PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

CRITÉRIOS DE ACEITAÇÃO: UMA COMPARAÇÃO ENTRE TESTES MANUAIS E AUTOMATIZADOS

GIULIANNI DOS SANTOS OLIVEIRA

GOIÂNIA
2022

GIULIANNI DOS SANTOS OLIVEIRA

CRITÉRIOS DE ACEITAÇÃO: UMA COMPARAÇÃO ENTRE TESTES MANUAIS E AUTOMATIZADOS

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. André Luiz Alves

Banca examinadora:

Prof. Me. Joriver Rodrigues Canedo

Prof. Dr. Sibelius Lellis Vieira

GOIÂNIA

2022

GIULIANNI DOS SANTOS OLIVEIRA

CRITÉRIOS DE ACEITAÇÃO: UMA COMPARAÇÃO ENTRE TESTES MANUAIS E AUTOMATIZADOS

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em_____/_____/_____.

_____

Prof. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Orientador: Prof. Me. André Luiz Alves

_____

Prof. Me. Joriver Rodrigues Canedo

_____

Prof. Dr. Sibelius Lellis Vieira

GOIÂNIA

2022

“O teste de software pode ser usado para mostrar a presença de defeitos,
mas nunca para mostrar a sua ausência.”

Edsger W. Dijkstra

AGRADECIMENTOS

Meus agradecimentos vão a todos que de alguma forma contribuíram para eu chegar com tanto esforço ao fim do curso. Em especial a minha família, principalmente ao meu pai que sempre me incentivou a gostar de ler desde criança e sempre investiu na minha educação.

Ao meu orientador André Luiz Alves que aceitou me orientar nessa etapa tão importante do curso, pelos ensinamentos e contribuições que me passou. E que sempre ficou feliz pelas conquistas que consegui.

Aos meus amigos de curso Amanda, Vinicius, Giovanna e Willgnner que me acompanham desde o início e que juntos passamos por tantos momentos durante esses cinco anos. Houve ocasiões felizes e tristes, mas sempre estivemos apoiando uns aos outros como podíamos.

E a Deus por estar tendo essa oportunidade de estar concluindo um curso superior e ter me dado forças e alento para conquistar os meus sonhos.

RESUMO

A qualidade de software é um fator importante para a construção de um produto. E para que ela seja alcançada se faz necessária a atividade de testes. O presente trabalho contém um estudo envolvendo modelos de desenvolvimento de software e técnicas usadas para testes de software. O objetivo é fazer a comparação entre testes manuais e automatizados utilizando critérios de aceitação e assim avaliar as diferenças entre os dois métodos de execução de testes. A partir da pesquisa bibliográfica é desenvolvido uma análise de como testes automatizados são utilizados em uma aplicação Web, utilizando os frameworks de automatização de testes de ponta a ponta: Cypress e Playwright. Os testes tanto manuais quanto automatizados são realizados em níveis de interface da aplicação. Através do estudo foram identificadas diferenças entre os tempos de execução dos testes manuais e automatizados.

***Palavras-Chave**: Engenharia de software. Qualidade de software. Testes de software. Automação de testes. BDD.*

ABSTRACT

Software quality is an important factor for product development. And for it to be achieved, testing is necessary. The present work contains a study involving software development models and techniques used for software testing. The objecive is to compare manual and automated tests using acceptance criteria and evaluate the differences between the two test execution methods. Based on a bibliographic research, an analysis about how automated tests are used in Web application is developed, using the end-to-end test automation frameworks: Cypress e Playwright. Both manual and automated tests are performed at application interface levels. Through this study, differences between the execution time of manual and automated tests were identified.

***Key-Words**: Software engeneering. Software Quality. Software Tests. Test Automation. BDD.*

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE SIGLAS

BDD	*Behaviour Driven Development* ou Desenvolvimento Baseado em Comportamento
CAPTCHA	*Completely Automated Public Turing test to tell Computers and Humans Apart* ou Teste de Turing Público Complemente Automatizado para distinguir entre Computadores e Pessoas
CORS	*Cross-Origin Resource Sharing* ou Compartilhamento de Recursos de Origem Cruzada
DOM	*Document Object Model* ou Modelo de Objeto de Documentos
E2E	*End-to-End* ou De Ponta a Ponta
HTML	*HyperText Markup Language* ou Linguagem de Marcação de HiperTexto
IDE	*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado
JSON	*JavaScript Object Notation* ou Notação de Objetos JavaScript
POM	*PageObjects*
QA	*Quality Assurance* ou Garantia de Qualidade
TDD	*Test Driven Development* ou Desenvolvimento Guiado por Testes
V&V	Verificação e validação
XP	*Extreme Programming* ou Programação Extrema

SUMÁRIO

1 INTRODUÇÃO

Os softwares são essenciais para facilitar a vida dos usuários, seja como um despertador ou um aplicativo de banco, eles se fazem presentes na vida de bilhares de pessoas ao redor do globo. Com o aumento da demanda por essas tecnologias, o uso da Engenheira de Software se fez mais necessária, para assegurar que estes programas sigam os aspectos para prover o que o usuário final necessita (SOMMERVILLE, 2013). Dentre algumas das abordagens da Engenharia de software, tem a qualidade.

Qualidade é um termo que possui diversas definições e que variam conforme a finalidade em que será aplicada. Por possuir tantos sentidos se faz necessário a contextualização em qual situação será abordada.

Segundo Pressman (2011), a qualidade de software é estabelecida em três pontos: deve possuir uma boa gestão durante seu desenvolvimento para possibilitar meios que a qualidade seja alcançada, o produto deve satisfazer as necessidades do usuário e consequentemente trazer benefícios para seu fabricante como a boa reputação da empresa que é o resultado dos dois primeiros itens citados.

Sommerville (2019), afirma que a garantia da qualidade envolve processos e padrões, que quando aplicados resultam na qualidade do produto. Esses processos devem envolver documentações e tarefas bem definidas a cada membro da equipe responsável pelo desenvolvimento. Um software com qualidade deve seguir à risca os requisitos escritos para sua construção.

Os testes de verificação buscam assegurar que os requisitos estão sendo seguidos conforme sua especificação. A verificação estática, foca na análise do código-fonte do produto e nos modelos de especificação do projeto para encontrar defeitos (SOMMERVILLE, 2013).

Uma outra forma de garantir a qualidade de software é realizando os testes de validação, que analisam se o produto corresponde as expectativas dos *stakeholders*. Os testes de validação também podem ser chamados de análise dinâmica (SOMMERVILLE, 2013). Estes possuem diferentes tipos: testes de unidade são realizados em pequenas unidades de código. Já os de integração

garantem que as unidades funcionam em conjunto. E por fim os de ponta a ponta avaliam o funcionamento da interface (BARTIE, 2002).

Quanto mais cedo os testes são realizados, menos custos geram para o projeto. É importante a execução dos testes durante todo o processo da criação de um software para prevenção de defeitos (MYERS; BADGETT; SANDLER, 2012). Assim são aumentadas as chances de um software ser aceito pelas partes interessadas.

Dentro da metodologia ágil os testes são realizados durante todo o processo de construção do software, diferente das metodologias tradicionais que são feitos no fim do processo (MUNIZ et al., 2019).

Nas fases finais do projeto é necessário realizar os testes de aceitação. Esse tipo de teste é o último antes do software ser lançado. Podendo ser executados pelas partes interessadas pelo software, como clientes e *stakeholders*, em uma versão não totalmente finalizada do produto (ISTQB, 2018).

Os testes de aceitação são o diferencial para avaliar se o software está seguindo corretamente os requisitos levantados, avaliando se corresponde às necessidades do cliente (ISTQB, 2018). Ou seja, o software deve estar dentro dos critérios de aceitação definidos entre a equipe responsável pelo desenvolvimento e os *stakeholders* durante a etapa de levantamento de requisitos.

Com o auxílio das divisões da pirâmide de testes, é possível saber quais camadas do software podem utilizar os testes automatizados e manuais. Automatizar os testes evitam tarefas repetitivas e economizam tempo, mas nem tudo pode ser automatizado e os testes manuais ainda se fazem necessários como exemplo, os testes que robôs não podem realizar (MUNIZ et al., 2019).

Diante do contexto introduzido, este trabalho tem como objetivo comparar testes manuais e automatizados utilizados critérios de aceitação em aplicações Web.

Este documento está organizado com a seguinte estrutura: no primeiro capítulo é introduzido o tema do trabalho e os objetivos esperados na pesquisa. No capítulo 2, os conceitos envolvendo engenharia de software e modelos para a

construção de software são descritos. O capítulo 3 apresenta os conceitos da qualidade de software assim como alguns fundamentos de testes.

O capítulo 4 introduz o que será utilizado no experimento, explicando o que são as tecnologias utilizadas para sua realização. Assim, no capítulo 5 é abordado a aplicação da proposta, comentando os resultados obtidos durante o experimento. E finalizando com o capítulo 6, que apresenta as conclusões de todo o fundamento utilizado no trabalho.

2 ENGENHARIA DE SOFTWARE

A Engenharia de Software é um termo que possui diversas definições, mas é possível constatar que ela apoia a criação profissional de um software (SOMMERVILLE, 2013). Pressman (2011, p. 39), refere-se à Engenharia de software como:

> “A engenharia de software é uma tecnologia em camadas. [...] qualquer abordagem de engenharia (inclusive engenharia de software) deve estar fundamentada em um comprometimento organizacional com a qualidade. A gestão da qualidade total [...] promovem uma cultura de aperfeiçoamento contínuo de processos, e é esta cultura que, [...] leva ao desenvolvimento de abordagens cada vez mais efetivas na engenharia de software.”.

O processo de construção de software deve seguir uma série de etapas para que a qualidade seja conquistada (PRESSMAN, 2011). É necessário que essas etapas sejam seguidas para que não somente a construção seja correta, mas que o cliente final seja agradado e a manutenção do software seja realizada pensando em seu tempo de vida a longo prazo (PRESSMAN, 2011).

Não existe apenas uma única forma de construir um software. Há vários modelos para construção de software, mas é necessário a escolha de um processo que seja implementável dentro de uma equipe (WAZLAWICK, 2013). Dentre eles existem alguns modelos: o modelo cascata e o modelo espiral como exemplos de modelos tradicionais, e Scrum e o *Extreme Programming* como exemplos de metodologias ágeis. Cada modelo de construção de software possui suas características que as diferenciam umas das outras. Todos os modelos possuem as mais variadas tarefas, todavia os modelos possuem em comum quatro atividades que são essenciais. Sommerville (2019) as citam como:

- Especificação do software: etapa de levantamento dos requisitos que o sistema deve ter e seguir;
- Desenvolvimento de software: etapa em que o software está sendo construído para que resulte em um executável;
- Validação de software: etapa de testes para verificar se o software atende as regras de negócio e apresenta o comportamento esperado que atenda o usuário final;

- Evolução do software: é a etapa de manutenção, muito importante para que o software continue em operação.

2.1 Modelos de desenvolvimento de software

Segundo a definição da ISO 24765, um modelo de ciclo de vida é uma série de processos e atividades necessárias para a evolução de um produto. Elas se tornam importantes para a comunicação entre a equipe quando organizadas corretamente.

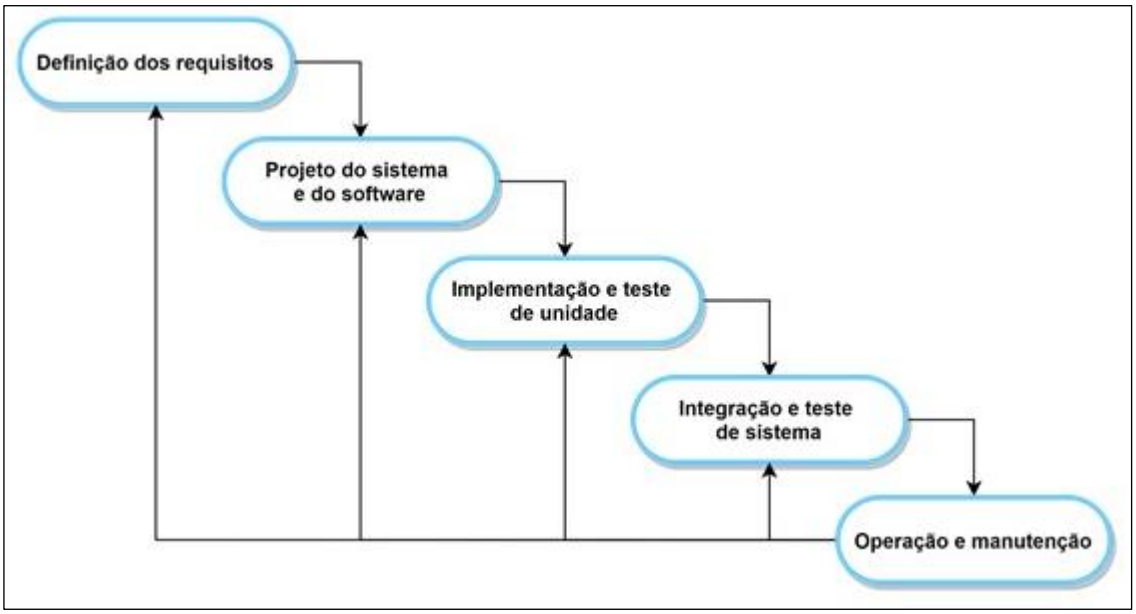
É necessário entender os principais modelos de desenvolvimento de software para que a Engenharia de Software seja bem compreendida. Assim como as tecnologias para a construção de um software se modernizaram, os processos adotados para seu desenvolvimento também evoluíram. Mas as metodologias ágeis não surgiram do nada, foi preciso a existência dos métodos tradicionais e a necessidade de adaptação deles para que essas metodologias fossem criadas.

Quando se trata de modelos tradicionais, são referidos a modelos com processos bem definidos em que uma tarefa depende do término da outra. Elas não podem ser revistas uma vez que finalizadas. Essas características não fazem um modelo ser ruim se comparados às metodologias ágeis, cabe à equipe adotar o que melhor se adapta às necessidades e ao software que se busca projetar (WAZLAWICK, 2013).

Dentre os modelos tradicionais, o mais conhecido é o modelo em cascata – *waterfall* também chamado de ciclo de vida clássico. Ele recebe esse nome por seus processos sequenciais resultarem em um formato semelhante à uma cascata. Por ser um dos primeiros modelos criados para o desenvolvimento de software, ele possui alguns pontos negativos. Dentre eles estão o fato de ser sequencial e impedir que fases anteriores sejam revistas uma vez que finalizadas. Isso se torna uma falha, considerando que em um projeto podem surgir imprevistos durante seu desenvolvimento (PRESSMAN 2011). Ou seja, a falta de flexibilidade acaba gerando custos a mais para o projeto. O modelo cascata possui outras variantes como o modelo V, Sashimi e W. A figura 1 exibe o funcionamento deste modelo.

No modelo cascata, os testes são feitos nas unidades do código para avaliar se eles atendem as próprias especificações. A próxima etapa de teste é o teste de sistema, que integra todas as unidades já avaliadas como um software completo. Esta tarefa tem o objetivo de garantir que os requisitos foram seguidos corretamente (SOMMERVILLE, 2019).

Figura 1 - Modelo em Cascata.



Fonte: Sommerville, 2019.

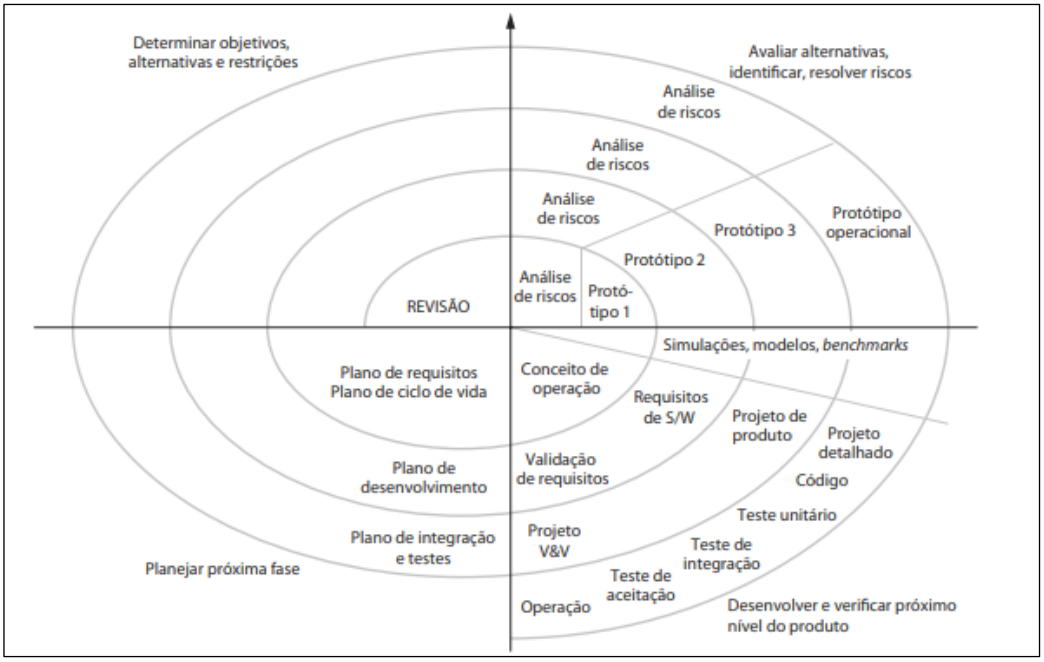
O modelo espiral é focado na redução de riscos, sendo uma versão mais flexível e interativa do modelo em cascata. Cada volta na espiral aborda um novo miniprojeto até que se transforma no projeto completo. Conforme avança e os riscos são analisados e tratados, a implementação vai se transformando em sequencial. A priorização de reduzir riscos pode ser vantajosa ou ser desastrosa, pois esse modelo requere um gerente experiente para que os riscos sejam analisados a princípio. Caso contrário traria graves problemas conforme o projeto segue em andamento. Mas é inegável que esse modelo veio como uma forma de trazer maior flexibilidade para os modelos tradicionais já existentes (WAZLAWICK, 2013).

Na figura 2 é possível visualizar como o processo do modelo espiral funciona, a cada volta uma etapa é adicionada e com a análise de riscos implementada. As primeiras espirais são focadas nas definições iniciais de planejamento do projeto

para confirmar sua viabilidade para que na última volta ele seja finalmente implementado e validado (SOMMERVILLE, 2013).

Conforme a figura 2, os testes no modelo espiral assim como no modelo cascata se encontram nas etapas finais do setor de desenvolvimento, apesar de terem sido planejados em fases anteriores (SOMMERVILLE, 2019). São divididos em três partes, se iniciando com o teste unitário, seguindo para o teste de integração e finalizando com o teste de aceitação.

Figura 2 - Modelo Espiral.

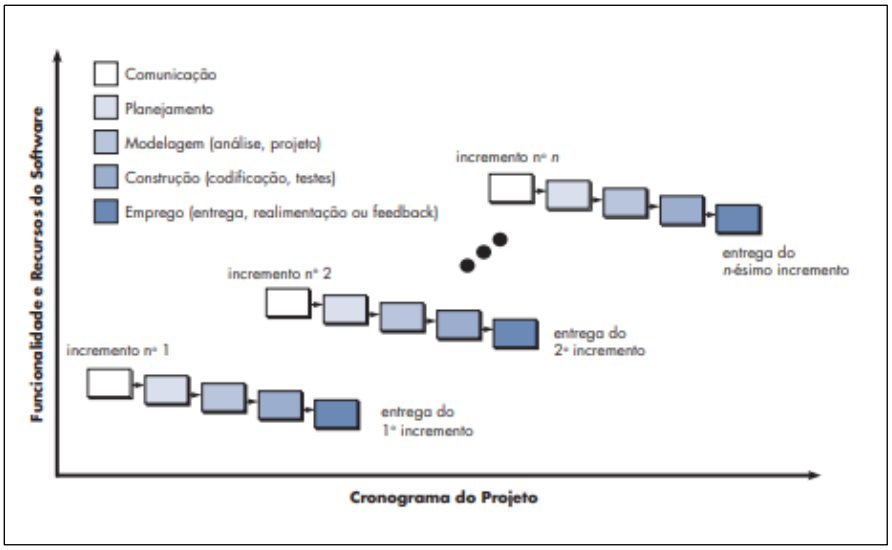


Fonte: Sommerville, 2013.

O modelo Incremental permite que partes do sistema sejam desenvolvidos em paralelo seguindo o que o cliente declarar como prioridade. Como o nome sugere, as entregas são divididas em incrementos, como na figura 3. Conforme eles são entregues, vão sendo juntados aos anteriores no ambiente que o cliente utiliza. Apesar de apresentar vantagens como permitir mudanças constantes e funcionar em equipes pequenas, possui as desvantagens da chance de os novos incrementos não serem totalmente aceitos pelos usuários. Dependendo do gerenciamento da equipe, pode não atender a todos os requisitos dos incrementos que foram lançados anteriormente (SOMMERVILLE 2019).

Os testes no modelo incremental são feitos durante a fase de construção do projeto. Ao final de cada incremento, como descrito na figura 3, os testes fazem parte da etapa anterior a entrega.

Figura 3 - Modelo Incremental.



Fonte: Pressman, 2011.

2.2 Metodologias ágeis

Assim como as tecnologias os processos também evoluem. As metodologias ágeis nasceram da necessidade de métodos para construção de software mais flexíveis e que se adaptam as reais demandas de um projeto. Há uma entrega constante ao invés da entrega de um software completo, o que possibilita que produtos sempre tenhas atualizações de melhoria contínua para que possa ser competitivo no mercado.

O processo se tornou mais adaptativo, desde mudanças de requisitos que ocorrem para condizer com novas atualizações a processos que são executados em sincronia. O cliente tem mais voz e define quais serão as novas implementações baseadas em suas necessidades. Pontes e Arthaud (2019) afirmam “Metodologias ágeis são orientadas a pessoas e não a processos”.

