bug foi encontrado, assim como no dia 02/06/22, quando foi realizado o teste de Aceitação, das funcionalidades desenvolvidas.

4 RESULTADOS E DISCUSSÃO

Conforme o que o trabalho estudou, o objetivo do teste é demonstrar que o sistema faz o que foi proposto a fazer e também para descobrir defeitos do programa, que devem ser corrigidos, antes da entrega ao cliente (SOMMERVILLE, 2011). Segundo Myers (2004), quando um sistema é testado, o intuito é adicionar valor a ele, aumentando a qualidade e a confiabilidade do programa, através da localização e remoção de erros.

Para Bartiê (2002), uma forma de definir a qualidade de um software é através do número de requisitos que foram efetivamente testados e em conformidade com o que foi especificado. Assim, seguindo a afirmação do autor, foram criados casos de testes para todos os requisitos funcionais da Aplicação LevantAí.

Além dos testes manuais, foram desenvolvidos os testes automatizados, com o objetivo de verificar se há vantagens em automatizar testes em aplicações de baixa complexidade. Foi verificado a automação é uma forma de ganhar tempo nos testes, quando o sistema LevantAí tiver novas funcionalidades adicionadas ou quando precisar de manutenção, corroborando com a afirmação de Delamaro; Maldonado e Jino (2016), de que quando uma modificação é feita no sistema, pode ocorrer a introdução de defeitos, assim é necessário testar o que foi modificado e verificar se está funcionando corretamente, assim como testar as demais funcionalidades para verificar se continuam válidas.

Os testes automatizados diminuem a intervenção humana, na execução dos testes, tendo as vantagens apresentados por Quintino; Motta e Vargas (2019), que referem que a automação de testes de software agiliza o processo de testes, aumentando a assertividade e a produtividade, assim visando reduzir a atuação humana na realização dos testes manuais repetitivos, testando aplicações e funcionalidades inúmeras vezes, buscando encontrar erros que impactem nos requisitos especificados ou na qualidade do Software. Ainda segundo estes autores as vantagens são: reutilização das informações, ganhos de produtividade, volume, redução de custos, redução de erros. E isso independe da complexidade do software. Ainda sobre automação dos testes temos Veloso *et al* (2010), que afirma que a atividade de testes demanda tempo e recursos, podendo chegar a ser responsável por 50% dos gastos do desenvolvimento, assim as empresas buscam formas de otimizar essa atividade.

Os resultados dos testes funcionais na Aplicação LevantAí foram adequadamente registrados, nos quadros 3 a 6. Na primeira rodada de testes, ocorrida entre os dias 12 e 18/05/22 (teste Alpha), foram executados 15 testes e deste 3 falharam. Assim temos que foram

encontrados 20% de erros na entrega testada. Os erros foram corrigidos e no dia 19/05/22 (teste Beta) foram executados novamente os testes e nenhum Bug foi encontrado, assim como no dia 02/06/22, quando foi realizado o teste de Aceitação, das funcionalidades desenvolvidas. corroborando com a afirmação de Machado (2016), de que os requisitos funcionais são onde se encontram as maiores dificuldades e maiores índices de falhas no desenvolvimento dos projetos. Ao final foi verificado que a qualidade da aplicação LevantAí foi melhorada após a aplicação dos testes funcionais, mas é importante salientar, conforme referido por Tosetto e Bellini (2008), que os testes têm a função de encontrar problemas, mas não garantem sua inexistência.

5 CONSIDERAÇÕES FINAIS

Em conformidade ao que o trabalho que foi desenvolvido, Machado (2016) refere que os requisitos funcionais são onde encontram-se as maiores dificuldades e maiores índices de falhas no desenvolvimento dos projetos. Portanto os testes funcionais ou de “Caixa Preta” tem grande impacto na qualidade final de um software, sendo de grande relevância que estes testes sejam elaborados e executados em conformidade com os requisitos funcionais. A execução prática dos estudos do trabalho, foram realizados na Aplicação Web LevantAí, com a criação dos casos de testes, execução, identificação e resolução de bugs, além do desenvolvimento de testes automatizados, para agilizar os testes futuros da aplicação, seja por incorporação de novas funcionalidades ou necessidade de manutenção do sistema. Os testes aplicados, encontraram bugs, que foram corrigidos, melhorando a qualidade da Aplicação LevantAí. Como Testes de Software é um assunto muito amplo, trabalhos futuros podem abordar outros tipos de testes e como eles impactam na qualidade do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

BARTIÊ, Alexandre. **Garantia da Qualidade de Software**. 1 ed. Rio de Janeiro: Campos, 2002.