Não somente o processo sofreu modificações, mas as equipes também. Além da mudança da forma de atuação dos profissionais, houve o surgimento de cargos como *Scrum Master*, Agilista e *Product Owner* para que sejam atendidas as novas necessidades das metodologias ágeis.

Criado em 2001, o Manifesto Ágil foi o responsável pela popularização das metodologias ágeis. Reuniu representantes e juntos entraram em consenso para criar uma lista de 12 princípios contendo concepções em comum a elas (ALMEIDA, 2017).

Os doze princípios acordados no Manifesto Ágil (BECK et al., 2001) demonstram a preocupação em satisfazer o cliente buscando modificações constantes para que esse proposito seja atendido com entregas frequentes. Também ressalta a importância do trabalho em conjunto da equipe, estando motivada e tendo foco em melhorias.

2.2.1 Extreme Programming

Extreme Programming, ou XP, é uma metodologia ágil que surgiu nos Estados Unidos durante os anos 1990. Entre suas características está o uso de conjuntos de regras para cada etapa. Wazlawick (2013) afirma que os princípios da XP são:

- Simplicidade: o time deve focar apenas nas funcionalidades necessárias abandonando as que não seriam utilizadas;
- Respeito: os membros da equipe devem demonstrar respeito para um com os outros;
- Comunicação: a melhor forma de se comunicar sempre deve ser buscada para que não haja erros pela falha de comunicação;
- Feedback: feedbacks constantes são importantes para que erros sejam evitados pela falta deles;
- Coragem: mudanças nos momentos corretos são importantes para evitar que problemas se intensifiquem.

Além dos princípios citados acima, o XP também segue regras para realizar seus processos. As tarefas são divididas em Planejamento, Projeto, Codificação e Testes. Pressman (2011) define essas etapas como:

- Planejamento: durante a atividade chamada Ouvir, há a criação das Histórias de Usuário, que são as especificações do que será construído e são escritos pelo próprio cliente. Quando as histórias são finalizadas, o Custo que é o tempo de implementação é estipulado pela equipe e elas são categorizadas em prioridades diferentes. O Custo das versões seguintes são baseados no período que essas histórias levaram para serem criadas;
- Projeto: a Simplicidade é um dos princípios da XP, e durante essa etapa é possível ver sua aplicação. Para que as histórias sejam mais claras é adotado o uso de guia. Para solucionar problemas complicados, são criados protótipos operacionais denominados solução pontual para que os riscos sejam reduzidos;
- Codificação: a programação em dupla é uma das recomendações da XP. Esse método tem o objetivo de garantir a qualidade do código e facilitar a resolução de problemas;
- Testes: testes de unidade são adotados assim que as histórias de usuário são entregues. Esses testes visam facilitar a automatização de testes de integração e sistema para serem realizados durante cada versão a ser entregue.

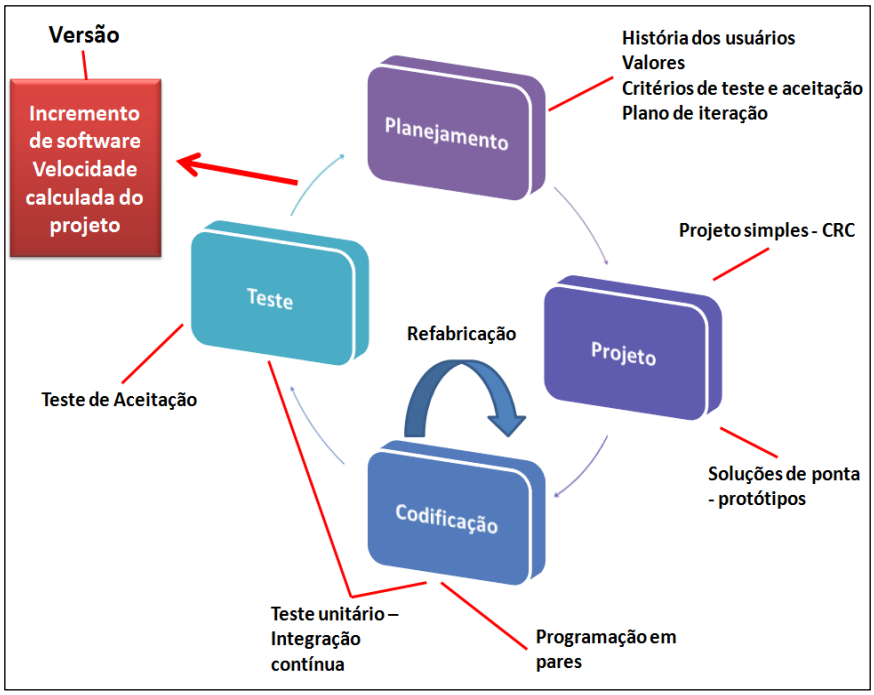
Na figura 4, é possível visualizar o funcionamento do XP. As etapas estão conforme o descrito, contendo: Planejamento, Projeto, Codificação e Testes.

O TDD, ou *Test Driven Development* sendo em português Desenvolvimento Guiado por Testes, é uma prática originária do XP. A execução de testes é realizada antes da implementação e a partir dos testes iniciais as outras estruturas do código são planejadas. Isso permite que o código tenha qualidade e encontre erros o mais cedo possível, resultado do feedback constante (ANDRADE et al. 2011).

Os testes de aceitação são realizados com base nos critérios de aceitação que foram definidos pelos clientes durante a fase de construção das histórias de usuário. No XP, o cliente participa ativamente durante o processo de

desenvolvimento e testes indicando o que deve ser priorizado para ser construído e participando na criação dos testes de aceitação. Assim, os critérios de aceitação se tornam métricas para avaliar se o software está conforme solicitado (SOMMERVILLE, 2011).

Figura 4 - O processo do XP.



Fonte: Kolb, 2013.

2.2.2 Scrum

Scrum é uma das metodologias ágeis mais conhecidas e utilizadas pelas equipes de desenvolvimento.

Foi criada por Jeff Sutherland em colaboração com Ken Schwaber em 1993, diante da necessidade de mudanças nos métodos utilizados. O Modelo em Cascata era muito utilizado. Mas por causa de suas características, ocasionavam atrasos aos projetos resultando estrapolações de custos e em softwares que não agradavam o cliente (SUTHERLAND, 2014).

O Scrum tem como características ser um processo adaptativo baseado no empirismo, tendo três pilares: Transparência, Inspeção e Adaptação. O

desenvolvimento ocorre em períodos denominados *Sprints* que podem durar a partir de uma semana. A cada *Sprint*, o *Product Owner* - que é o representante dos clientes - repassa para a equipe através do *backlog* aquilo que ele planejou para ser entregue durante esse período (RIBEIRO, 2015). O *Backlog* reúne aquilo que será implementado.

Schawaber e Sutherland (2020) definem os três pilares do Scrum como:

- Transparência: o processo deve ser visível a todos que participam dele. Quanto mais baixa a transparência, maiores são os riscos;
- Inspeção: todo o processo do Scrum deve ser inspecionado para que se descubram problemas e como consequência gerar mudanças;
- Adaptação: o processo deve ser modificado conforme o resultado para que os desvios que poderiam surgir sejam ajustados o quanto antes.

As cerimônias são importantes para o andamento das Sprints. Durante o planejamento, reuniões de aberturas com a equipe são realizadas para apresentação do *Backlog* e o time define aquilo que entrará na *Sprint* e seu principal objetivo a ser alcançado. Assim como o planejamento, encerramentos são realizados. Uma das cerimônias de encerramento é a reunião de revisão. Aquilo que foi construído durante a *Sprint* é debatido e no final um feedback é dado em relação ao que foi entregue. Outra reunião de encerramento é a reunião de retrospectiva que consiste em interações entre a equipe para que verifiquem os pontos bons e os que podem ser melhorados em *Sprints* futuras (RIBEIRO, 2015).

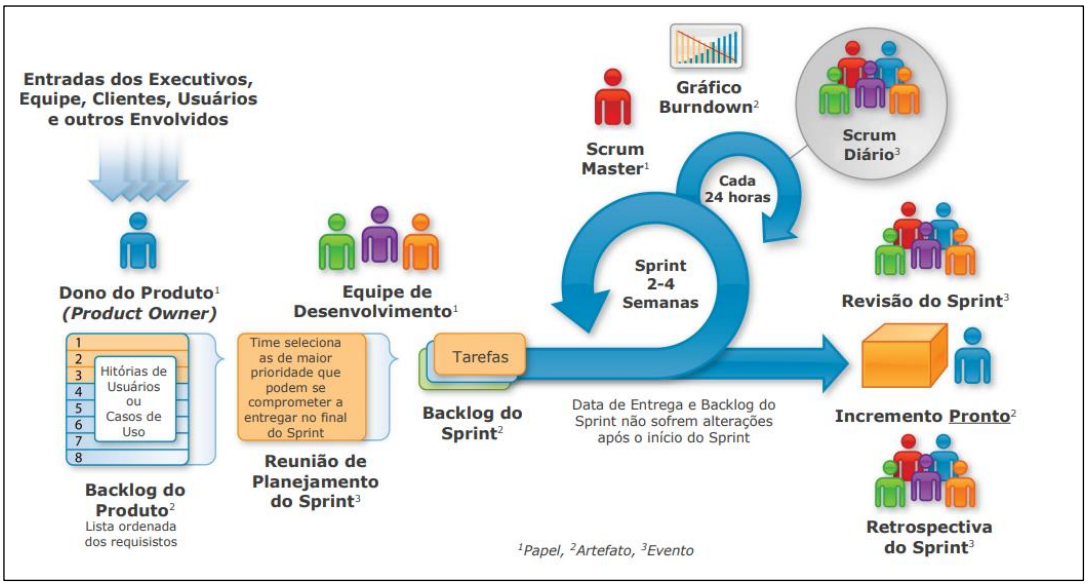
Reuniões diárias também ocorrem, elas são nomeadas como *Daily* e cada membro do time descreve sobre as atividades que realizou no dia anterior, o que irá realizar durante o dia atual e se possui algum impedimento para alguma tarefa que foi ou será realizada (RIBEIRO, 2015).

Na figura 5, são ilustrados todos os processos envolvidos na execução do *Scrum*.

O papel do analista de qualidade, ou QA, num time Scrum contribui auxiliando o time durante toda a execução da *sprint*. Esse profissional é envolvido com a construção dos critérios de aceite a partir das histórias de usuário que são feitas pelo *Product Owner*. O QA também ajuda os desenvolvedores em testes unitários e

desenvolve testes funcionais automatizados. Outra responsabilidade desse profissional é planejar e executar os testes funcionais (HASIJA, 2012).

Figura 5 - Ciclo do Scrum.



Fonte: Braz, 2011.

3 QUALIDADE DE SOFTWARE

Qualidade de software é um termo debatido por vários autores e apesar da variedade de significados, alguns pontos entram em consenso. Para a definição de qualidade, neste presente trabalho, será utilizada a definição dada pela ISO/IEC/IEEE 24765 (2017). Um software com qualidade é aquele que satisfaz os requisitos levantados em acordo com as partes interessadas, assim como deve atender as necessidades dos usuários.

Para que seus requisitos sejam atendidos, o produto precisa passar por um processo de construção com qualidade. Quando os requisitos não são totalmente atendidos demonstra os problemas do processo em o software que foi construído (PAULA FILHO, 2000). Para cada requisito levantado não satisfeito resulta em um defeito.

Além dos requisitos, o sistema também deve atender as expectativas do usuário. Isso quer dizer que características como usabilidade, manutenibilidade, segurança e confiabilidade devem ser atendidos. Esses requisitos são denominados não funcionais (SOMMERVILLE 2019). Na figura 6, além dos que foram mencionados, outros requisitos não funcionais de um software são listados.

Um software com usabilidade ruim afasta seus usuários, que iriam procurar por outros sistemas que tenham funcionalidades semelhantes e sejam mais simples de utilizar. Nem sempre é possível atender todos os atributos, mas para cada projeto deve verificar quais são as prioridades e quais devem ser atendidos (SOMMERVILLE 2019).

Figura 6 - Requisitos não funcionais.

Segurança	Compreensibilidade	Portabilidade
Proteção	Testabilidade	Usabilidade
Confiabilidade	Adaptabilidade	Reusabilidade
Resiliência	Modularidade	Eficiência
Robustez	Complexidade	Capacidade de aprendizado

Fonte: Sommerville, 2019.

3.1 Erros, defeitos e falhas

Há o equívoco que garantia da qualidade está relacionado somente a testes, mas ela está diretamente ligada aos processos de construção de software. Não seguir os processos de forma correta resultam no surgimento de defeitos, mas isso não isenta um software de não possuir defeitos (ISTQB, 2018).

Ainda segundo a ISTQB (2018), erro, defeito e falha são definidos como:

- Erro: um erro é ocasionado por fatores humanos, gerado pela inconsistência de uma pessoa;
- Defeito: também denominado *bug*, ocorre quando o erro não foi corrigido. Podem ser encontrados no código, na documentação ou em algum processo relacionado ao software;
- Falha: elas podem derivar de defeitos, mas nem sempre um defeito ocasiona uma falha. Ocorre quando o comportamento de uma função em um software não é o esperado e quando o defeito é executado.

Há uma pequena diferença entre as definições de erro e defeito, quando analisados sob a perspectiva da IEEE, na qual defeito se caracteriza pelo engano de alguém, enquanto o erro é consequência executada desse defeito que por fim se torna uma falha (DELAMARO; MALDONADO; JINO, 2007).

3.2 Verificação e validação

Para assegurar que o software esteja seguindo os processos corretos para alcançar a qualidade, se faz necessário a realização das atividades de validação e verificação.

A definição de Verificação e Validação (V&V) segundo a ISO/IEC/ IEEE 24765 (2017) é a comprovação de que o software está buscando a garantia de qualidade. Enquanto a verificação confirma se os requisitos levantados estão sendo seguidos, a validação analisa se o produto está satisfazendo o que os clientes esperam e necessitam.

As técnicas de verificação estática têm como característica não envolver um arquivo executável. Ou seja, elas visam o código-fonte e a documentação do

programa para descobrir erros antes da geração de um programa. A inspeção, uma técnica que avalia o código-fonte de um software e a revisão, que examina a documentação de um projeto são técnicas de verificação estática. Essas atividades buscam a qualidade do projeto (SOMMERVILLE, 2013).

O teste de software é um processo que deve contemplar tanto a verificação quanto a validação e não apenas um dos procedimentos (ISTQB, 2018).

3.3 Testes de software

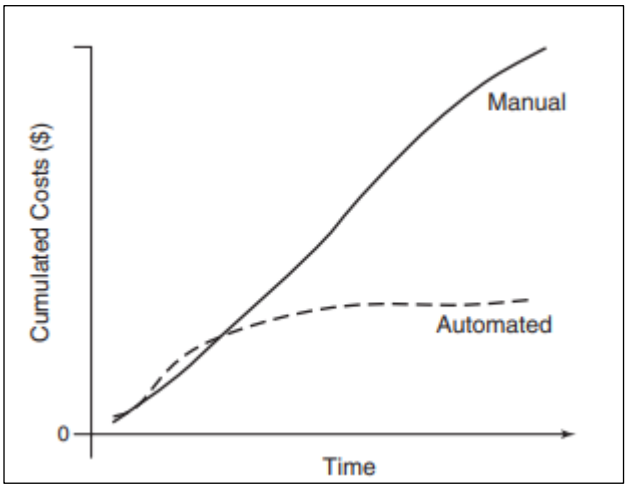
Os testes de software são técnicas de validação dinâmica, pois tem como um dos objetivos analisar se o comportamento está conforme o esperado (BARTIE, 2002). Testes de software podem ser definidos como uma série de atividades que expõem a existência de defeitos. Para que os testes sejam eficazes não basta apenas que eles sejam executados, é necessário que sejam utilizadas estratégias para que eles sejam competentes. Dentre essas estratégias tem a aplicação de técnicas, elas podem ser aplicadas tanto no código fonte quanto no sistema já contendo interface. Os testes devem ser feitos tanto por desenvolvedores quanto por analistas de testes (PRESSMAN, 2011).

Sommerville (2019) afirma que os testes devem revelar duas coisas, que o software está seguindo os requisitos propostos e encontrar defeitos para que sejam corrigidos. Além das estratégias de testes, existem outras formas de realizá-los. Tanto manualmente, feito pelo analista de testes diretamente na aplicação quanto de modo automatizado em que são utilizados *frameworks* como intermédio para criar *scripts* para que seja sumulado o comportamento de um usuário real. Porém há algumas situações em que não é possível realizar a automatização, um exemplo é escrever o código de um CAPTCHA durante a autenticação que geralmente são feitos para não permitir que *bots* realizem essa tarefa (MARGALHAES, 2021).

Os testes manuais dificilmente serão extintos, principalmente quando se trata da averiguação da usabilidade de um sistema ou a sua acessibilidade. Porém, muitas vezes é necessário a automatização para que evite que tarefas repetitivas sejam feitas, o que resulta em economia de tempo e como consequência a redução

de custos (MUNIZ et al., 2019). A figura 7 ilustra o comparativo do custo da manutenção entre o teste manual e automatizado em relação ao tempo.

Figura 7 - Comparativo de custos entre testes manuais e automatizados.



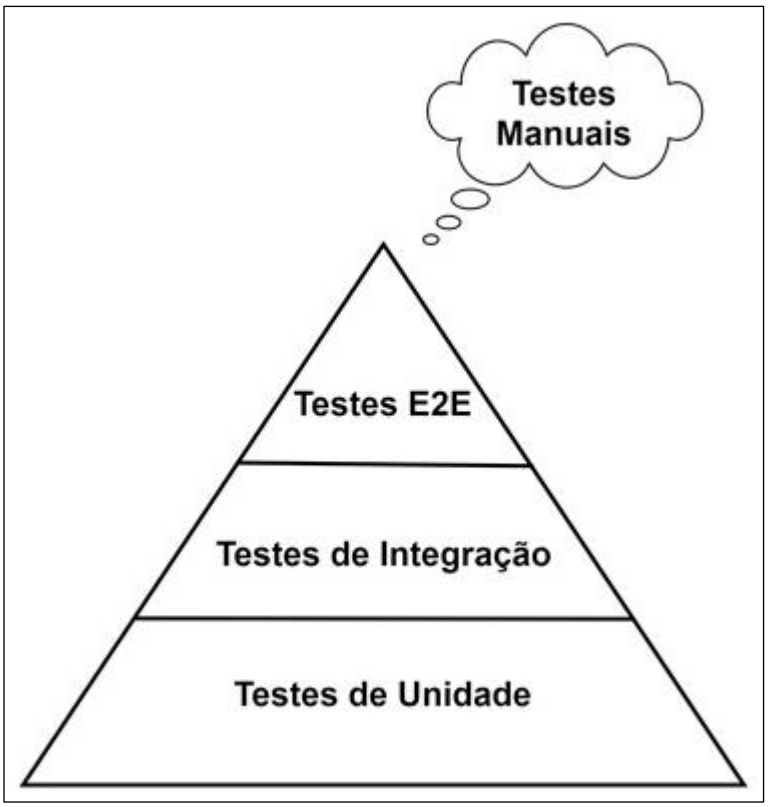
Fonte: Dustin; Garret; Gauf, 2009.

3.3.1 Pirâmide de testes

Cohn (2010) afirma que antes da implementação das metodologias ágeis, automatizar testes era um processo muito caro e muitos times não adotavam a prática. Mas com a adoção de metodologias ágeis como o Scrum, a automatização de testes se faz necessária quando a questão do tempo entre as *sprints* em que o tempo disponível para correções é curto. Os testes que poderiam ser feitos de forma manual em horas podem ser reduzidos a minutos.

A pirâmide de testes se faz necessária para que a qualidade seja garantida, realizando menos testes conforme vai se aproximando do topo. Porém, com essa proximidade eles ficam cada vez maiores e mais lentos. A automatização de testes deve ser executada conforme a figura 8, em que os testes de unidade necessitam de mais avaliações para que as unidades do código sejam cobertas. Em contraposição, há os testes de interface que devem ser realizados de forma mais detalhada para a obtenção dos resultados em cima de cenários e testes de regressão (CAMPOS, 2019).

Figura 8 - Pirâmide de testes.



Fonte: Adaptado de Zarelli, 2020.

Iniciando pela base da pirâmide, os testes de unidade são aqueles que de responsabilidade do desenvolvedor, mas há empresas que atribuem essa tarefa aos analistas de testes para que eles auxiliem o desenvolvedor nessa tarefa. O teste consiste na separação do código em pequenas partes para analisar se elas funcionam separadamente (ZARELLI, 2020). Os testes de unidade também podem ser divididos em componentes maiores, mas que contenha unidades coesas. Esses testes são realizados com a ajuda de ferramentas de depuração (SWEBOK GUIDE, 2004).

Em um degrau mais acima, existem os testes de integração. Esse tipo de teste tem como objetivo analisar se as unidades funcionam de forma integrada. É importante a execução dessa etapa para que evite que os defeitos passem para o próximo nível (ISTQB, 2018).

E por fim, o teste *End to End* (E2E), ou em português, testes de Ponta a Ponta. Esses testes acontecem com o software executado em um ambiente próprio para a realização deles. Na maioria das equipes eles são criados pelo analista de

testes e tem como objetivo simular o comportamento de um usuário (CAMPOS, 2019).

3.3.2 Tipos de teste

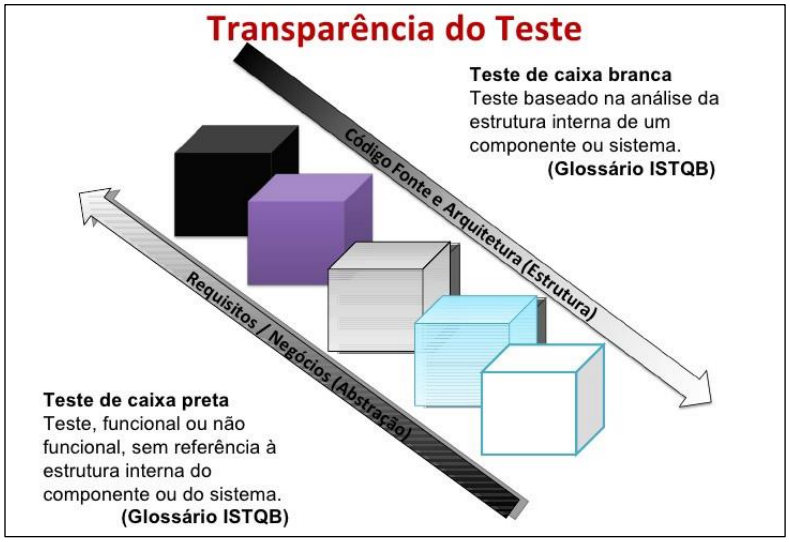
Para que os testes sejam praticados, é importante saber em que se deseja aplicá-los. Por isso se faz necessário entender como são classificadas essas características (ISTQB, 2018).

Os testes de caixa preta têm como diferencial não conhecer a parte interna da aplicação, ou seja, é feita a partir das funções do sistema sem precisar ter conhecimento de seu código (MYERS; BADGETT; SANDLER, 2012). O comportamento do software é avaliado e é necessário o conhecimento dos requisitos funcionais para realizar este teste. Há um equívoco quanto a sua utilização em relação ao de caixa branca, esta é uma técnica complementar a de caixa preta e não alternativa (PRESSMAN, 2011).

A abordagem de caixa branca busca realizar testes na estrutura interna de um sistema. Para que sejam realizados, é preciso entender como o software foi construído para buscar defeitos em estruturas internas como arquitetura, banco de dados e código (BARTIE, 2002).

A figura 9 demonstra como funcionam os testes de Caixa Preta e Caixa Branca. Por não conhecer a estrutura interna do software o teste se intitula Caixa Preta, já Caixa Branca se dá por ter acesso ao código. A partir desses níveis de teste é possível adotar as técnicas derivadas para que o software seja testado corretamente (CARVALHO, 2019).

Figura 9 - Esquema da representação dos testes de caixa preta e caixa branca.



Fonte: Carvalho, 2019 (apud RIBEIRO, 2010).

3.3.3 Testes de aceitação

Testes de aceitação são testes que avaliam se o software está em pleno funcionamento e se agrada o usuário final. O objetivo é determinar que o sistema esteja seguindo os critérios necessários para ser aceito, tanto em seu desenvolvimento e manutenção (CSTE, 2006).

Existem algumas classificações para os testes de aceitação. Segundo o ISTQB (2018), as classificações são as seguintes:

- Teste de aceitação de usuário (UAT): realizado em um ambiente para testes ou em produção, busca analisar se o sistema está em conformidade com o que o usuário espera;
- Teste de aceite operacional (OAT): este teste avalia se o sistema atende as especificações não funcionais, como segurança, performance entre outros;
- Testes Alfa e Beta: são realizados por usuários reais, mas tem certas diferenças. Enquanto o teste Alfa é realizado por usuários em um grupo mais fechado com usuários internos, o teste Beta é realizado por prováveis clientes.

O *agile testing* permitiu que o teste entrasse em todas as etapas de desenvolvimento, com a implementação de testes de unidade aos testes de interface. Com o curto período que tem para entrega, é necessário obter um feedback constante em relação a usabilidade do sistema. Para isso as equipes buscam cada vez mais automatizar testes para agilizar o processo (MYERS; BADGETT; SANDLER, 2012).

Em uma perspectiva ágil, o uso do BDD - uma técnica de desenvolvimento orientado a comportamento - permite que as histórias de usuário se tornem parte do critério de aceitação e com isso os cenários escritos a partir delas permitem a automatização desses testes. Ou seja, os testes de aceitação validam os cenários das histórias de usuário (ASSIS, 2016).

Na pirâmide de testes da figura 8 citada na seção 3.3.1, os testes de aceitação estão encaixados na última camada, equivalente aos testes E2E. Isso acontece por sua característica de se aproximar do comportamento de um usuário (MACHADO, 2017).

Nem sempre as empresas utilizam usuários reais para realizar esses testes, então muitas vezes automatizar se faz menos custoso do que realizar testes manuais com os clientes. Esses testes também são capazes de captar problemas que somente testes de unidade e testes de integração não conseguem (HUMBLE; FARLEY, 2010).

4 PROPOSTA DE APLICAÇÃO

Para construção da proposta de solução, é necessário o uso de algumas ferramentas para realizar a abordagem dos critérios de aceitação em uma aplicação web.

Uma aplicação Web é definida como um sistema que executa em navegadores (como o Firefox, Chrome e Edge). Isso quer dizer que não é necessário a instalação da aplicação por causa da sua comunicação com protocolos HTTPS. A diferença entre um site comum e um aplicativo Web é permitir que o usuário tenha mais interações com suas funcionalidades, como uma página de login por exemplo (NOLETO, 2020).

4.1 Ferramenta de automatização

Para a automatização dos testes, os *scripts* de testes estão na linguagem JavaScript. *Scripts* de testes são definidos como uma série de instruções para o procedimento de teste (ISO/IEC/IEEE 24765, 2017). Os *scripts* automatizados devem ser redigidos em uma ferramenta capaz de interpretar e realizar os procedimentos.

4.1.1 Cypress

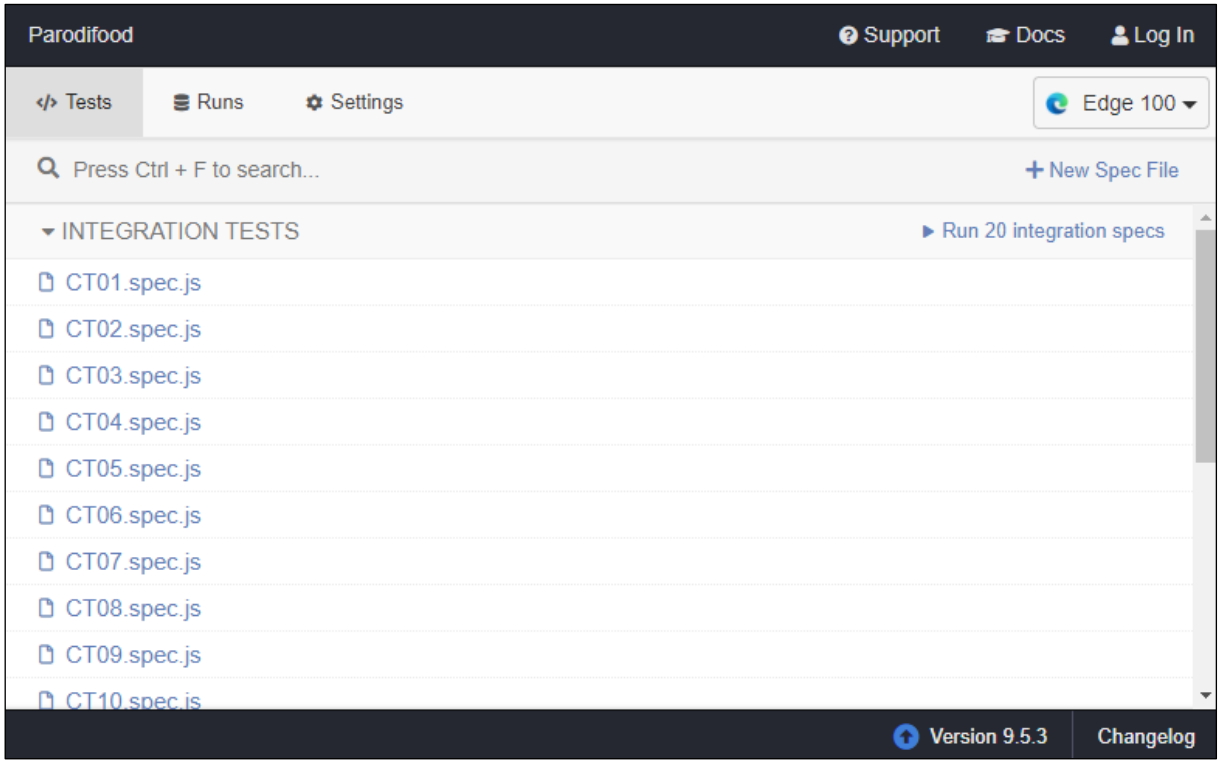
O *Cypress* é uma ferramenta de automatização de testes E2E. Para a criação de seus *scripts* ele aceita as linguagens JavaScript e TypeScript, porém tem o diferencial do uso da sintaxe “*cy*” antes de qualquer comando. Sua arquitetura é construída sobre um processo de servidor node, então é possível configurar seus arquivos para aceitar comandos node (CYPRESS, 2022). Node é uma ferramenta JavaScript assíncrona (NODE, 2022).

Diferente de seus concorrentes, como o *Selenium*, o *Cypress* executa seus testes diretamente no navegador. Apesar disso, ele apenas opera em alguns navegadores com base no *Chromium* e no Firefox (CYPRESS, 2022).

Os navegadores com base em *Chromium* são aqueles que foram desenvolvidos originários do projeto *Chromium* do Google, que disponibiliza o código fonte de forma que todos tenham acesso, sendo assim chamado de *open source*. Dentre esses navegadores existem alguns como exemplo: o Microsoft Edge, Brave, Opera e o próprio Google Chrome (LISBOA, 2021).

Porém, para este trabalho será utilizado apenas o navegador Microsoft Edge, que já vem instalado com o próprio *Cypress*, conforme a figura 10.

Figura 10 - Interface do *Cypress Test Runner*.



Fonte: De autoria própria.

Para a sintaxe da estrutura dos testes, apenas o *Mocha* é suportado. Mocha se trata de um *framework* para estrutura de teste que faz testes assíncronos, a figura 11 traz um exemplo da sintaxe que o *Mocha* permite (MOCHA, 2022). Vale ressaltar que um *framework* se trata de um conjunto de códigos prontos que juntos resultam em uma funcionalidade (NOLETO, 2020).

Figura 11 - Exemplo de estrutura de teste Mocha.

```
var assert = require('assert');
describe('Array', function () {
  describe('#indexOf()', function () {
    it('should return -1 when the value is not present', function () {
      assert.equal([1, 2, 3].indexOf(4), -1);
    });
  });
});
```

Fonte: Mocha, 2022.

Já para as asserções, é utilizado a biblioteca *Chai* (CYPRESS, 2022). Alguns exemplos das asserções podem ser vistos na figura 12. Todos são instalados em conjunto com o *Cypress* mediante o uso de um comando node.

Figura 12 - Asserções para testes da biblioteca *Chai*.

Fonte: Chai, 2022.

4.1.3 Playwright

Playwright é uma ferramenta de testes *End-to-End* para aplicativos web. Tem como principais características funcionar em navegadores com base no

Chromium, Firefox e Webkit. Também aceita linguagens de programação como TypeScript, JavaScript e Python, .NET e Java (PLAYWRIGHT, 2022).

4.1.4 Padrão PageObject

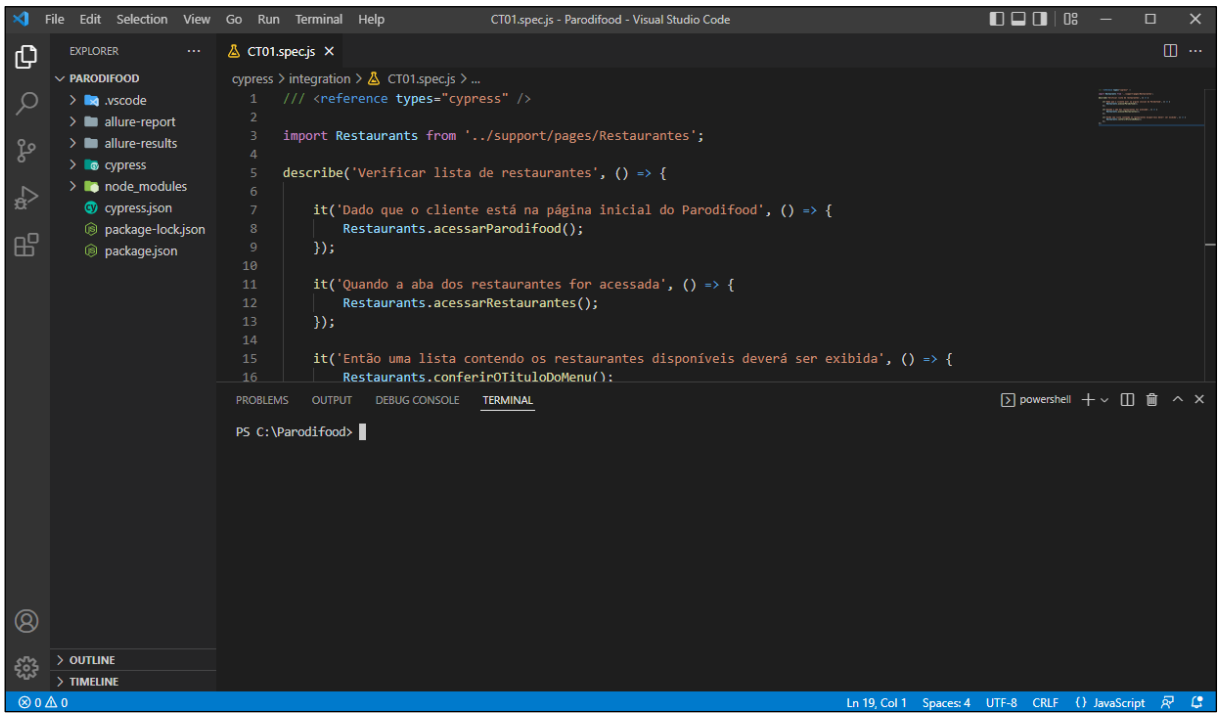
O padrão *PageObject*, ou POM, está sendo adotado para que no arquivo do *script* em que o teste será aplicado, estejam disponíveis apenas os passos que simulam a atividade de um usuário. Os objetos da página são mantidos em uma classe a parte para que não haja a repetição de asserções nos *scripts* de teste (FOWLER, 2013).

A importância da adoção desse padrão está facilitar a manutenibilidade dos testes pois uma aplicação sempre sofre mudanças em seu código, além de deixar os *scripts* mais limpos (MARINHEIRO, 2020).

4.2 Visual Studio Code

O IDE para criação e edição dos *scripts* será o *Visual Studio Code*, da Microsoft. A ferramenta é gratuita e que suporta diversas linguagens de programação assim como plugins para auxiliar em seu funcionamento (VITORIANO, 2020). A figura 13 mostra a interface do *Visual Studio Code*, com um projeto aberto. É visível algumas funcionalidades sendo utilizadas como o *explorer* e o terminal.

Figura 13 - Interface do *Visual Studio Code*.



Fonte: De autoria própria.

4.3 Casos de teste

Os casos de testes são descritos como um conjunto de entradas, pré-condições e resultados esperados cujo objetivo é confirmar se um requisito foi atendido. É um documento usado por testes que contém um passo a passo bem definido (ISO/IEC/IEEE 24765, 2017).

Os testes de aceitação são utilizados para analisar se os requisitos estão sendo atendidos e as expectativas dos clientes (SWEBOK, 2004), porém o foco do trabalho não está na realização da etapa de requisitos, mas sim na realização dos testes. Então para que se tenha o conhecimento do comportamento das aplicações que serão testadas sem o acesso aos requisitos em que foi projetado, foram criados casos de testes mapeando suas principais funções, assim se conhece as entradas utilizadas para que se tenha conhecimento dos resultados que a aplicação retornará. Os casos de teste atuarão como os critérios de aceite para o software.

Para a criação dos scripts de teste foram definidos os seguintes itens:

- Título: título do caso de teste;

- Cenário de teste: cenário relacionado ao caso de teste;
- Ator: aquele que realizaria a ação;
- Objetivo: qual o objetivo que se deve obter ao realizar o teste;
- Pré-condições: como o ambiente deve estar preparado para que seja possível a execução do teste;
- Entradas: o que se deve fazer para completar o teste;
- Resultado esperado: resultado que seria esperado com a execução do teste;
- Data e hora da execução: data e hora em que o teste foi executado;
- Analista responsável pela execução do teste: quem foi o responsável pela realização do teste;
- Resultado real: como o sistema se comportou em resposta ao teste realizado.

4.4 Cenários de teste

Os cenários de testes serão escritos na linguagem *Gherkin*. O *Gherkin* é uma terminologia que faz parte da prática BDD – *Behaviour Driven Development* (Desenvolvimento Baseado em Comportamento) – que visa a criação de software considerando o comportamento do usuário. A linguagem *Gherkin* foi criada para que simplifique a comunicação entre os clientes e a equipe de desenvolvimento. Sua sintaxe consiste no uso dos termos **Dado**, **Quando** e **Então** (WYNNE; HELLESOY, 2012).

Os cenários serão escritos sob a perspectiva de dois tipos de cenários: os que seguem o fluxo “feliz” e os que seguem o fluxo “alternativo”. O fluxo “feliz” é definido como aquele que simulam situações de sucesso, enquanto os caminhos alternativos buscam situações que os usuários nem sempre conseguem concluir uma tarefa (HUMBLE; FARLEY, 2010).

Os cenários acompanharão os scripts de teste para que ter uma forma de entender os passos que serão realizados naquele arquivo. Os cenários devem ser escritos de forma declarativa, ou seja, não devem possuir muitos detalhes como

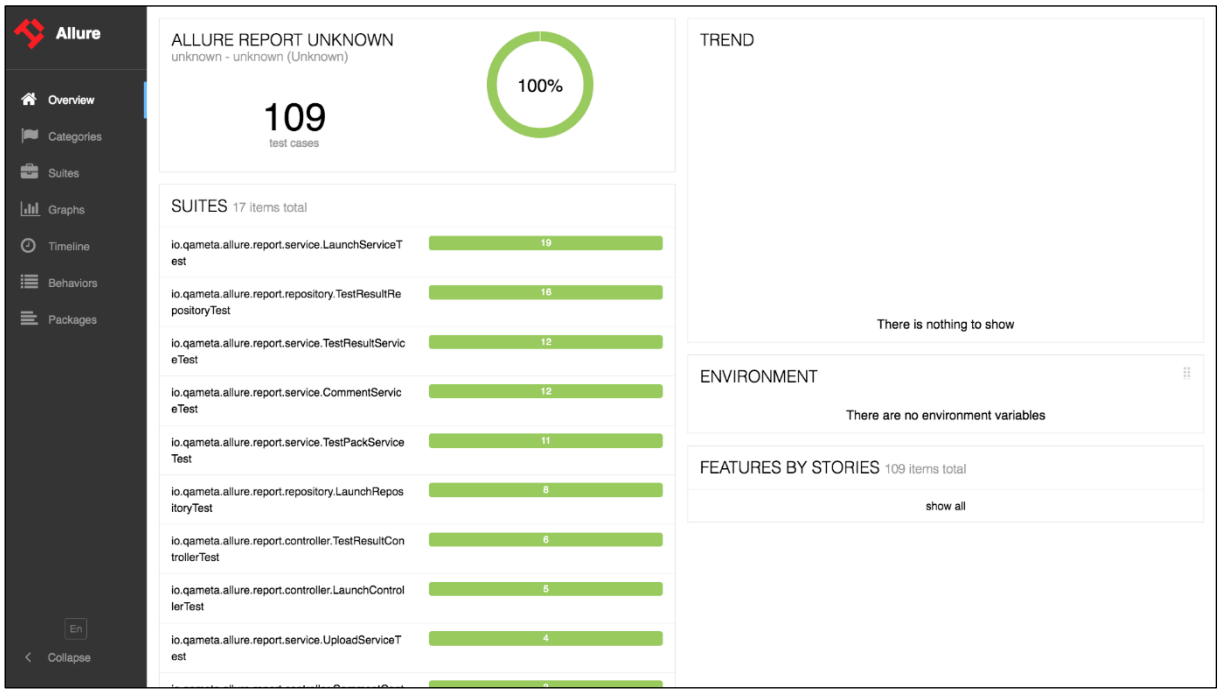
inserções de campos e clique em botões. Apenas o que necessário para que seja simples de entender (CUCUMBER, 2022).

4.5 Allure Framework

O *Allure Framework* é uma ferramenta que elabora relatórios de *framewoks* de testes representando dados dos resultados. Retorna informações como tempo de execução, porcentagem de sucesso e falha, entre outros. Tudo organizado em gráficos (ALLURE FRAMEWORK, 2022).

A figura 14, demonstra um exemplo de uma representação visual dos testes executados em um relatório gerado pelo *Allure*.

Figura 14 – Painel de um relatório gerado pelo *Allure*.



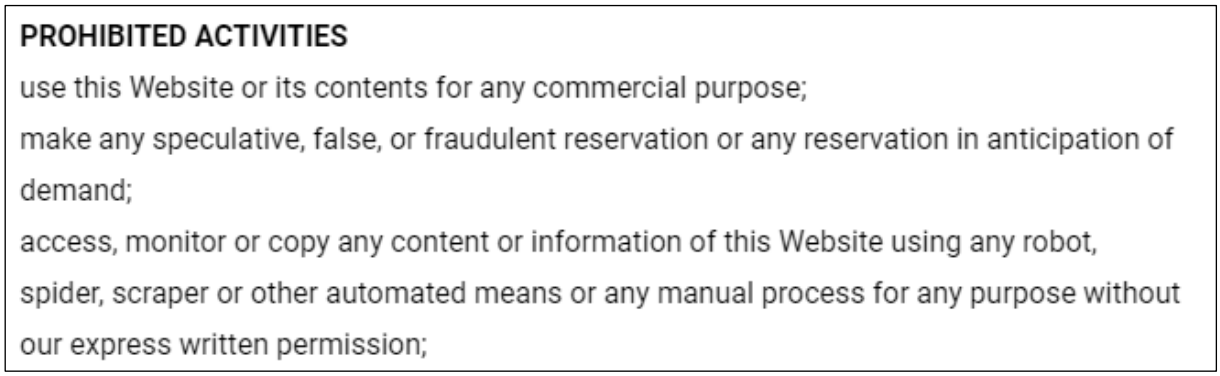
Fonte: Allure, 2022.