BERNARDO, P. C; KON, F. A importância dos testes automatizados. **Engenharia de Software Magazine**, v. 1, n. 3, p. 54-57, 2008. Disponível em: < https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf >. Acesso em: 15 nov. 2022.

Cypress: o novo conceito em testes automatizados. Disponível em: < https://atech.com.br/cypress-o-novo-conceito-em-testes-automatizados/ >. Acesso em: 12 out. 2022.

DELAMARO, M. E; MALDONADO, J.C; JINO, M. **Introdução ao Teste de Software**. 2 ed. Rio de Janeiro: Elsevier, 2016.

MACHADO, Felipe Nery R. **Análise e Gestão de Requisitos de Software** - onde nascem os sistemas. 3 ed. São Paulo: Érica, 2016.

Método Pomodoro de estudo: saiba como essa técnica ajuda estudantes a criar rotina de estudos. Disponível em:< https://conexao.pucminas.br/blog/dicas/metodo-pomodoro-de-estudo/ >. Acesso em: 04 fev. 2023

MYERS, Glenford J. **The Art of Software Testing**. 2 ed. New Jersey: John Wiley &Sons, 2004. Disponível em: < http://www.51testing.com/N_download/lib/TestingTechDL/ArtofSoftwareTesting.pdf >. Acesso em: 06 jun. 2022.

PAULA FILHO,Wilson de Pádua. **Engenharia de Software** – Fundamentos, Métodos e Padrões. 3 ed. Rio de Janeiro: LTC, 2015.

PIZZANI, L. et al. A Arte da Pesquisa Bibliográfica na Busca do Conhecimento. **Revista Digital de Biblioteconomia e Ciência da Informação**, Campinas, v.10, n.1 p.53-66, 2012. Disponível em: < https://periodicos.sbu.unicamp.br/ojs/index.php/rdbci/article/view/1896/pdf_28 >.Acesso em: 14 mai. 2021

PRESSMAN, Roger S. **Engenharia de Software** Uma Abordagem Profissional. 7 ed. Porto Alegre: AMGH, 2011

QUINTINO, R. C. G; MOTTA, E. de C; VARGAS, A. A. F. Principais gerenciamentos e planejamentos para automação de testes de software - **PEAD**, n 17, 2019 - Disponível em: < http://www.revista.universo.edu.br/index.php?journal=2013EAD1&page=article&op=view&path%5B%5D=8311&path%5B%5D=4010 > . Acesso em: 12 out. 2022.

RODRIGUES, R. Testes de Aceitação com BDD – Como obter os melhores resultados. **Softdesign**. 2017. Disponível em:< https://softdesign.com.br/blog/testes-de-aceitacao-com-bdd-como-obter-melhores-resultados/ >. Acesso em: 02 nov. 2022

SOMMERVILLE, Ian. **Engenharia de Software.** 9ª ed. São Paulo: Pearson Education, 2011. Disponível em: < http://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf >. Acesso em: 19 abr. 2021.

TOSETTO, M; BELLINI, C.G.P. Gestão Sociotécnica do Teste de Software em Projetos de Sistemas de Informação. **Revista de Gestão da Tecnologia e Sistemas de Informação**, v. 5, n. 2, p.325-346, 2008. Disponível em: < https://www.scielo.br/pdf/jistm/v5n2/08.pdf >. Acesso em: 19 abr. 2021.

TROVÃO, J. C.; DIAS NETO, A. C. Especificação de Processos de Apoio Gerencial ao Processo de Testes de Software. **iSys - Revista Brasileira de Sistemas de Informação**, Rio de Janeiro, v. 8, n. 2, p. 05-33, 2015. Disponível em: < https://www.researchgate.net/publication/342426175_Especificacao_de_Processos_de_Apoio_Gerencial_ao_Processo_de_Testes_de_Software >. Acesso em: 19 abr. 2021.

VELOSO, J. S. et al. Avaliação de Ferramentas de Apoio ao Teste de Sistemas de Informação. **iSys - Revista Brasileira de Sistemas**, Rio de Janeiro, v.3, p. 1-17, 2010. Disponível em: < https://www.researchgate.net/publication/263331201_Avaliacao_de_Ferramentas_de_Apoio_ao_Teste_de_Sistemas_de_Informacao >. Acesso em: 19 abr. 2021.