5 APLICAÇÃO E ANÁLISE DOS RESULTADOS

Nesta sessão é descrito o processo da implementação do estudo, assim como seus resultados.

Para a aplicação do experimento prático, originalmente se buscava três aplicações diferentes desenvolvidas para estudo da prática de testes automatizados. Porém, um deles, o *Php Travels* não está mais destinado para a automatização. O que foi informado no próprio site como descrito como atividades proibidas, conforme está descrito na figura 15. O uso de qualquer meio automatizado sem a permissão escrita está proibido.

Figura 15 - Termo de atividades proibidas pelo site *PHP Travels*.

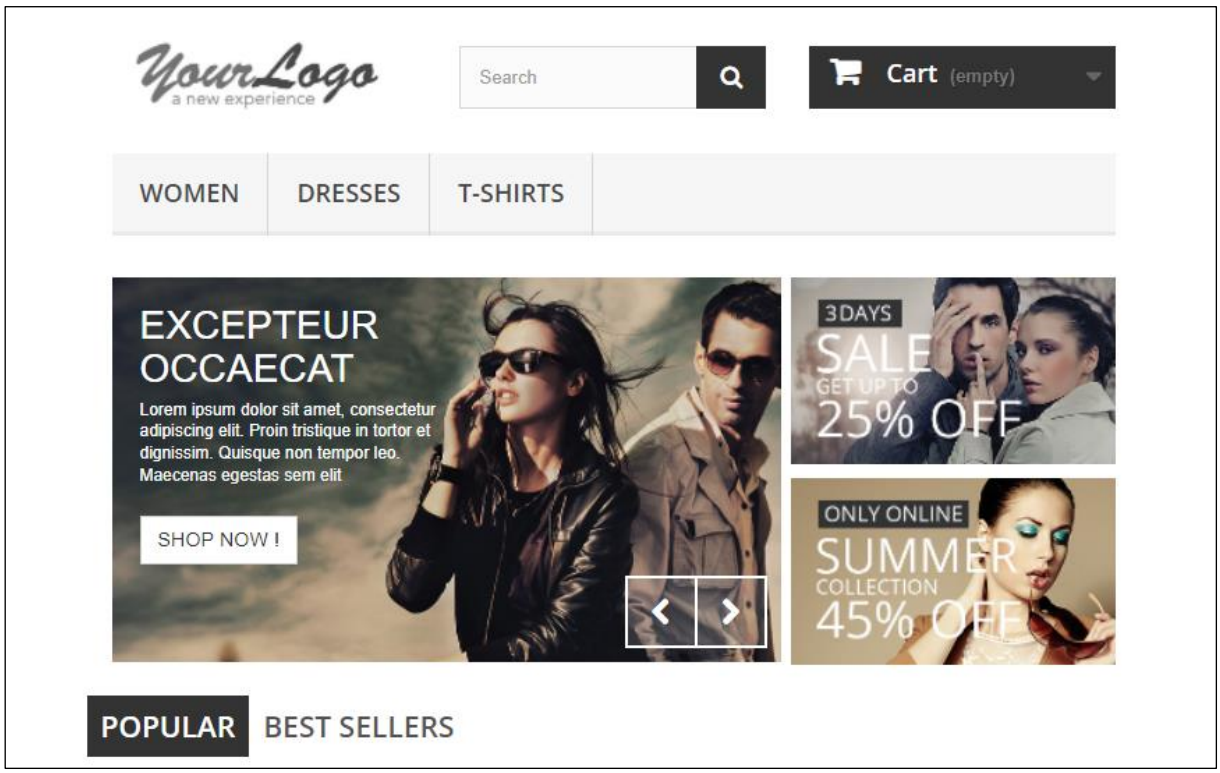


Fonte: Php Travels, 2022.

5.1 Website My Store

O *My Store* é um website voltado para a prática da automatização. Ele simula uma loja de roupas fictícia, conforme mostrado na figura 16. É possível simular compras, consultas por produtos e criar uma conta de cliente que permite a adição e modificação de dados pessoais.

Figura 16 - Página inicial do website *MyStore*.

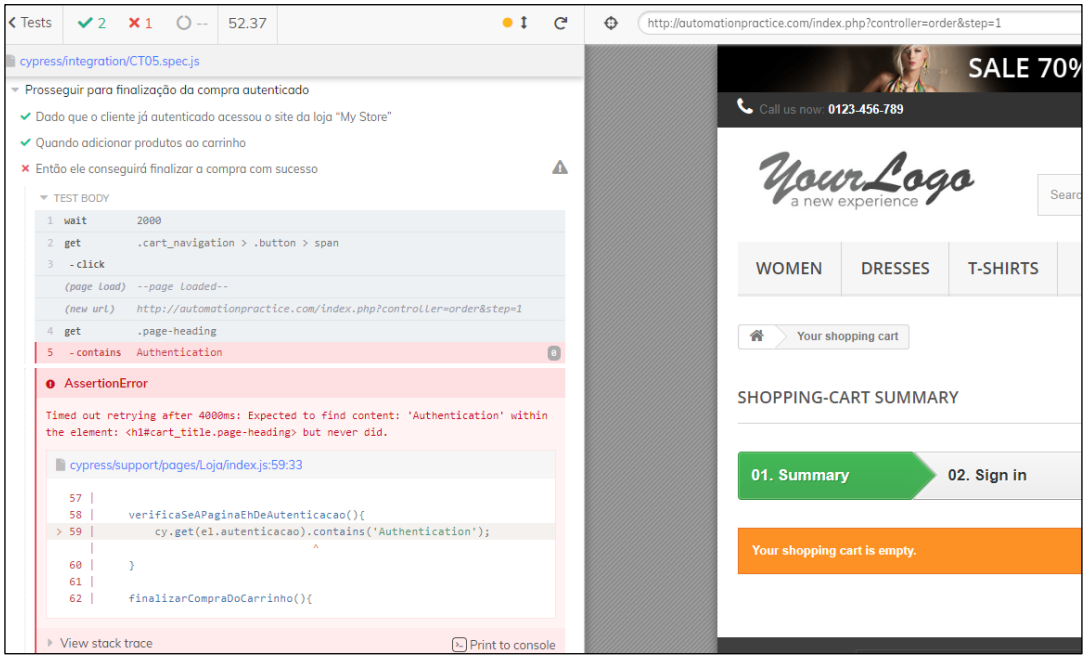


Fonte: My Store, 2022.

Era pretendido o uso do *framework Cypress* para a automatização do website *My Store*, porém foi encontrado um empecilho e não foi possível prosseguir com a automatização através dele. O site *My Store* foi construído com o mecanismo CORS (*Cross-Origin Resource Sharing*), ou seja, o website compartilha recursos entre diferentes domínios (HILLMAN, 2022).

O *Cypress* possui uma limitação que não permite a visita de mais de um superdomínio por teste (CYPRESS, 2022). Então ao mudar o domínio como por exemplo, durante a finalização de uma compra, o *Cypress* não será capaz de terminar a execução do teste, como apresenta na figura 17. As compras foram adicionadas ao carrinho, mas ao prosseguir para a finalização da compra uma nova url é carregada, assim o carrinho é dado como vazio por causa da limitação do *Cypress* e o teste resulta em erro.

Figura 17 - Erro do *Cypress* ao não conseguir completar o teste.



Fonte: Autoria própria.

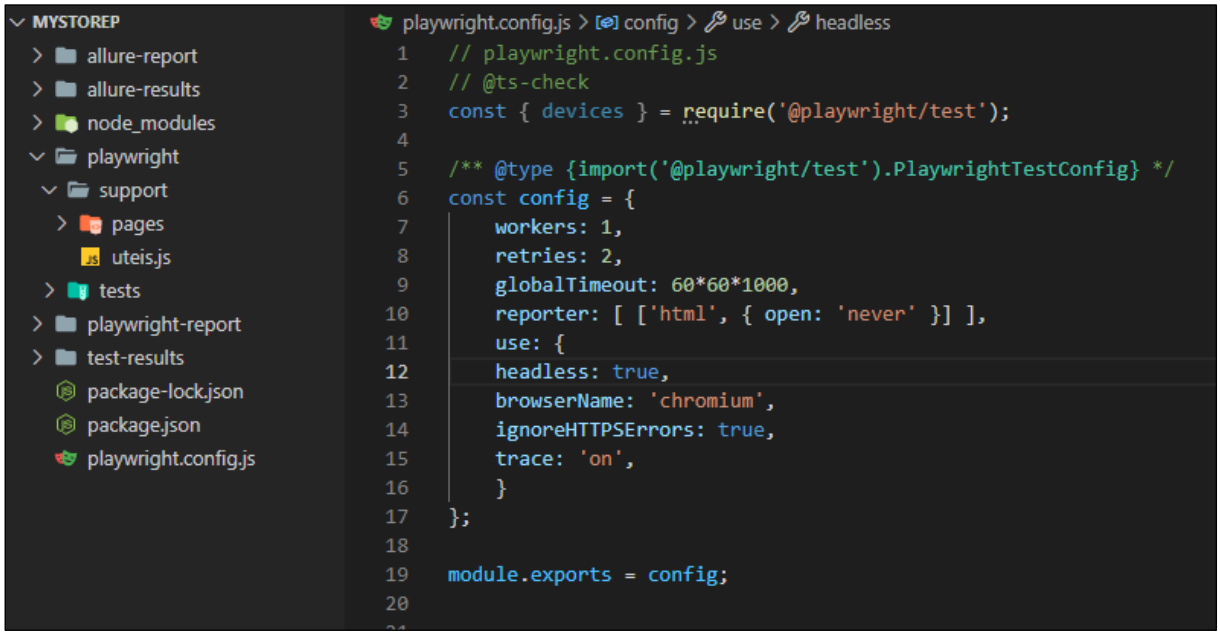
5.1.1 Preparação do ambiente

Para a automatização do My Store foi utilizado o Playwright, que não possui as limitações citadas anteriormente. A estrutura montada para o Playwright está descrita na figura 18, que possui os seguintes componentes:

- Pastas do *Allure*: pastas dos resultados captados pelo *Allure*, assim como a do *report*;
- Node *modules*: contém as configurações do node;
- *Support* > *Pages*: conjunto das páginas no padrão *PageObjects*;
- *Support* > uteis: contém funções em JavaScript que serão utilizadas nos scripts de testes;
- *Tests*: contém os scripts dos testes que foram criados a partir dos casos de testes criados para o site que possuem o formato *spec.js*;
- *Playwright-report*: contém o relatório próprio do Playwright;
- *Test-results*: contém os resultados dos testes separados em pastas;
- *Package-lock.json*: contém configurações do node;
- *Package.json*: configurações do projeto;

- *Playwright.config.js*: configurações do *Playwright*.

Figura 18 - Interface do projeto *My Store* com as configurações do *Playwright*.



Fonte: Autoria própria.

Seguindo o padrão *PageObject*, o site foi dividido em páginas para que fiquem mais organizados e mais fáceis de manter os testes. Para cada pasta de cada página foi criada um arquivo *index.js* em que se localiza uma classe contendo os elementos das páginas e as funções em que serão realizadas as ações dos testes.

A figura 19 apresenta um exemplo de uma das páginas, contendo os elementos e as funções que realizam alguma ação. As funções utilizadas no *Playwright* devem ser assíncronas. Isso quer dizer que assim que ela executa um *Promisse* – um objeto que representa um valor em uma atividade assíncrona – deve ser retornada. Por isso devem ser utilizados os termos *async/await* (MDN WEB DOCS, 2022).

Todos os elementos da página, conforme a figura 19, são declarados no construtor, o parâmetro *page* representa a página da janela do navegador que irá interagir com o conteúdo da página. Por isso ela deve ser referenciada toda vez em

que terá uma interação com um item da página. Os outros objetos declarados são os elementos da página que serão utilizados pelas funções.

Figura 19 - Parte da estrutura da classe MeusEnderecos que segue o padrão *PageObjects*.

```
const { expect } = require('@playwright/test');

class MeusEnderecos {
    /**
     * @param {import('playwright').Page} page
     */
    constructor(page) {
        this.page = page;
        this.novoEndereco = page.locator('text=Add a new address');
        this.endereco = page.locator('input[name="address1"]');
        this.cidade = page.locator('input[name="city"]');
        this.estado = page.locator('select[name="id_state"]');
        this.cep = page.locator('input[name="postcode"]');
        this.celular = page.locator('input[name="phone_mobile"]');
        this.botaoSalvar = page.locator('button:has-text("Save")');
        this.alias = page.locator('input[name="alias"]');
        this.abaEndereco = page.locator('h1:has-text("My addresses")');
        this.enderecoJaCadastrado = page.locator('text=Meu novo endereco').nth(1);
        this.botaoAtualizar = page.locator('span:has-text("Update")').nth(4);
    }

    async clicarAdicionarNovoEndereco() {
        await this.novoEndereco.click();
    }

    async informarEndereco(text) {
        await this.endereco.type('');
        await this.endereco.click();
        await this.endereco.type(text);
    }
```

Fonte: Autoria própria.

Para que os scripts realizarem os passos descritos nos casos de testes foi necessário o uso de algumas expressões, como asserções, bibliotecas, comandos e *matchers.* As asserções são as combinações de bibliotecas, comandos, *matchers* que realizam uma comparação entre o que foi recebido e o que foi obtido no teste. Elas determinam se um teste está aprovado ou não. Antes de citá-los, é necessário comentar o conceito de DOM, ou *Document Object Model* - Modelo de Objeto de Documentos em português. DOM é uma interface que contém documentos HTML, que podem ser manipulados através de linguagens de programação (MALDONADO, 2018). As expressões estão listadas no quadro 1.

Quadro 1 - Lista das expressões utilizadas para os testes automatizados no Playwright.

locator()	É um localizador utilizado para encontrar elementos na página.
expect()	É uma biblioteca que permite o uso de várias expressões em conjunto.
toHaveText()	Um dos *matchers* contidos na biblioteca *expect*, permite a asserção de elementos com texto.
toBeVisible()	Um dos *matchers* contidos na biblioteca *expect*, confere se o elemento está visível no DOM.
click()	Simula o *click* do mouse em uma página.
type()	Escreve em um campo de texto caractere por caractere.
selectOption()	Seleciona uma das opções em que o elemento é selecionável.
fill()	Escreve em um campo de texto.
check()	Assinala um elemento.
goto()	Navega até a url atribuída.
waitForLoadState()	Aguarda pelo estado de carga.

Fonte: Autoria própria.

A figura 20 exibe um exemplo de um arquivo *spec.js*, que são os arquivos de teste. Para a execução dos testes ser possível, as funções criadas nas páginas POM são referenciadas neste arquivo. A intenção é deixar o arquivo o mais simples e fácil de ler. Para ter acesso as *pages* criadas é necessário importar os módulos das classes e criar objetos para poder utilizar as funções da classe.

A estrutura dos testes é própria do *Playwright*, tendo o uso do termo *test*, que permite adicionar uma descrição ao *script*.

Figura 20 - Estrutura de um arquivo *spec.js*.

```
const { test } = require('@playwright/test');
const { Autenticacao } = require('../support/pages/Autenticacao/index');
const { MinhaConta } = require('../support/pages/MinhaConta/index');


test('Autenticar como cliente que já possui conta', async ({ page }) => {

    let autenticacao = new Autenticacao(page);

    //Dado que um cliente não autenticado acessou o site da loja "My Store"
    await autenticacao.visitarMyStore();
    await autenticacao.irParaPaginaDeLogin();

    //Quando ele inserir suas credenciais
    const email = 'giulianni@yopmail.com',
        senha = '123456';

    await autenticacao.inserirEmail(email);
    await autenticacao.informarSenha(senha);
    await autenticacao.clicarEmEntrar();

    //Então será redirecionado para a página "Minha conta"
    let minhaconta = new MinhaConta(page);
    await minhaconta.verificarSeEstaEmMinhaConta();
});
```

Fonte: Autoria Própria.

5.1.2 Avaliação dos resultados

A principal comparação feita será em relação do tempo de execução, tanto dos testes automatizados quanto dos manuais.

O tempo que levou para a preparação dos ambientes para realizar os testes também é considerado. Para a configuração do *Playwright* e preparação da ordenação do código no padrão *PageObjects* levaram aproximadamente 5 horas para que fosse possível iniciar a automatização os casos de testes corretamente. Esse tempo foi necessário para que se construa uma curva de conhecimento para o preparo do ambiente para as automatizações.

Mas para automatizar todos os casos de testes levaram aproximadamente 15 horas, além de garantir que todos estejam funcionando corretamente ao serem testados individualmente.

Para a execução de todos os scripts foi necessário adicionar algumas configurações a mais ao arquivo de configuração do *Playwright*, conforme está na figura 21. As configurações foram:

- Número de *workers*: são as quantidades de janelas de navegadores que serão abertas ao executar os scripts em sequência;
- *Retries*: quantidade de tentativas que o mesmo teste será executado novamente ao falhar;
- *globalTimeout*: tempo total do timeout dos testes;
- *reporter*: relatório padrão do *Playwright*;
- *headless*: os testes serão executados com navegador em segundo plano;
- *browserName*: nome do navegador utilizado;
- *ignoreHTTPSErrors*: ignorar os erros de segurança do navegador ao acessar o site;
- *trace*: uma forma de verificar o passo a passo da execução dos testes através do relatório padrão do *Playwright*;

Figura 21 – Interface do arquivo *playwright.config.js*.

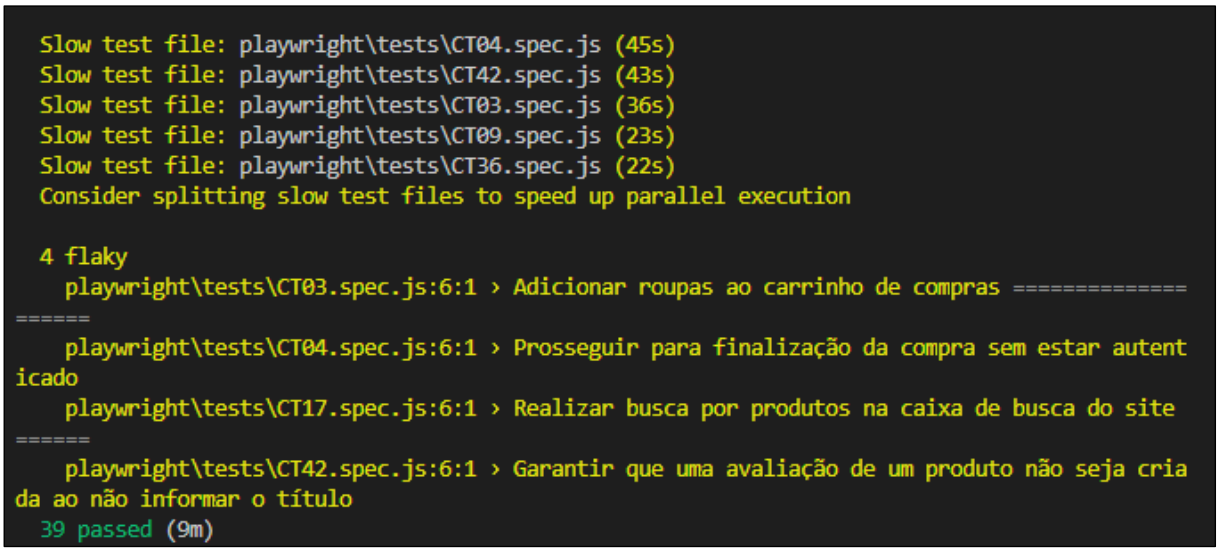
```
// playwright.config.js
// @ts-check
const { devices } = require('@playwright/test');

/** @type {import('@playwright/test').PlaywrightTestConfig} */
const config = {
    workers: 1,
    retries: 2,
    globalTimeout: 60*60*1000,
    reporter: [ ['html', { open: 'never' }] ],
    use: {
    headless: true,
    browserName: 'chromium',
    ignoreHTTPSErrors: true,
    trace: 'on',
    }
};

module.exports = config;
```

Fonte: Autoria própria.

Ao analisar os resultados dos testes na figura 22, os 46 scripts de teste demoraram 9 minutos para serem executados. No total 39 testes passaram sem

apresentarem problemas, mas 4 testes tiveram que ser reexecutados por mais de uma vez, então ficaram com a nomenclatura *flaky*.

Figura 22 - Resultados dos *scripts* de teste executados.



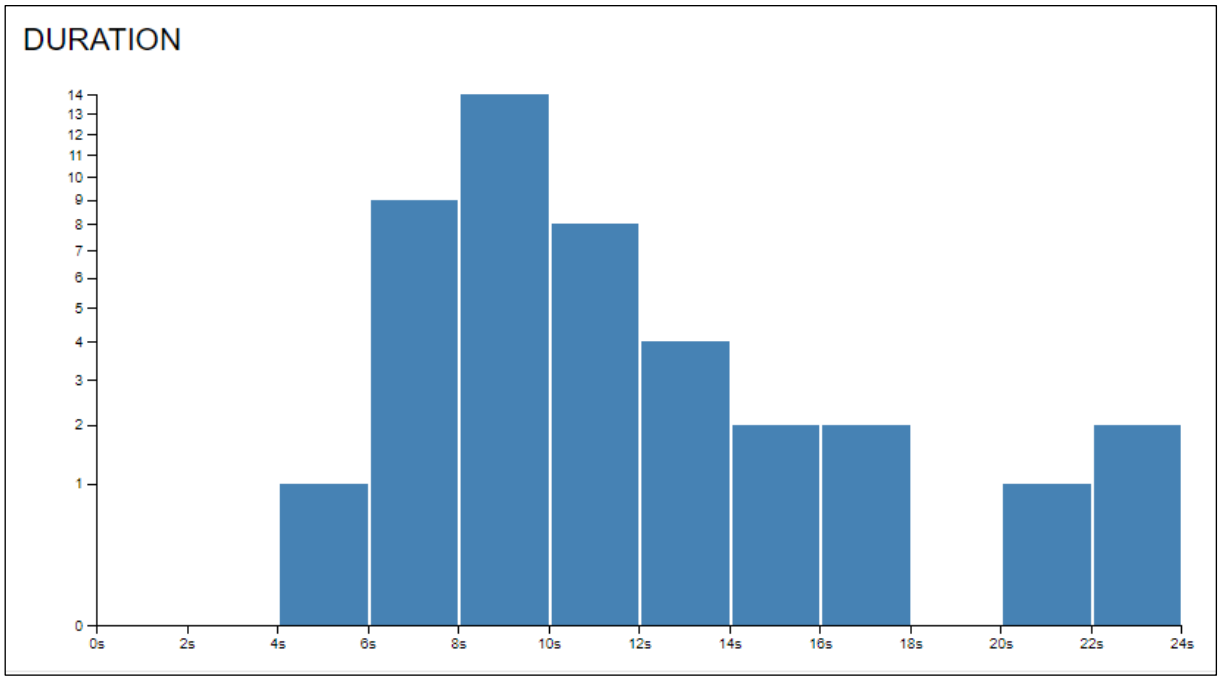
Fonte: Autoria própria.

O relatório gerado no *Allure Reporter* é possível analisar o tempo de cada teste baseado em gráficos. A partir das figuras 23, 24 e 25 é possível concluir o seguinte:

- 100% dos testes foram executados com sucesso;
- Em relação a duração é constatado que:
  - 14 casos de testes foram executados entre 8s e 10s, a maior quantidade de testes foi executada durante essa faixa de tempo;
  - 1 caso de teste foi executado entre 4s e 6s;
  - 9 casos de testes foram executados entre 6s e 8s;
  - 8 casos de testes foram executados entre 10s e 12s;
  - 4 casos de testes foram executados entre 12s e 14s;
  - 2 casos de testes foram executados entre 14s e 16s;
  - 2 casos de testes foram executados entre 16s e 18s;
  - 1 caso de teste foi executado entre 20s e 22s;
  - 2 casos de testes foram executados entre 22s e 24s;

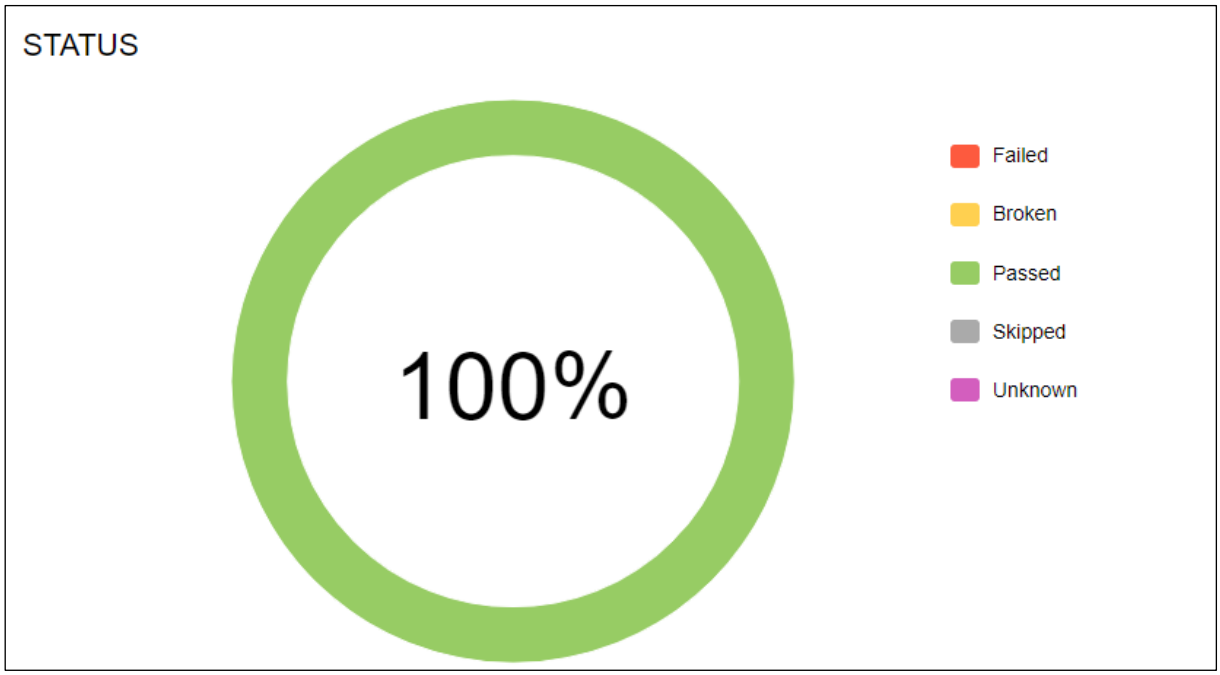
- A severidade de todos os testes executados foi classificada como normal.

Figura 23 - Gráfico gerado no Allure Framework referente a duração de execução dos testes do My Store.



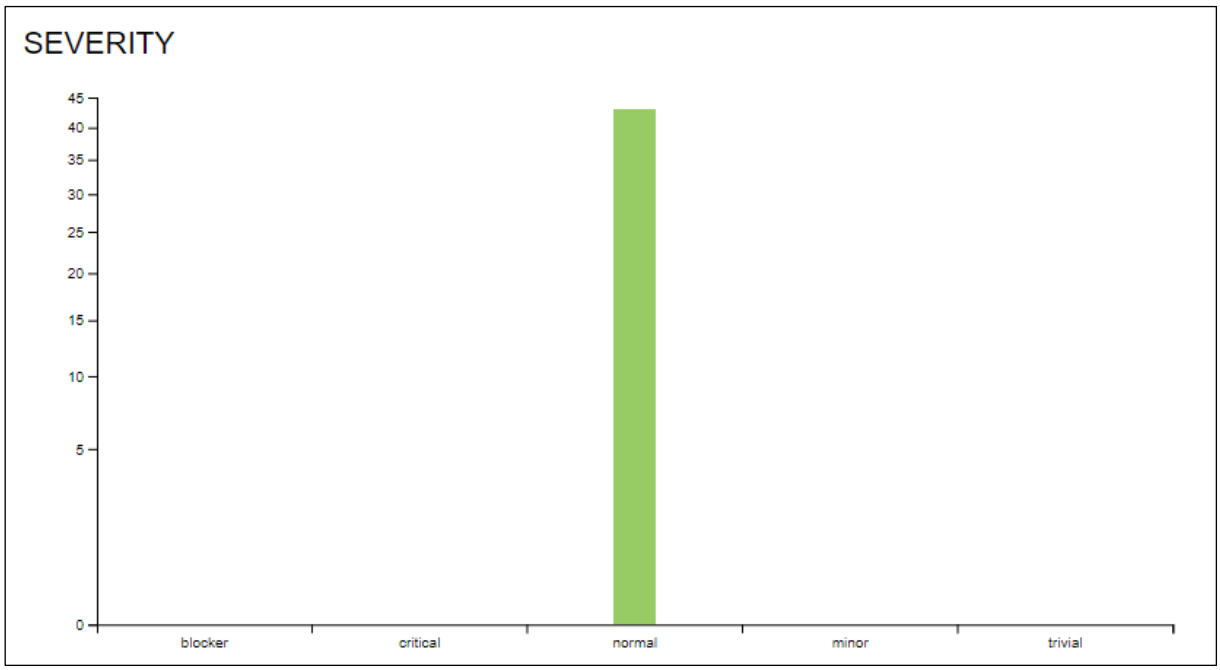
Fonte: Autoria própria.

Figura 24 - Gráfico gerado no Allure Framework referente ao status dos testes executados do My Store.



Fonte: Autoria própria.

Figura 25 - Gráfico gerado no Allure Framework referente a severidade dos testes executados do My Store.



Fonte: Autoria própria.

Em relação aos testes manuais, todos os casos de testes foram executados. Alguns polparam a repetição de passos já feitos anteriormente pelo caso de teste ser da mesma funcionalidade em que o próximo será executado. No total todos foram feitos em 40 minutos, mas outros critérios a serem considerados são a experiencia do testador responsável e o conhecimento que ele já tem a aplicação. O que diminui consideravelmente o tempo se for comparado com um testador que não conhece bem o produto, o que gera alguns enganos em funções distintas do site ao testar.

O site *My Store* é muito simples de navegar, seus casos de teste não são complexos. Mas através do tempo coletado é constatado que a execução do teste manual demora 4 vezes o tempo de execução dos testes automatizados. Apesar do longo tempo para criação e preparação do ambiente, a longo prazo se tem uma economia de tempo e esforço por parte do analista de testes responsável. Não sendo necessário o retrabalho de refazer a tarefa repetitiva de efetuar todos os casos de teste.

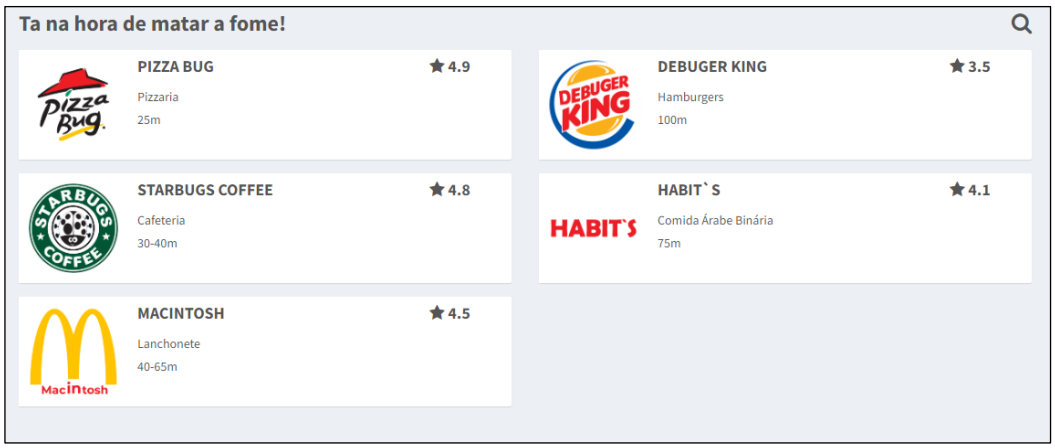
Vale ressaltar o fato de a aplicação ser pequena e possuir inicialmente apenas 43 casos de teste. Num produto real a diferença no ganho de tempo e economia é ainda mais discrepante.

5.2 Website Parodifood

O *Parodifood*, assim como o nome faz alusão, é um site paródia do Ifood. Ele foi desenvolvido pela *QAninja Academy*, que fornece cursos para analistas de testes (PARODIFOOD, 2022). Esse profissional também é chamado de QA, diminutivo do inglês de *Quality Assurance* e traduzido para o português como Garantia da Qualidade.

A figura 26 demonstra o fato de o site ser uma paródia do aplicativo de entrega de comida *Ifood*. Porém os restaurantes remetem a itens relacionados a tecnologia da informação.

Figura 26 - Menu de restaurantes do *Parodifood*.



Fonte: Parodifood, 2022.

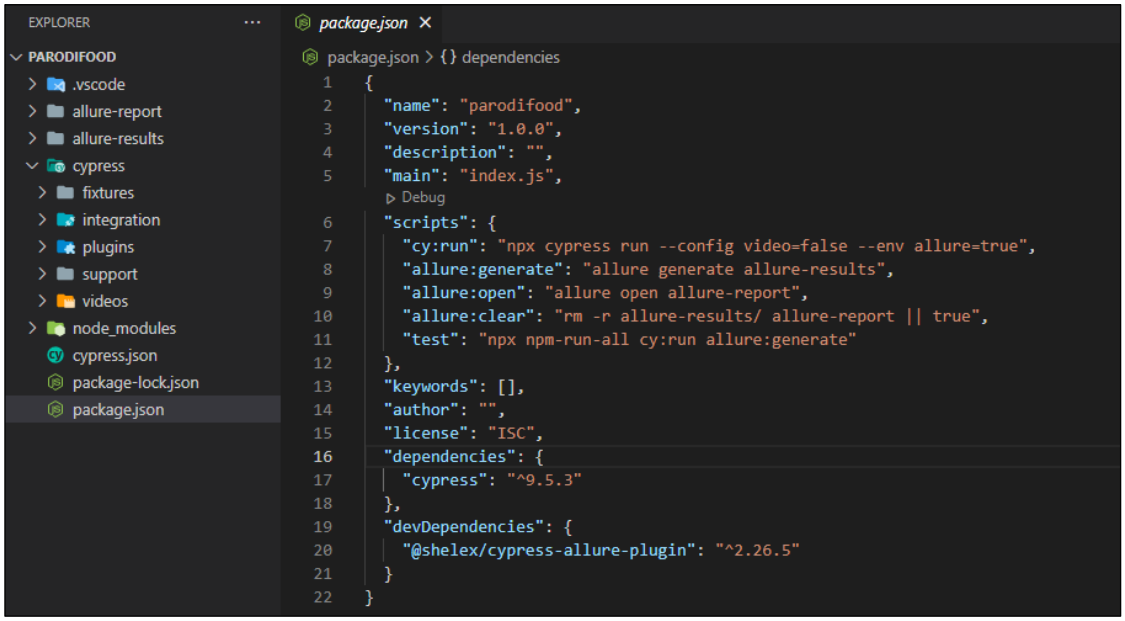
Para a construção dos scripts de testes foi utilizado o *Cypress*. Como o *Parodifood* não apresenta os fatores limitantes do *Cypress*, foi possível realizar a automatização sem empecilhos.

5.2.1 Preparação do ambiente

O projeto do *Parodifood*, como demonstrado na figura 27, ficou estruturado da seguinte forma:

- Pastas do *Allure*: pastas dos resultados captados pelo *Allure*, assim como a do *report*;
- *.vscode*: contém um arquivo do formato *.json* com extensões de ícones;
- *Fixtures*: pasta está vazia, porém originalmente deveria conter os arquivos de *fixtures* do Cypress que são arquivos com formato *.json* que permitem o compartilhamento de dados entre os testes;
- *Integration*: os arquivos de teste no formado *spec.js* estão nesta pasta;
- *Plugins*: contém os plugins instalados no *Cypress* em um arquivo;
- *Support*: contém os arquivos das páginas e comandos dando suporte aos testes;
- Vídeos: pasta está vazia, mas é usada para guardar os vídeos retirados de testes;
- Node *modules*: contém as configurações do node;
- *Cypress.json*: arquivo de configurações próprio do *Cypress*;
- *Package-lock.json* e *Package.json*: arquivo que contém configurações do projeto;

Figura 27 - Interface do projeto do *Parodifood*.



Fonte: Autoria Própria.

Assim como no *Playwright*, o projeto do *Cypress* também foi organizado no padrão *PageObjects*. Mas diferente do *Playwright* em que é usado apenas um arquivo por página, no *Cypress* ele é dividido em dois.

O arquivo *index.js* contém a classe da página com todos as funções para o teste enquanto o arquivo *element.js* são declarados os elementos da página. Para conexão entre os dois arquivos é necessário exportar os elementos para que sejam referenciados nas funções, como ilustrado nas figuras 28 e 29.

Para nomear as funções, foram utilizados verbos no infinitivo. Além de iniciar com um verbo, o nome da função é autoexplicativo sendo essas características de boas práticas de programação (SCHULTS, 2016).

Figura 28 – Parte do arquivo *elements.js* da página Pagamento.

```
export const ELEMENTS = {
    nome: ':nth-child(2) > .col-sm-6 > mt-input-container > .form-group > .form-control',
    email: ':nth-child(2) > :nth-child(4) > mt-input-container > .form-group > .form-control',
    confirmacaoEmail: ':nth-child(5) > mt-input-container > .form-group > .form-control',
    endereco: ':nth-child(3) > .col-sm-6 > mt-input-container > .form-group > .form-control',
    numero: ':nth-child(3) > :nth-child(3) > mt-input-container > .form-group > .form-control',
    complemento: ':nth-child(3) > :nth-child(4) > mt-input-container > .form-group > .form-control',
```

Fonte: Autoria própria.

Figura 29 – Parte do arquivo *index.js* da página Pagamento.

```
const el = require('./elements').ELEMENTS;

class Pagamento {

    preencherNome(text){
        cy.get(el.nome).type(text).should('have.value', text);
        cy.wait(1000);
    }
```

Fonte: Autoria própria.

Como no *Playwright*, o *Cypress* também utiliza de expressões. Essas expressões são combinadas com o uso de comandos, bibliotecas e matchers. Eles em conjunto formam asserções que tem como objetivo comparar o valor esperado e o recebido durante o teste. No cypress as asserções se iniciam com a expressão "*cy*". As seguintes expressões foram utilizadas nos arquivos *index.js*:

Quadro 2 - Lista das expressões utilizadas para os testes automatizados no Cypress.

get()	Obtém elementos do DOM.
wait()	Adiciona um tempo de espera ou habilita a espera de um elemento.
route()	Realiza solicitações relacionadas a rede.
type()	Escreve em um campo de texto.
click()	Simula o click do mouse em uma página.
should()	Permite o uso de várias asserções em conjunto.
visit()	Realiza a navegação até a url atribuída.
as()	Cria um alias a algum elemento que pode ser referenciado em outra parte do

	código.
expect()	Permite o uso de asserções do BDD.
contains()	Confirma se o elemento contém o parâmetro repassado.
eq()	Verifica se o elemento possuí o valor equivalente ao seu parâmetro.

Fonte: Autoria Própria.

A figura 30 demonstra que os arquivos dos testes têm uma estrutura similar as do *Playwright*. Porém a estrutura dos testes aqui são as do Mocha contendo o termo *describe*, que permite atribuir uma descrição ao teste. Dentro dele todos os passos do teste são subdivididos em outras descrições chamadas *it*. As páginas POM relacionadas ao teste também devem ser exportadas para que as funções da classe sejam utilizadas.

Por causa da estrutura *PageObjects*, o código se torna mais fácil de ler e compreender quais são os passos do teste.

Figura 30 - Estrutura de um script de teste do *Cypress*.

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

import Restaurants from '../support/pages/Restaurantes';

describe('Verificar lista de restaurantes', () => {

    it('Dado que o cliente está na página inicial do Parodifood', () => {
        Restaurants.acessarParodifood();
    });

    it('Quando a aba dos restaurantes for acessada', () => {
        Restaurants.acessarRestaurantes();
    });

    it('Então uma lista contendo os restaurantes disponíveis deverá ser exibida', () => {
        Restaurants.conferirOTituloDoMenu();
    });
});
```

Fonte: Autoria Própria.

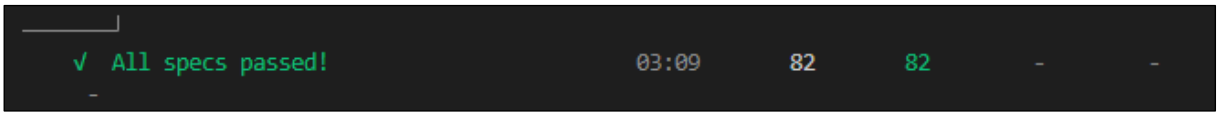
5.2.2 Avaliação dos resultados

Assim como no projeto *My Store*, está sendo avaliado o fator tempo de execução dos testes automatizados e dos manuais.

O tempo que foi necessário para iniciar a configuração do *Cypress* até obter uma curva de aprendizado o suficiente para dar início a automatização dos casos de uso foi de 6 horas. Mas o tempo utilizado para automatizar todos os 20 casos de teste foi de 8 horas. Sendo assim a preparação do ambiente foi realizada em 14 horas no total.

A figura 31 demonstra o tempo total em que os testes automatizados levaram para ser executados. O total foi de 03 minutos e 09 segundos, sendo 82 a quantidade de testes que foram executados. Vale ressaltar que o *Cypress* considera cada estrutura *it* como um caso de teste diferente.

Figura 31 - Resultado da execução dos testes no *Cypress*.

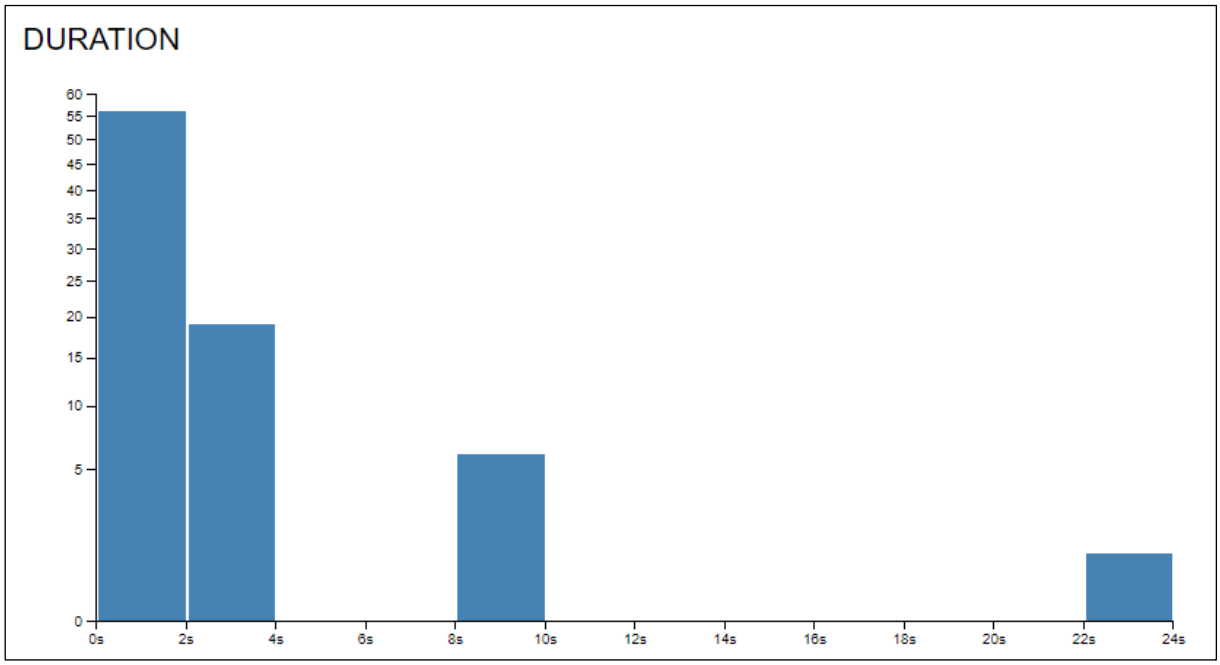


Fonte: Autoria Própria.

A partir do relatório do *Allure Report*, nas figuras 32, 33 e 34, é possível concluir algumas evidências:

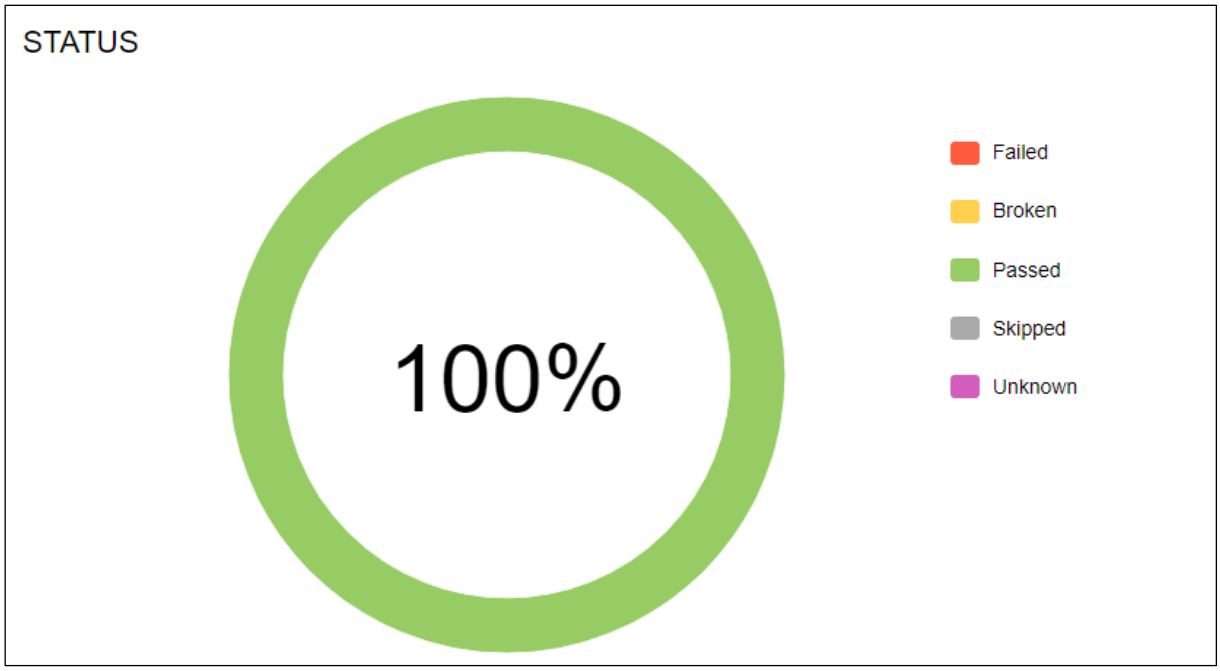
- 100% dos testes passaram;
- A duração de tempo da execução variou entre 2s e 24s;
    - 56 testes levaram até 2s para serem executados;
    - Apenas 1 teste levou entre 22s e 24s para ser executado;
- A severidade de todos os testes executados foi classificada como normal.

Figura 32 - Gráfico gerado no Allure Framework referente a duração de execução dos testes do Parodifood.



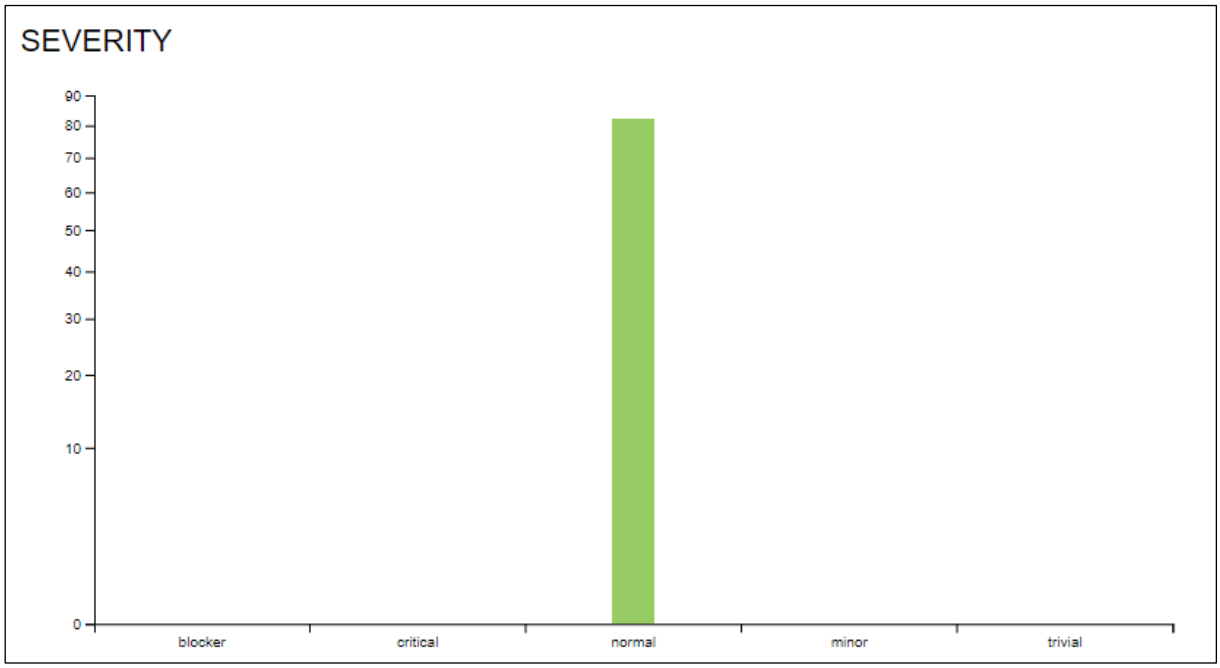
Fonte: Autoria própria.

Figura 33 - Gráfico gerado no Allure Framework referente ao status dos testes executados do Parodifood.



Fonte: Autoria própria.

Figura 34 - Gráfico gerado no Allure Framework referente a severidade dos testes executados do Parodifood.



Fonte: Autoria própria.

Os 20 casos de teste levaram, de forma manual, 20 minutos para serem executados. O *Parodifood* é um site com muito menos funções se comparado com o *My Store*. Porém ele contém muitos casos de uso de preenchimento de formulários com dados, o que torna seus testes manuais ainda mais repetitivos.

O tempo de execução entre os testes automatizados e os manuais se tornam ainda mais discrepantes comparados com o do *My Store*, aqui o ganho seria mais que 6 vezes mais ao utilizar os testes automatizados. Isso desconsiderando o tempo de preparo do ambiente.

Porém se for considerado também o tempo de preparo de ambiente, ao longo do tempo o teste automatizado também vem com um benefício de evitar que o analista de testes faça muitos passos repetitivos, já que o preparo do ambiente só ocorre uma vez. O que pode ocorrer futuramente é a manutenção do código e automatização de novas *features*.

6 CONCLUSÃO

Este trabalho abordou a utilização de testes de aceitação em diferentes aplicações web ao utilizar os casos de teste como critérios de aceitação e o uso de ferramentas de testes de ponta a ponta para simular o comportamento de um usuário real.

Apesar das limitações que ocorreram com o uso do *Cypress* no site *My Store*, ele pôde ser aplicado no site *Parodifood* e seus *scripts* de testes foram executados sem problemas em um curto período. O que prova a discussão teórica sobre os testes automatizados e a sua curva de tempo de execução ao serem comparados com os testes manuais.

O mesmo pode ser afirmado sobre os resultados da execução no site *My Store* através da ferramenta *Playwright*. Apesar da quantidade de vezes que o teste teve que ser realizado novamente, o ponto discutido entre a diferença de tempo de execução de testes automatizados x manuais também foi comprovada. Em um contexto de construção de um software, quanto mais tempo se economiza, menos dinheiro se perde.

Os *frameworks* utilizados apresentaram os resultados esperados nos dois casos. Demonstrando que eles podem ser utilizados em aplicações diferentes dependendo das tecnologias em que elas foram construídas. Mesmo que elas sejam similares a instalação e terem configurações em comum. Cabendo a comunidade acadêmica e aos analistas de testes, interessados no tema deste trabalho, escolher qual ferramenta seria aplicável a um projeto.

A construção deste trabalho proporcionou aprendizado em conceitos de testes e qualidade de software e em automatização de testes durante as dificuldades em relação as limitações das ferramentas e mapeamento do site. Também foi realizada a construção Casos de Uso, trazendo conhecimentos novos a respeito de formas tradicionais de documentações de teste.

Para trabalhos futuros, sugere-se os seguintes temas:

- Aplicação de testes não funcionais em aplicações web, como teste de carga, segurança e *stress*;

- Estudo dos casos de uso das aplicações utilizadas neste trabalho.

REFERÊNCIAS

ALMEIDA, Guilherme A. M. de. **Fatores de escolha entre metodologias de desenvolvimento de software tradicionais e ágeis**. 2017. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Produção, São Paulo.

ALLURE FRAMEWORK (org.). **About.** Disponível em: <https://docs.qameta.io/allure/>. Acesso em 23 abr. 2022.

ANDRADE, Bruno E. et al. **TDD-Test Driven Development**. Docplayer, 2011. Disponível em: < https://docplayer.com.br/15609123-Tdd-test-driven-development.html>. Acesso em: 20 out. 2021.

ASSIS, Daniel M. de. **Dominando os tipos de testes automatizados**. Devmedia, 2016. Disponível em: <https://www.devmedia.com.br/dominando-os-tipos-de-testes-automatizados/33867>. Acesso em: 01 nov. 2021.

BARTIE, Alexandre. **Garantia da qualidade de software**. Rio de Janeiro: Campus, 2002.

BECK, Kent. et al. **Manifesto para Desenvolvimento Ágil de Software**. Agile Manifesto, 2001. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 19 de out. de 2021.

BECK, Kent. **TDD Desenvolvimento Guiado por Teste.** São Paulo: Bookman, 2010.

BRAZ, Alan. **Introdução ao Scrum**. Alan Braz, 2011. Disponível em: <http://www.alanbraz.com.br/ic/scrum.pdf>. Acesso em: 27 de nov. de 2021.

CAMPOS, Camila. **A pirâmide de testes**. 06 fev. 2019. Disponível em: < https://medium.com/creditas-tech/a-pir%C3%A2mide-de-testes-a0faec465cc2>. Acesso em: 30 out. 2021.

CARVALHO, Ingrid. **Você sabe o que é Teste Caixa Branca e Teste Caixa Preta?** 01 apr. 2019. Disponível em: <https://medium.com/@ingrid.carvalho.mo/voc%C3%AA-sabe-o-que-%C3%A9-teste-caixa-branca-e-teste-caixa-preta-9a2d08fe9d0c>. Acesso em: 22 apr. 2022.

CHAI (org.). **Chai Assertion Library**. Disponível em: < https://www.chaijs.com/>. Acesso em: 23 abr. 2022.

COHN, Mike. **Succeeding with agile**: Software Development Using Scrum. Boston: Addison Wesley, 2010.

CSTE. **Guide to the CSTE common body of knowledge**. Quality Assurance Institute. 2006.

CUCUMBER (org.). **Indroduction**. Disponível em: < https://cucumber.io/docs/guides/overview/>. Acesso em: 01 nov. 2021.

CYPRESS (org). **Why Cypress?** Disponível em: < https://docs.cypress.io/guides/overview/why-cypress#In-a-nutshell>. Acesso em: 01 nov. 2021.

DELAMARO, Marcio E.; MALDONADO, José C.; JINO, Mario. (2007). **Introdução ao Teste de Software**. Rio de Janeiro: Elsevier, 2007.

HASIJA, Pryanka. **Garantia de qualidade no Scrum: muito além dos testes**. 31 out. 2012. Disponível em: < https://www.infoq.com/br/articles/experience-qa-scrum/>. Acesso em: 21 maio 2022.

FOWLER, Martin. **PageObject**. 10 set. 2013. Disponível em: < https://martinfowler.com/bliki/PageObject.html>. Acesso em: 02 nov. 2021.

GARTNER, Markus. **ATDD by Example A Practical Guide to Acceptance Test-Driven Development**. Boston: Pearson Education Inc, 2013.

HILLMAN, Mônica M. **Como resolver um erro de Cross-Origin Resource Sharing (CORS)?** Alura, 20 jan. 2022. Disponível em: < https://www.alura.com.br/artigos/como-resolver-erro-de-cross-origin-resource-sharing>. Acesso em 23 abr. 2022.

ISO/IEC/IEEE. **ISO/IEC/IEEE 24765**: Systems and software engineering – Vocabulary, 2. Ed. 2017.

ISTQB. **Certified Tester Foundation Level Syllabus**. BSTQB, 2018. Disponível em: <https://bstqb.org.br/b9/doc/syllabus_ctfl_2018br.pdf>. Acesso em: 18 ago. 2021.

KOLB, Juliana K. **Extreme Programming (XP).** 12 set. 2013. Disponível em: < https://julianakolb.wordpress.com/2013/12/19/extreme-programming-xp/>. Acesso em: 22 abr. 2022.

LISBOA, Alveni. **Chromium vs. Chrome: qual é a diferença entre os dois navegadores?** Canaltch, 12 nov. 2021. Disponível em: <https://canaltech.com.br/apps/qual-a-diferenca-entre-chromium-chrome-192599/>. Acesso em: 22 abr. 2022.

MACHADO, Otavio**. Ferramentas de testes de aceitação**: uma odisseia. 09 out. 2017. Disponível em: <https://medium.com/@otavio/ferramentas-de-testes-de-aceita%C3%A7%C3%A3o-uma-odiss%C3%A9ia-8eff4c96da9e>. Acesso em: 01 nov. 2021.

MAGALHAES, Andre L. **O que é CAPTCHA e reCAPTCHA?** Canaltech, 2021. Disponível em: <https://canaltech.com.br/internet/o-que-e-captcha-recaptcha/>. Acesso em: 01 nov. 2021.

MALDONADO, Leonardo. **Entendendo o DOM (Document Object Model).** Tableless, 2018. Disponível em: < https://tableless.com.br/entendendo-o-dom-document-object-model/>. Acesso em 05 maio 2022.

MARINHEIRO, Vitor. **Cypress + Page Object = Sucesso**. 12 nov. 2020. Disponível em: <https://vitormarinheiroautomation.medium.com/cypress-page-object-sucesso-6841cb7c19a0>. Acesso em: 01 nov. 2021.

MDN WEB DOCS. **Funções assíncronas**. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/async_function>. Acesso em: 01 maio 2022.

MDN WEB DOCS. **Promise**. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise>. Acesso em: 01 maio 2022.

MOCHA (org.). **Mocha**. Disponível em: < https://mochajs.org/>. Acesso em: 23 abr. 2022.

MUNIZ, Antônio. et al. **Jornada Ágil de Qualidade**: Aplique práticas no início do ciclo para potencializar a implementação contínua de software com qualidade. Rio de Janeiro: Brasport, 2019.

MYERS, Glenford, J.; BADGETT, Tom; SANDLER, Corey. **The Art of Software Testing**, 3. ed. New Jersey: Wiley Publishing, 2012.

MY STORE. **My Store**. Disponível em: < http://automationpractice.com/index.php>. Acesso em: 10 mar. 2022.

NODE (org.). **About Node.Js**. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 02 nov. 2021.

NOLETO, Cairo. **Aplicações web**: entenda o que são e como funcionam! 18 mar. 2020. Disponível em: < https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/>. Acesso em: 02 nov. 2021.

NOLETO, Cairo. **Framework**: o que é, como ele funciona e para que serve? 13 fev. 2020. Disponível em: < https://blog.betrybe.com/framework-de-programacao/o-que-e-framework/>. Acesso em: 23 abr. 2022.

PAULA FILHO, Wilson de Pádua. **Engenharia de software: fundamentos, métodos e padrões**. 2ª edição. Rio de Janeiro: LTC, 2003.

PHP TRAVELS (org.). **Travel Technology Partner**. Disponível em: <https://phptravels.com/demo>. Acesso em 10 mar. 2022.

PONTES, Thiago B.; ARTHAUD, Daniel D. B. **Metodologias Ágeis para o Desenvolvimento de Softwares**. Ciência e Sustentabilidade, v. 4, n. 2, p. 173-213, 14 mar. 2019.

PLAYWRIGHT (org.) **Getting started**. Disponível em: < https://playwright.dev/>. Acesso em: 23 abr. 2022.

PRESSMAN, Roger S. **Engenharia de Software**: Uma Abordagem Profissional. 7ª Edição, McGraw-Hill. 780 p., 2011.

QANINJA ACADEMY (org). **Parodifood**. Disponível em: < https://parodifood.herokuapp.com/>. Acesso em: 10 mar. 2022.

RIBEIRO, Rafael D.; RIBEIRO, Horácio C. S. **Métodos ágeis em gerenciamento de projetos**. 1 ed. Rio de Janeiro: SPIN, 2015.

SCHULTS, Carlos. **Dez dicas para lhe ajudar a escolher bons nomes.** 22 maio 2016. Disponível em < https://carlosschults.net/pt/como-escolher-bons-nomes/>. Acesso em: 02 maio 2022.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide**. The Definitive Guide to Scrum: The Rules of the Game. Scrum Guides, 2020. Disponível em: < https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100>. Acesso em: 23 out. 2021.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2013.

SOMMERVILLE, Ian. **Engenharia de Software**. 10ª ed. São Paulo: Pearson, 2019.

SUTHERLAND, Jeff. **Scrum A Arte De Fazer O Dobro Do Trabalho Na Metade Do Tempo**. São Paulo: LeYa, 2014.

SWEBOK. **Guide to the Software Engineering Body of Knowledge**. 2004. Disponível em: <https://www.computer.org/education/bodies-of-knowledge/software-engineering>. Acesso em: 20 mar 2022.

VITORIANO, Dan. **Visual Studio Code**. 29 mar. 2020. Disponível em: < https://danvitoriano.medium.com/visual-studio-code-fdaf5aef736e>. Acesso em: 02 nov. 2021.

WAZLAWICK, Raul S. **Engenharia de software**: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

WYNNE, Matt; HELLESOY, Aslak. **The cucumber book**: behaviour-driven development for testers and developers. Pragmatic Bookshelf, 2012

ZARELLI, Guilherme B. **Pirâmide de Testes** — Definindo uma boa suíte de testes para seu Software. 31 ago. 2020. Disponível em: < https://medium.com/luizalabs/pir%C3%A2mide-de-testes-definindo-uma-boa-su%C3%ADte-de-testes-para-seu-software-a6864886f29b>. Acesso em: 30 out. 2020.

APÊNDICE A – CASOS DE TESTE DO WEBSITE MY STORE

Caso de teste número 1	
Título:	Realizar login com uma conta nova.
Cenário de teste:	@CNT001
Ator:	Cliente.
Objetivo:	Realizar o login logo após criar uma conta.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente não deve possuir uma conta; 3 – O cliente deve estar na página de autenticação; 4 – O cliente deve possuir um e-mail válido.
Entradas:	1 – Na aba “criar uma conta” informe o seguinte e-mail “cliente.novo@yopmail.com”. 2 – Clique no botão “Criar uma conta”. 3 – No formulário insira as informações abaixo: • Primeiro nome: Mariana • Sobrenome: Silva • Senha: teste • Endereço: Silver leaf, 10 • Cidade: Alabaster • Estado: Alabama • Cep: 35007 • Celular: 55555555 4 – Clique no botão “Registrar”.
Resultado esperado:	Após prosseguir com a criação da conta, a autenticação deve redirecionar para a página “Minha conta”.
Data e hora da execução:	22/03/2022 20:30
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Após a criação da conta, o sistema realiza a autenticação redirecionando para a página “minha conta”.

Caso de teste número 2	
Título:	Autenticar como cliente que já possui conta.

Cenário de teste:	@CNT002
Ator:	Cliente.
Objetivo:	Autenticar como um usuário que já possui conta.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar na página de autenticação.
Entradas:	1 – Na aba “anteriormente registrado?” informe as seguintes credenciais: • E-mail: giulianni@yopmail.com • Senha: 123456 2 – Clique no botão “Entrar”.
Resultado esperado:	Após a autenticação deve redirecionar para a página “Minha conta” contendo os dados do cliente.
Data e hora da execução:	22/03/2022 20:33
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Ao informar as credenciais corretas o usuário consegue realizar o procedimento de login e é redirecionado a página “Minha conta”.

Caso de teste número 3	
Título:	Adicionar roupas ao carrinho de compras.
Cenário de teste:	@CNT003
Ator:	Cliente.
Objetivo:	Adicionar roupas ofertadas pela loja ao carrinho de compras.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve estar autenticado; 3 – O cliente deve estar na página dos produtos ofertados.
Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Printed Summer Dress”; 2 – No modal aberto clique em “continue comprando”; 3 – Adicione o item “Blouse” ao carrinho. 4 – No modal aberto clique novamente em “continue comprando”;
Resultado esperado:	Na aba carrinho deve conter os produtos que haviam sido adicionados, assim como os valores unitários de cada produto, o frete e seu valor total.
Data e hora da execução:	22/03/2022 20:35

Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	No card do carrinho os itens que foram adicionados aparecem visíveis assim que o mouse for posto sobre ele.

Caso de teste número 4	
Título:	Prosseguir para finalização da compra sem estar autenticado.
Cenário de teste:	@CNT004
Ator:	Cliente.
Objetivo:	Iniciar a finalização da compra e autenticar durante o processo.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente não deve estar autenticado; 4 – O cliente deve estar finalizando a compra.
Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Printed Summer Dress”; 2 – No modal aberto clique em “continue para finalizar”; 3 – Na página “sumário do carrinho” pressione novamente “continue para finalizar”; 4 – Na aba entrar, insira as seguintes credenciais: • E-mail: giulianni@yopmail.com • Senha: 123456 5 – Em seguida clique em entrar.
Resultado esperado:	Após realizar a autenticação será apresentada a aba endereço, a partir dela será possível finalizar a compra.
Data e hora da execução:	22/03/2022 20:36
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	As credenciais do usuário são solicitadas na aba login para que prossiga com a compra.

Caso de teste número 5	
Título:	Prosseguir para finalização da compra autenticado.
Cenário de teste:	@CNT005
Ator:	Cliente.

Objetivo:	Iniciar a finalização da compra com o usuário autenticado.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente não deve estar autenticado; 4 – O cliente deve estar finalizando a compra.
Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Printed Summer Dress”; 2 – No modal aberto clique em “prosseguir para finalizar”; 3 – Na página “sumário do carrinho” pressione novamente “continue para finalizar”;
Resultado esperado:	A etapa para realizar o login será saltada redirecionando para a aba de endereço e a partir dela será possível finalizar a compra.
Data e hora da execução:	22/03/2022 20:38
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	As credenciais para o login não serão solicitadas e será possível finalizar a compra.

Caso de teste número 6	
Título:	Cadastrar um novo endereço.
Cenário de teste:	@CNT006
Ator:	Cliente.
Objetivo:	Realizar o cadastro de um novo endereço.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página “minha conta”.
Entradas:	1 – Clique no botão “meu endereço”; 2 – Em seguida clique no botão “adicionar um novo endereço”; 3 – Na página “seu endereço”, preencha os dados nos campos obrigatórios: • Endereço: Oak street, 91 • Cidade: Springfield • Estado: Florida • Cep: 25022 • Telefone residencial: 55432102 • Celular: 45091352

	• Aliás: Meu Novo Endereco 4 – Clique no botão “Salvar”.
Resultado esperado:	O novo endereço deverá ser listado.
Data e hora da execução:	22/03/2022 20:39
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O endereço cadastrado aparecerá entre as opções de endereço.

Caso de teste número 7	
Título:	Atualizar um endereço já existente.
Cenário de teste:	@CNT007
Ator:	Cliente.
Objetivo:	Atualizar um endereço pré-cadastrado.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página “minha conta”.
Entradas:	1 – Clique no botão “meu endereço”; 2 – Em seguida no card do endereço “my address” clique em “atualizar”; 3 – Faça as seguintes modificações: • Endereço: Oak street, 200 4 – Clique no botão “Salvar”.
Resultado esperado:	No card do endereço atualizado deve estar as informações que haviam sido modificadas.
Data e hora da execução:	22/03/2022 20:40
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O endereço será atualiza e exibido no card correspondente.

Caso de teste número 8	
Título:	Escolher uma opção de frete.
Cenário de teste:	@CNT008
Ator:	Cliente.

Objetivo:	Selecionar uma opção de frete durante a finalização da compra.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar finalizando uma compra.
Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Printed Summer Dress” 2 – No modal aberto clique em “continue para finalizar” 3 – Na página “sumário do carrinho” pressione novamente “continue para finalizar” 4 – Durante o processo de finalização da compra prosseguir pela opção “prosseguir para finalizar” 5 – Escolha o endereço “seu endereço de entrega” 6 – Clique novamente no botão “prosseguir para finalizar” 7 – Na aba frete clique na caixa de seleção para concordar com o termo de serviço
Resultado esperado:	Será apresentada a aba de seleção de opções de frete.
Data e hora da execução:	22/03/2022 20:41
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	A página de frete será apresentada com as opções de frete disponibilizadas pela loja.

Caso de teste número 9	
Título:	Finalizar uma compra
Cenário de teste:	@CNT009
Ator:	Cliente.
Objetivo:	Realizar todo o processo de uma compra.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar finalizando uma compra.
Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Printed Summer Dress” 2 – No modal aberto clique em “continue para finalizar” 3 – Na página “sumário do carrinho” pressione novamente “continue para

	finalizar” 4 – Durante o processo de finalização da compra prosseguir pela opção “prosseguir para finalizar” 5 – Escolha o endereço “seu endereço de entrega” 6 – Clique novamente no botão “prosseguir para finalizar” 7 – Na aba frete clique na caixa de seleção para concordar com o termo de serviço para concordar 6 – Clique novamente no botão “prosseguir para finalizar” 7 – Na aba de pagamento escolha a opção para pagar por transferência bancária 8 – Clique no botão “Confirmar o pedido”
Resultado esperado:	Será apresentada uma página confirmando o pedido contendo informações sobre a compra.
Data e hora da execução:	22/03/2022 20:42
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	A tela de confirmação do pedido aparecerá apresentando informações a respeito da compra.

Caso de teste número 10	
Título:	Filtrar produtos por vestido.
Cenário de teste:	@CNT010
Ator:	Cliente.
Objetivo:	Exibir os produtos da loja que sejam vestidos utilizando os filtros disponíveis no menu “vestidos”.
Pré-condições:	1 – O cliente deve acessar o site “My Store”.
Entradas:	1 – Clique no menu “vestidos” localizado abaixo do logo da loja.
Resultado esperado:	Será apresentado uma página contendo todos os produtos do tipo vestido vendidos pela loja.
Data e hora da execução:	22/03/2022 20:44
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Todos os vestidos vendidos pela loja aparecem listados.

Caso de teste número 11	
Título:	Filtrar produtos por camisa.
Cenário de teste:	@CNT011
Ator:	Cliente.
Objetivo:	Exibir os produtos da loja que sejam vestidos utilizando os filtros disponíveis no menu “camisas”.
Pré-condições:	1 – O cliente deve acessar o site “My Store”.
Entradas:	1 – Clique no menu “camisas” localizado abaixo do logo da loja.
Resultado esperado:	Será apresentado uma página contendo todos os produtos do tipo camisa vendidos pela loja.
Data e hora da execução:	22/03/2022 20:44
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Todas as camisetas vendidas pela loja aparecem para compra.

Caso de teste número 12	
Título:	Visualizar detalhes do pedido.
Cenário de teste:	@CNT012
Ator:	Cliente.
Objetivo:	Exibir os detalhes dos pedidos realizados.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve ter realizado ao menos um pedido de compra; 5 – O cliente deve estar na página “minha conta”.
Entradas:	1 – Clique no botão “histórico de pedidos e detalhes”; 3 – Clique no botão “detalhes” ao lado do card do pedido “NSCQFOLOS”
Resultado esperado:	Será apresentado os detalhes com todos os dados do pedido.
Data e hora da execução:	22/03/2022 20:45
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Os dados do pedido aparecendo as informações a respeito do pedido.

Caso de teste número 13	
Título:	Visualizar o carrinho.
Cenário de teste:	@CNT013
Ator:	Cliente.
Objetivo:	Visualizar os itens disponíveis no carrinho.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O menu do carrinho deve estar visível; 3 – O menu carrinho deve possuir pelo menos um produto.
Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Blouse” 2 – No modal aberto clique em “continue para finalizar”
Resultado esperado:	O sumário do carrinho estará disponível exibindo os produtos adicionados ao carrinho, assim como informações de valores unitários de cada produto, o frete e seu valor total
Data e hora da execução:	22/03/2022 20:46
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Os produtos adicionados ao carrinho são apresentados contendo seu valor unitário e o total da compra.

Caso de teste número 14	
Título:	Visualizar o carrinho vazio.
Cenário de teste:	@CNT014
Ator:	Cliente.
Objetivo:	Visualizar os itens disponíveis no carrinho.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O menu do carrinho deve estar visível; 3 – O menu carrinho deve estar vazio.
Entradas:	1 – Clique no título “carrinho”;
Resultado esperado:	O sumário do carrinho se abrirá informando que o carrinho está vazio.
Data e hora da execução:	22/03/2022 20:47
Analista responsável pela execução do teste:	Giulianni Oliveira

Resultado real:	O menu está vazio e a mensagem “Seu carrinho está vazio” é exibida.

Caso de teste número 15	
Título:	Atualizar o cadastro do cliente.
Cenário de teste:	@CNT015
Ator:	Cliente.
Objetivo:	Realizar a atualização dos dados pessoais do cliente.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página “minha conta”.
Entradas:	1 – Clique no botão “minha informação pessoal”; 2 – No formulário aberto modifique a seguinte informação: • Dia da data de nascimento: 10 3 – Informe a senha atual “12345” 4 – Clique no botão “salvar”.
Resultado esperado:	O sistema deverá exibir a mensagem “Suas informações pessoais foram atualizadas com sucesso”.
Data e hora da execução:	22/03/2022 20:47
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou uma mensagem de sucesso.

Caso de teste número 16	
Título:	Visualizar o crédito de compra.
Cenário de teste:	@CNT016
Ator:	Cliente.
Objetivo:	Visualizar os créditos de compra disponíveis a conta do usuário.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página “minha conta”.

Entradas:	1 – Clique no botão “meus recibos de crédito”;
Resultado esperado:	No menu Recibos de crédito devem estar disponíveis os créditos de compra canceladas do cliente.
Data e hora da execução:	22/03/2022 20:48
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	São retornados os créditos de compra disponíveis do cliente.

Caso de teste número 17	
Título:	Realizar busca por produtos na caixa de busca do site.
Cenário de teste:	@CNT017
Ator:	Cliente.
Objetivo:	Utilizar a caixa de busca para procurar por produtos disponíveis.
Pré-condições:	1 – O cliente deve acessar o site “My Store”;
Entradas:	1 – Clique no campo de busca. 2 – Digite o produto “Printed Chiffon Dress”. 3 – Clique no ícone da lupa.
Resultado esperado:	Deverá ser retornado como resposta da pesquisa o produto utilizado na busca.
Data e hora da execução:	22/03/2022 20:49
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	É retornado o produto que possui o nome utilizado na busca.

Caso de teste número 18	
Título:	Adicionar um produto a lista de desejos.
Cenário de teste:	@CNT018
Ator:	Cliente.
Objetivo:	Favoritar um produto a lista de desejos.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página inicial

Entradas:	1 – Clique em “Mais” no produto “Printed Dress”. 2 – Na página do produto, clique em “Adicionar a lista de desejos”
Resultado esperado:	Um modal deverá se abrir para com a mensagem de confirmação em que o produto foi adicionado à lista de desejos.
Data e hora da execução:	22/03/2022 20:49
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou a mensagem “Adicionado a sua lista de desejos”.

Caso de teste número 19	
Título:	Visualizar a lista de desejos.
Cenário de teste:	@CNT019
Ator:	Cliente.
Objetivo:	Validar que um produto favoritado esteja na lista de desejos.
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página inicial
Entradas:	1 – Clique em “Mais” no produto “Printed Dress”. 2 – Na página do produto, clique em “Adicionar a lista de desejos” 3 – Clique no nome do usuário para abrir o menu “Minha conta” 4 – Clique em “Minhas listas de desejos” 5 – Na lista desejos “My wishlist” clique em “Visualizar”
Resultado esperado:	Deverá abrir os produtos adicionados à lista de desejos e o produto “Printed Dress” deverá estar entre os produtos.
Data e hora da execução:	22/03/2022 20:50
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O produto “Printed Dress” se encontra entre os produtos favoritados.

Caso de teste número 20	
Título:	Garantir a inserção de um e-mail válido ao criar conta.
Cenário de teste:	@CNT020
Ator:	Cliente.

Objetivo:	O sistema deve garantir a validade do e-mail inserido no campo “Endereço de e-mail” em “Criar uma conta”
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente não deve possuir uma conta; 3 – O cliente deve estar na página “autenticação”;
Entradas:	1 – No card “Criar uma conta” insira no campo “Endereço de e-mail” a seguinte informação: • 1234 2 – Clique em “Criar uma conta”
Resultado esperado:	Uma mensagem informando que o e-mail é inválido deverá aparecer impedindo a criação da conta.
Data e hora da execução:	22/03/2022 20:51
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema apresentou a mensagem “Endereço de e-mail inválido” e não prosseguiu com a criação da conta.

Caso de teste número 21	
Título:	Garantir a inserção de um e-mail válido ao realizar o login.
Cenário de teste:	@CNT021
Ator:	Cliente.
Objetivo:	O sistema deve garantir a validade do e-mail inserido no campo “Endereço de e-mail” em “Já registrado?”
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar na página “autenticação”;
Entradas:	1 – No card “Já registrado?” insira no campo “Endereço de e-mail” a seguinte informação: • 1234 2 – Clique em “Entrar”
Resultado esperado:	Uma mensagem informando que o e-mail é inválido deverá aparecer impedindo a realização do login
Data e hora da execução:	22/03/2022 20:51
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema apresentou a mensagem “Endereço de e-mail inválido” e não

	prosseguiu com o login

Caso de teste número 22	
Título:	Garantir a inserção da senha correta ao realizar o login.
Cenário de teste:	@CNT022
Ator:	Cliente.
Objetivo:	O sistema deve garantir que a senha esteja correta impedindo o login do cliente ao inserir uma senha errada no campo “Senha” em “Já registrado?”
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar na página “autenticação”;
Entradas:	1 – No card “Já registrado?” insira as seguintes informações: • Endereço de e-mail: giulianni@yopmail.com • Senha: cliente 2 – Clique em “Entrar”
Resultado esperado:	Uma mensagem informando que a autenticação falhou.
Data e hora da execução:	22/03/2022 20:52
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema apresentou a mensagem “Autenticação falhou” e não prosseguiu com o login

Caso de teste número 23	
Título:	Solicitar a redefinição de senha através da função “Esqueceu sua senha?”
Cenário de teste:	@CNT023
Ator:	Cliente.
Objetivo:	Solicitar redefinição da senha do usuário com sucesso utilizando a função “Esqueceu sua senha?”
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar na página “autenticação”;
Entradas:	1 – No card “Já registrado?” clique na opção “Esqueceu sua senha?”: 2 – Insira o e-mail “giulianni@yopmail.com” 3 – Clique em “Recuperar senha”

Resultado esperado:	O sistema deve retornar uma mensagem informando o envio de um e-mail para realizar a recuperação da senha
Data e hora da execução:	22/03/2022 20:52
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema apresentou a mensagem “Um e-mail de confirmação foi enviado para seu endereço: ”

Caso de teste número 24	
Título:	Garantir a validação de um e-mail já utilizado para a criação de uma nova conta
Cenário de teste:	@CNT024
Ator:	Cliente.
Objetivo:	O sistema deve impedir a criação da conta com o e-mail já utilizado
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente não deve possuir uma conta; 3 – O cliente deve estar na página “autenticação”;
Entradas:	1 – No card “Criar uma conta” insira no campo “Endereço de e-mail” a seguinte informação: • giulianni@yopmail.com 3 – Clique em “Criar uma conta”
Resultado esperado:	O sistema deve retornar uma mensagem informando que o e-mail já pertence a uma conta registrada
Data e hora da execução:	22/03/2022 20:53
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema apresentou a mensagem “Uma conta usando esse endereço de e-mail já foi registrada. Por favor insira uma senha válida ou solicite uma nova.” e impediu a criação de uma nova conta

Caso de teste número 25	
Título:	Garantir a inserção dos campos obrigatórios no cadastro de nova conta
Cenário de teste:	@CNT025

Ator:	Cliente.
Objetivo:	O sistema deve impedir a criação da conta sem informar os campos obrigatórios
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente não deve possuir uma conta; 3 – O cliente deve estar na página “autenticação”;
Entradas:	1 – No card “Criar uma conta” insira no campo “Endereço de e-mail” a seguinte informação: • novocliente@yopmail.com 2 – Clique em “Criar uma conta” 3 – No formulário apresentado não inserir informações 4 – Clicar no botão “Registrar”
Resultado esperado:	O sistema deve retornar uma mensagem informando os campos que deverão ser preenchidos
Data e hora da execução:	22/03/2022 20:54
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema apresentou a mensagem “Existem 8 erros”, apresentando a lista dos campos não preenchidos que são obrigatórios.

Caso de teste número 26	
Título:	Garantir que o cep informado siga as regras da quantidade de caracteres
Cenário de teste:	@CNT026
Ator:	Cliente.
Objetivo:	O sistema deve impedir a criação da conta ao informar um cep inválido
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente não deve possuir uma conta; 3 – O cliente deve estar na página “autenticação”;
Entradas:	1 – No card “Criar uma conta” insira no campo “Endereço de e-mail” a seguinte informação: • novocliente@yopmail.com 2 – Clique em “Criar uma conta” 3 – No formulário apresentado inserir as seguintes informações: • Primeiro nome: Matheus • Último nome: Pereira • Senha: 123456

	• Data de aniversário: 5 setembro 2000 • Primeiro nome: Matheus • Último nome: Pereira • Endereço: 7th Street, 42 • Cidade: North Town • Estado: Colorado • Cep: 7568907 • Telefone celular: 555478994451 4 – Clicar no botão “Registrar”
Resultado esperado:	O sistema deve retornar uma mensagem informando que o cep é inválido.
Data e hora da execução:	22/03/2022 20:55
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema apresentou a mensagem “O cep que foi informou é inválido. Ele deve seguir este formato: 00000”.

Caso de teste número 27	
Título:	Filtrar catálogo de roupas por tamanho
Cenário de teste:	@CNT027
Ator:	Cliente.
Objetivo:	Filtrar as roupas do catálogo feminino a partir do tamanho
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de catálogo de roupa feminina
Entradas:	1 – Nos filtros disponíveis, clique no filtro “S” do tipo tamanho
Resultado esperado:	A página irá carregar o filtro e serão apresentadas as roupas que possuem o tamanho informado
Data e hora da execução:	22/03/2022 20:57
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou as roupas que possuem o tamanho “S”.

Caso de teste número 28	
Título:	Filtrar catálogo de roupas por cor

Cenário de teste:	@CNT028
Ator:	Cliente.
Objetivo:	Filtrar as roupas do catálogo feminino a partir da cor
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de catálogo de roupa feminina
Entradas:	1 – Nos filtros disponíveis, clique no filtro “White” do tipo cor
Resultado esperado:	A página irá carregar o filtro e serão apresentadas as roupas que possuem variações na cor branca
Data e hora da execução:	22/03/2022 20:57
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou as roupas que possuem versões na coloração branca

Caso de teste número 29	
Título:	Filtrar catálogo de roupas por preço
Cenário de teste:	@CNT029
Ator:	Cliente.
Objetivo:	Filtrar as roupas do catálogo a partir do preço
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de catálogo de roupa feminina
Entradas:	1 – Nos filtros disponíveis, posicione o filtro preço entre os seguintes valores: • Aproximadamente $20,00 • Aproximadamente $40,00
Resultado esperado:	A página irá carregar o filtro e serão apresentadas as roupas que possuem o preço dentre os valores informados
Data e hora da execução:	22/03/2022 20:57
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou as roupas que possuem o preço entre os valores informados.

Caso de teste número 30	
Título:	Filtrar catálogo de roupas por tipo de composição

Cenário de teste:	@CNT030
Ator:	Cliente.
Objetivo:	Filtrar as roupas do catálogo a partir do tipo de composição de tecido
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de catálogo de roupa feminina
Entradas:	1 – Nos filtros disponíveis, clique no filtro “Cotton” do tipo composição
Resultado esperado:	A página irá carregar o filtro e serão apresentadas as roupas que possuem algodão em sua composição
Data e hora da execução:	22/03/2022 20:58
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou as roupas que possuem algodão em sua composição

Caso de teste número 31	
Título:	Filtrar catálogo de roupas por propriedade
Cenário de teste:	@CNT031
Ator:	Cliente.
Objetivo:	Filtrar as roupas do catálogo a partir de sua propriedade
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de catálogo de roupa feminina
Entradas:	1 – Nos filtros disponíveis, clique no filtro “Midi Dress” do tipo propriedade
Resultado esperado:	A página irá carregar o filtro e serão apresentadas as roupas são definidos como “Midi Dress”
Data e hora da execução:	22/03/2022 20:59
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou as roupas que são possuem a propriedade “Midi Dress”

Caso de teste número 32	
Título:	Filtrar catálogo de roupas por estilo
Cenário de teste:	@CNT032
Ator:	Cliente.

Objetivo:	Filtrar as roupas do catálogo a partir de seu estilo
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de catálogo de roupa feminina
Entradas:	1 – Nos filtros disponíveis, clique no filtro “Casual” do tipo estilo
Resultado esperado:	A página irá carregar o filtro e serão apresentadas as roupas são do estilo casual
Data e hora da execução:	22/03/2022 20:59
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou as roupas que são do estilo casuais

Caso de teste número 33	
Título:	Garantir que o sistema impeça a atualização da senha do usuário ao informar a senha atual incorreta
Cenário de teste:	@CNT033
Ator:	Cliente.
Objetivo:	Impedir a atualização da senha ao informar senha atual incorreta
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página “minha conta”.
Entradas:	1 – Clique no botão “minha informação pessoal”; 2 – Informe as seguintes informações nos campos: • Senha atual: cliente • Nova senha: 123456 • Confirmação: 123456 3 – Clique no botão “Salvar”
Resultado esperado:	O sistema deverá retornar uma mensagem informando da senha incorreta e impedindo a troca de senha.
Data e hora da execução:	22/03/2022 21:00
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou a mensagem “A senha que você informou é incorreta” e impediu a troca da senha.

Caso de teste número 34	
Título:	Informar senhas diferentes nos campos nova senha e confirmação ao atualizar a senha
Cenário de teste:	@CNT034
Ator:	Cliente.
Objetivo:	Impedir a atualização da senha ao informar a nova senha e sua confirmação incorretas
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar na página “minha conta”.
Entradas:	1 – Clique no botão “minha informação pessoal”; 2 – Informe as seguintes informações nos campos: • Senha atual: 123456 • Nova senha: cliente123 • Confirmação: cliente1 3 – Clique no botão “Salvar”
Resultado esperado:	O sistema deverá retornar uma mensagem informando que a nova senha e confirmação não conferem impedindo a troca de senha
Data e hora da execução:	22/03/2022 21:01
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornou a mensagem “A senha e a confirmação não conferem” e impediu a troca da senha

Caso de teste número 35	
Título:	Realizar pagamento por cheque
Cenário de teste:	@CNT035
Ator:	Cliente.
Objetivo:	Realizar o pagamento da compra usando cheque como forma de pagamento
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar finalizando a compra.

Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Printed Summer Dress” 2 – No modal aberto clique em “continue para finalizar” 3 – Na página “sumário do carrinho” pressione novamente “continue para finalizar” 4 – Durante o processo de finalização da compra prosseguir pela opção “prosseguir para finalizar” 5 – Escolha o endereço “seu endereço de entrega” 6 – Clique novamente no botão “prosseguir para finalizar” 7 – Na aba frete clique na caixa de seleção para concordar com o termo de serviço para concordar 6 – Clique novamente no botão “prosseguir para finalizar” 7 – Na aba de pagamento escolha a opção para pagar com cheque
Resultado esperado:	Será apresentada uma página informando a confirmação do pagamento por meio do método de transferência por cheque
Data e hora da execução:	22/03/2022 21:01
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	A tela de checagem do método de pagamento escolhido aparecerá apresentando informações a respeito da compra possibilitando a confirmação do pedido

Caso de teste número 36	
Título:	Realizar pagamento com transferência por conta bancária
Cenário de teste:	@CNT036
Ator:	Cliente.
Objetivo:	Realizar o pagamento da compra usando transferência por conta bancária como forma de pagamento
Pré-condições:	1 – O cliente deve acessar o site “My Store”; 2 – O cliente deve possuir uma conta; 3 – O cliente deve estar autenticado; 4 – O cliente deve estar finalizando a compra.
Entradas:	1 – Clique no botão “Adicionar ao carrinho” localizado abaixo do produto “Printed Summer Dress” 2 – No modal aberto clique em “continue para finalizar” 3 – Na página “sumário do carrinho” pressione novamente “continue para finalizar”

	4 – Durante o processo de finalização da compra prosseguir pela opção “prosseguir para finalizar” 5 – Escolha o endereço “seu endereço de entrega” 6 – Clique novamente no botão “prosseguir para finalizar” 7 – Na aba frete clique na caixa de seleção para concordar com o termo de serviço para concordar 6 – Clique novamente no botão “prosseguir para finalizar” 7 – Na aba de pagamento escolha a opção para pagar por transferência bancária
Resultado esperado:	Será apresentada uma página informando a confirmação do pagamento por meio do método de transferência por conta bancária
Data e hora da execução:	22/03/2022 21:02
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	A tela de checagem do método de pagamento escolhido aparecerá apresentando informações a respeito da compra possibilitando a confirmação do pedido

Caso de teste número 37	
Título:	Visualizar detalhes de um produto
Cenário de teste:	@CNT037
Ator:	Cliente.
Objetivo:	Ver mais informações a respeito do produto ofertado como tamanhos disponíveis e cor
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar com o catálogo de roupas aberto
Entradas:	1 – No produto “Blouse”, clique na imagem do produto
Resultado esperado:	Será apresentada a página da roupa contendo informações do material da roupa assim como os tamanhos e cores disponíveis para compra
Data e hora da execução:	22/03/2022 21:03
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Foi aberta a página da roupa mostrando os detalhes do produto e formas de pagamento

Caso de teste número 38	
Título:	Escolher a cor do produto para compra
Cenário de teste:	@CNT038
Ator:	Cliente.
Objetivo:	Escolher uma das colorações disponíveis de um produto e adicioná-lo ao carrinho
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar com o catálogo de roupas aberto
Entradas:	1 – No produto “Blouse”, clique na imagem do produto 2 – Na página do produto clique no ícone da cor branca localizado na seção cor 3 – Clique em “Adicionar ao carrinho”
Resultado esperado:	No modal aberto o produto deverá estar na cor escolhida
Data e hora da execução:	22/03/2022 21:04
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O produto na cor branca foi adicionado ao carrinho

Caso de teste número 39	
Título:	Escolher a quantidade do produto para compra
Cenário de teste:	@CNT039
Ator:	Cliente.
Objetivo:	Escolher a quantidade de um produto para compra e adicioná-lo ao carrinho
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar com o catálogo de roupas aberto
Entradas:	1 – No produto “Blouse”, clique na imagem do produto 2 – Na página do produto, digite 2 na sessão “Quantidade” 3 – Clique em “Adicionar ao carrinho”
Resultado esperado:	No modal aberto o produto deverá estar listado o produto com a quantidade escolhida
Data e hora da execução:	22/03/2022 21:04
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Dois produtos “Blouse” foram adicionados ao carrinho

Caso de teste número 40	
Título:	Escolher o tamanho de um produto para compra
Cenário de teste:	@CNT040
Ator:	Cliente.
Objetivo:	Escolher o tamanho de um produto para compra e adicioná-lo ao carrinho
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar com o catálogo de roupas aberto
Entradas:	1 – No produto “Blouse”, clique na imagem do produto 2 – Na página do produto selecione o tamanho M 3 – Clique em “Adicionar ao carrinho” 4 – Feche o modal de confirmação da adição do produto ao carrinho 5 – Abra o carrinho
Resultado esperado:	No carrinho deverá estar listado o produto com o tamanho selecionado
Data e hora da execução:	22/03/2022 21:05
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O produto “Blouse” no tamanho M foi adicionado ao carrinho

Caso de teste número 41	
Título:	Escrever a avaliação de um produto
Cenário de teste:	@CNT041
Ator:	Cliente.
Objetivo:	Escrever a avaliação de um produto ofertado pela “My Store”
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de um dos produtos
Entradas:	1 – No produto “Blouse”, clique em “Seja o primeiro a escrever sua avaliação!” 2 –Preencha os campos no modal da avaliação com os seguintes dados: • Dê 3 para a quantidade de estrelas • Título: Blusa bonita, mas peca na qualidade do tecido • Comentário: Gostei muito do modelo da blusa, mas achei que o tecido muito fino o que deixa a roupa um pouco transparente 3 – Clique no botão “Enviar”

Resultado esperado:	Será apresentado um modal informando do sucesso ao adicionar o comentário e que ele será avaliado para aprovação por um moderador
Data e hora da execução:	22/03/2022 21:05
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Um modal aparece contendo a seguinte mensagem “Seu comentário foi adicionado e será avaliado para aprovação por um moderador”

Caso de teste número 42	
Título:	Garantir que uma avaliação de um produto não seja criada ao não informar o título
Cenário de teste:	@CNT042
Ator:	Cliente.
Objetivo:	O sistema não deve permitir a criação de uma avaliação ao não informar o título
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de um dos produtos
Entradas:	1 – No produto “Blouse”, clique em “Seja o primeiro a escrever sua avaliação!” 2 –Preencha os campos no modal da avaliação com os seguintes dados: • Dê 3 para a quantidade de estrelas • Comentário: Gostei muito do modelo da blusa, mas achei que o tecido muito fino o que deixa a roupa um pouco transparente 3 – Clique no botão “Enviar”
Resultado esperado:	O sistema impedirá a criação da avaliação informando que o título está incorreto
Data e hora da execução:	22/03/2022 21:06
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornará a mensagem “Título é incorreto” impedindo a criação da avaliação

Caso de teste número 43	
Título:	Garantir que uma avaliação de um produto não seja criada ao não informar o comentário

Cenário de teste:	@CNT043
Ator:	Cliente.
Objetivo:	O sistema não deve permitir a criação de uma avaliação ao não informar o comentário
Pré-condições:	1 – O cliente deve acessar o site “My Store” 2 – O cliente deve estar na página de um dos produtos
Entradas:	1 – No produto “Blouse”, clique em “Seja o primeiro a escrever sua avaliação!” 2 –Preencha os campos no modal da avaliação com os seguintes dados: • Dê 3 para a quantidade de estrelas • Título: Blusa bonita, mas peca na qualidade do tecido 3 – Clique no botão “Enviar”
Resultado esperado:	O sistema impedirá a criação da avaliação informando que o comentário está incorreto
Data e hora da execução:	22/03/2022 21:07
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema retornará a mensagem “Comentário é incorreto” impedindo a criação da avaliação

APÊNDICE B – CASOS DE TESTE DO WEBSITE PARODIFOOD

Caso de teste número 1	
Título:	Visualizar a lista de restaurantes disponíveis.
Cenário de teste:	@CNT001
Ator:	Cliente.
Objetivo:	Visualizar a lista de restaurantes parceiros do aplicativo Parodifood.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela.
Resultado esperado:	Ao acessar o menu deve ser exibida a lista dos restaurantes conveniados ao Parodifood.
Data e hora da execução:	27/03/2022 21:00
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	É apresentada a lista dos restaurantes disponíveis no Parodifood.

Caso de teste número 2	
Título:	Visitar a página do restaurante.
Cenário de teste:	@CNT002
Ator:	Cliente.
Objetivo:	Abrir a página do restaurante para serem apresentados.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”.
Resultado esperado:	Deve ser aberta a página do restaurante apresentando as informações do restaurante.
Data e hora da execução:	27/03/2022 21:01
Analista responsável pela execução do teste:	Giulianni Oliveira

Resultado real:	É apresentada as informações do restaurante assim como seu menu.

Caso de teste número 3	
Título:	Adicionar itens ao carrinho.
Cenário de teste:	@CNT003
Ator:	Cliente.
Objetivo:	Adicionar um item vendido por um restaurante ao carrinho.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no prato pizza de muçarela.
Resultado esperado:	O item adicionado deve estar visível no card do carrinho, contendo informações como quantidade e preço do item.
Data e hora da execução:	27/03/2022 21:02
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O item é adicionado ao carrinho, que está contendo quantidade e preço dos itens.

Caso de teste número 4	
Título:	Realizar um pedido com pagamento por cartão refeição.
Cenário de teste:	@CNT004
Ator:	Cliente.
Objetivo:	Realizar um pedido de compra utilizando cartão refeição como método de pagamento.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza

	de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – Na página para finalizar o pedido insira os seguintes dados: • Nome: Juliana Souza • E-mail: julianasouza_12@yopmail.com • Confirmação do e-mail: julianasouza_12@yopmail.com • Endereço: Avenida Barão do Rio Branco • Número: 200 • Complemento: Vila São José • Formas de pagamento: Cartão refeição 6 – Após os itens adicionados, clicar no botão “Concluir Pedido”
Resultado esperado:	O sistema deverá exibir uma mensagem de sucesso.
Data e hora da execução:	27/03/2022 21:03
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema exibirá a mensagem de sucesso “Pedido Concluído” e poderá avaliar a experiência do site.

Caso de teste número 5	
Título:	Realizar um pedido com pagamento por cartão débito.
Cenário de teste:	@CNT005
Ator:	Cliente.
Objetivo:	Realizar um pedido de compra utilizando cartão débito como método de pagamento.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – Na página para finalizar o pedido insira os seguintes dados: • Nome: Juliana Souza • E-mail: julianasouza_12@yopmail.com • Confirmação do e-mail: julianasouza_12@yopmail.com

	• Endereço: Avenida Barão do Rio Branco • Número: 200 • Complemento: Vila São José • Formas de pagamento: Cartão débito 6 – Após os itens adicionados, clicar no botão “Concluir Pedido”
Resultado esperado:	O sistema deverá exibir uma mensagem de sucesso.
Data e hora da execução:	27/03/2022 21:05
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema exibirá a mensagem de sucesso “Pedido Concluído” e poderá avaliar a experiência do site.

Caso de teste número 6	
Título:	Realizar um pedido com pagamento por dinheiro.
Cenário de teste:	@CNT006
Ator:	Cliente.
Objetivo:	Realizar um pedido de compra utilizando dinheiro como método de pagamento.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – Na página para finalizar o pedido insira os seguintes dados: • Nome: Juliana Souza • E-mail: julianasouza_12@yopmail.com • Confirmação do e-mail: julianasouza_12@yopmail.com • Endereço: Avenida Barão do Rio Branco • Número: 200 • Complemento: Vila São José • Formas de pagamento: Dinheiro 6 – Após os itens adicionados, clicar no botão “Concluir Pedido”
Resultado esperado:	O sistema deverá exibir uma mensagem de sucesso.

Data e hora da execução:	27/03/2022 21:08
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema exibirá a mensagem de sucesso “Pedido Concluído” e poderá avaliar a experiência do site.

Caso de teste número 7	
Título:	Visualizar as avaliações realizadas por clientes do restaurante.
Cenário de teste:	@CNT007
Ator:	Cliente.
Objetivo:	Abrir as avaliações do restaurante.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – Na página do restaurante clique na aba “Avaliações”.
Resultado esperado:	Deverão ser exibidas as avaliações realizadas por clientes que já fizeram pedidos no restaurante.
Data e hora da execução:	27/03/2022 21:10
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Ao carregar a aba serão exibidas as avaliações dos clientes que fizeram pedidos no restaurante.

Caso de teste número 8	
Título:	Realizar busca por restaurantes através do nome.
Cenário de teste:	@CNT008
Ator:	Cliente.
Objetivo:	Conseguir encontrar um restaurante a partir da busca.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da

	tela. 2 – Na lista de restaurantes apresentada, clique no ícone de pesquisa. 3 – Digite o nome “Pizza Bug”.
Resultado esperado:	Deverá ser exibido o card correspondente ao nome do restaurante utilizado na busca.
Data e hora da execução:	27/03/2022 21:10
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	A pesquisa retornará o restaurante utilizado na busca.

Caso de teste número 9	
Título:	Garantir o mínimo de caracteres seja informado ao campo “Nome” ao finalizar um pedido.
Cenário de teste:	@CNT009
Ator:	Cliente.
Objetivo:	O sistema deve validar a quantidade de caracteres do campo.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – No campo “Nome” insira “Ana”.
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando a obrigatoriedade do campo e a quantidade de caracteres necessários.
Data e hora da execução:	27/03/2022 21:11
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo “Nome”, exibindo em conjunto a mensagem “Campo obrigatório e com 5 caracteres”.

Caso de teste número 10

Título:	Garantir a inserção de um e-mail válido ao campo “E-mail” ao finalizar um pedido.
Cenário de teste:	@CNT010
Ator:	Cliente.
Objetivo:	O sistema deve garantir a validade do e-mail inserido ao campo “E-mail”.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – No campo “e-mail” insira a entrada “abcd”.
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando que o e-mail é inválido.
Data e hora da execução:	27/03/2022 21:12
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo exibindo em conjunto a mensagem “E-mail inválido”.

Caso de teste número 11	
Título:	Garantir a inserção de um e-mail idêntico no campo Confirmação de e-mail ao que foi informado em E-mail durante a execução da finalização de um pedido.
Cenário de teste:	@CNT011
Ator:	Cliente.
Objetivo:	O sistema deve garantir que o e-mail informado no campo E-mail seja o mesmo que o de Confirmação do e-mail
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza

	de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – No campo “e-mail” insira a entrada “clientenovo@yopmail.com”. 6 – No campo “Confirmação do e-mail insira “clienteantigo@yopmail.com”.
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando que os e-mails não conferem.
Data e hora da execução:	27/03/2022 21:12
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo exibindo em conjunto a mensagem “Email-s não conferem”.

Caso de teste número 12	
Título:	Garantir o mínimo de caracteres no campo “Endereço” ao finalizar um pedido.
Cenário de teste:	@CNT012
Ator:	Cliente.
Objetivo:	O sistema deve validar a quantidade de caracteres do campo.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – No campo “Endereço” insira “Rua”.
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando a obrigatoriedade do campo e a quantidade de caracteres necessários.
Data e hora da execução:	27/03/2022 21:13
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo “Endereço”, exibindo em conjunto a mensagem “Campo obrigatório e com 5 caracteres”.

Caso de teste número 13	
Título:	Garantir que o campo “Número” contenha apenas números.
Cenário de teste:	@CNT013
Ator:	Cliente.
Objetivo:	O sistema deve validar a inserção de números ao campo.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – No campo “Número” insira “Num”.
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando a obrigatoriedade do campo e que o campo aceita somente números.
Data e hora da execução:	27/03/2022 21:13
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo “Número”, exibindo a mensagem “Obrigatório e somente números”.

Caso de teste número 14	
Título:	Garantir que uma forma de pagamento seja marcada.
Cenário de teste:	@CNT014
Ator:	Cliente.
Objetivo:	O sistema deve impedir a finalização do pedido ao não selecionar uma forma de pagamento.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela.

	4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – Na página para finalizar o pedido insira os seguintes dados: • Nome: Juliana Souza • E-mail: julianasouza_12@yopmail.com • Confirmação do e-mail: julianasouza_12@yopmail.com • Endereço: Avenida Barão do Rio Branco • Número: 200 • Complemento: Vila São José
Resultado esperado:	O botão “Concluir Pedido” deve permanecer desativado.
Data e hora da execução:	27/03/2022 21:14
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O botão “Concluir Pedido” permanecerá desativado e a compra não poderá ser concluída.

Caso de teste número 15	
Título:	Garantir que tenha ao menos um item no carrinho.
Cenário de teste:	@CNT015
Ator:	Cliente.
Objetivo:	O sistema deve impedir o prosseguimento da compra com a falta de itens no carrinho.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – Na página para finalizar o pedido insira os seguintes dados: • Nome: Juliana Souza • E-mail: julianasouza_12@yopmail.com • Confirmação do e-mail: julianasouza_12@yopmail.com • Endereço: Avenida Barão do Rio Branco • Número: 200 • Complemento: Vila São José

	• Formas de pagamento: Dinheiro 6 – Em itens do pedido, excluir o item “Pizza de mussarela”.
Resultado esperado:	O botão “Concluir Pedido” deve permanecer desativado.
Data e hora da execução:	27/03/2022 21:15
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O botão “Concluir Pedido” permanecerá desativado e a compra não poderá ser concluída.

Caso de teste número 16	
Título:	Adicionar itens de restaurantes diferentes ao carrinho.
Cenário de teste:	@CNT016
Ator:	Cliente.
Objetivo:	Adicionar itens de restaurantes diferentes ao carrinho.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no menu “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – Clique novamente no menu “Restaurantes”. 5 – Clique no card “Starbugs Coffe”. 6 – Clique no botão “+ Adicionar” no card do cappuccino com chantilly. 7 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 8 – Na página para finalizar o pedido insira os seguintes dados: • Nome: Juliana Souza • E-mail: julianasouza_12@yopmail.com • Confirmação do e-mail: julianasouza_12@yopmail.com • Endereço: Avenida Barão do Rio Branco • Número: 200 • Complemento: Vila São José • Formas de pagamento: Dinheiro 9 – Clique em “concluir pedido”.
Resultado esperado:	O sistema deverá exibir uma mensagem de sucesso.

Data e hora da execução:	27/03/2022 21:16
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	O sistema exibirá a mensagem de sucesso “Pedido Concluído” e poderá avaliar a experiência do site.

Caso de teste número 17	
Título:	Garantir a inserção de um e-mail válido ao campo “Confirmação do e-mail” ao finalizar um pedido.
Cenário de teste:	@CNT017
Ator:	Cliente.
Objetivo:	O sistema deve garantir a validade do e-mail inserido no campo “Confirmação do e-mail”.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – No campo “Confirmação do e-mail” insira a entrada “abcd”.
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando que o e-mail é inválido.
Data e hora da execução:	27/03/2022 21:18
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo exibindo em conjunto a mensagem “E-mail inválido”.

Caso de teste número 18	
Título:	Garantir que o campo “Endereço” seja preenchido ao finalizar um pedido.

Cenário de teste:	@CNT012
Ator:	Cliente.
Objetivo:	O sistema deve validar a obrigatoriedade do campo.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – Clique no campo “Endereço”, mas deixe o campo vazio. 6 – Clique no campo “Número”
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando a obrigatoriedade do campo e a quantidade de caracteres necessários.
Data e hora da execução:	27/03/2022 21:18
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo “Endereço”, exibindo em conjunto a mensagem “Campo obrigatório e com 5 caracteres”.

Caso de teste número 19	
Título:	Garantir que o campo “Nome” seja preenchido ao finalizar um pedido.
Cenário de teste:	@CNT019
Ator:	Cliente.
Objetivo:	O sistema deve validar a obrigatoriedade do campo.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”.

	5 – Clique no campo “Nome”, mas deixe o campo vazio. 6 – Clique no campo “E-mail”
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando a obrigatoriedade do campo.
Data e hora da execução:	27/03/2022 21:19
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo junto com a mensagem “Campo obrigatório e com 5 caracteres”.

Caso de teste número 20	
Título:	Garantir que o campo “Número” seja preenchido ao finalizar um pedido.
Cenário de teste:	@CNT020
Ator:	Cliente.
Objetivo:	O sistema deve validar a obrigatoriedade do campo.
Pré-condições:	1 – O cliente deve acessar o site “Parodifood”.
Entradas:	1 – Clique no botão “Restaurantes” localizado no canto esquerdo superior da tela. 2 – Na lista de restaurantes apresentada, clique no card do restaurante “Pizza Bug”. 3 – No menu do restaurante, clique no botão “+ Adicionar” no card da pizza de mussarela. 4 – No card intitulado “carrinho”, clique no botão “Fechar pedido”. 5 – Clique no campo “Número”, mas deixe o campo vazio. 6 – Clique no campo “Complemento”
Resultado esperado:	O sistema deverá retornar um alerta abaixo informando a obrigatoriedade do campo.
Data e hora da execução:	27/03/2022 21:20
Analista responsável pela execução do teste:	Giulianni Oliveira
Resultado real:	Será evidenciado o campo junto com a mensagem “Obrigatório e somente números”.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DA REITORA

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO nº 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Giulianni dos Santos Oliveira
do Curso de Engenharia da computação, matrícula 2017100330053O,
telefone: (62) 993179796 e-mail 2017100330053O@pucgo.edu.br, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Critérios de aceitação: Uma comparação entre testes manuais e automatizados gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 15 de fevereiro de 2022.

Assinatura do autor: Giulianni dos S. Oliveira

Nome completo do autor: Giulianni dos Santos Oliveira

Assinatura do professor–orientador: _____

Nome completo do professor-orientador: _____