UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO - UFES

CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE

CURSO DE SISTEMAS DE INFORMAÇÃO

SARA OLÍMPIO DO CARMO

IMPLEMENTAÇÃO DE TESTES AUTOMATIZADOS UTILIZANDO FRAMEWORK EM COMPARAÇÃO COM TESTES MANUAIS: UM ESTUDO DE CASO

ALEGRE
2019

SARA OLÍMPIO DO CARMO

IMPLEMENTAÇÃO DE TESTES AUTOMATIZADOS UTILIZANDO FRAMEWORK EM COMPARAÇÃO COM TESTES MANUAIS: UM ESTUDO DE CASO

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.
Orientadora: Profa. MSc. Valéria Alves da Silva

Alegre
2019

SARA OLÍMPIO DO CARMO

IMPLEMENTAÇÃO DE TESTES AUTOMATIZADOS UTILIZANDO FRAMEWORK EM COMPARAÇÃO COM TESTES MANUAIS: UM ESTUDO DE CASO

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em 12 de dezembro de 2019.

COMISSÃO EXAMINADORA

Profa. MSc. Valéria Alves da Silva
Universidade Federal do Espírito Santo
Orientadora

Prof. DSc. Bruno Vilela Oliveira
Universidade Federal do Espírito Santo

Prof. MSc. Fabrício Guedes Bissoli
Universidade Federal do Espírito Santo

Agradecimentos

Agradeço primeiramente a Deus por ter me permitido fazer minha faculdade que eu pedi em oração e por estar ao meu lado durante todo o percurso.

Agradeço à minha mãe por me incentivar a tentar mesmo quando eu não acreditava ser possível.

Agradeço à Universidade e seu corpo docente pelo conhecimento a mim entregue, este que levarei por toda a minha trajetória com imensa gratidão.

Agradeço à minha orientadora Valéria por toda paciência, ajuda e conhecimento transmitido.

Agradeço ao meu padrasto Davi por toda a boa vontade quando precisei e pelas caronas tarde da noite pra casa.

Agradeço à toda minha família pelo suporte, especialmente meus irmãos Isaque e Tim.

Agradeço ao meu empregador Márcio pela compreensão e paciência durante minha formação.

Agradeço à Gabriela que como minha irmã, sempre esteve ao meu lado quando achei que eu não seria capaz: "You are my person" .

Agradeço ao Ramon que por tantas vezes cuidou de mim e esteve ao comigo nos momentos mais difíceis como um irmão "You are my person too".

Agradeço aos meus amigos Jordana, Sebastião, Giseli, Jessé, Luís e Felipe que sempre estiveram ao meu lado.

Agradeço à Jane “Xani” por nunca deixar meu guarda-roupa e estômago ficarem vazios.

A todos que direta e indiretamente fizeram parte da minha formação, o meu muito obrigada.

"Que o medo de falhar não supere a vontade de conquistar".

Resumo

O software está presente em nosso cotidiano, desde atividades rotineiras a contextos cuja criticidade é maior. Garantir a qualidade e a confiabilidade desses sistemas é fundamental, e isto é possível por meio das atividades de teste de software. No entanto, certas funcionalidades podem ser complexas, repetitivas, ou ainda demandar diversas alterações ao longo do processo de desenvolvimento, o que exige muito esforço na execução dos testes, gera mais custos, e se torna exaustivo para o ser humano. Neste trabalho foi realizado um estudo de caso da implementação e execução de testes automatizados utilizando o *framework* Cypress em comparação com a execução manual. Foram realizados testes em funcionalidades com diferentes níveis de complexidade. A partir dos resultados obtidos, foi feita uma análise do impacto na utilização da automação de testes, baseando-se no tempo gasto na execução de um conjunto de testes, no total de testes executados durante um período de desenvolvimento e na quantidade de defeitos revelados.

Palavras-chave: Teste de software, teste automatizado, teste manual.

ABSTRACT

Software has been inserted into our daily tasks, since we wake up until we go to sleep. Besides the common tasks, there are also contexts in which the risks are higher, such as the systems running on airplanes, or equipment used in radiation therapy. Assure the quality and reliability of these systems is fundamental, and this is possible through software testing activities. However, certain features implemented in a software can be complex, repetitive, or even require several changes throughout the development process. Running the tests in these scenarios takes a lot of effort, generates more costs for the project, and becomes exhausting for humans to perform. In addition, the risk of defects being revealed close to the project delivery date is greater. In this scenario, a suitable way to run the test suite is through the automated testing. Among the several benefits the automation might provide, we can cite the faster defect identification, especially when introduced during feature changes or bug fixes, and the higher test case execution coverage throughout the project lifecycle.

Keywords: Software testing, automated testing, manual testing, testing coverage.

Lista de abreviaturas e siglas

[BDD] - *Behaviour Driven Development*

[PO] - *Product owner*

[VS Code] - *Visual Studio Code*

Lista de ilustrações

Figura 1 – Teste Funcional 6
Figura 2 – Behaviour Driven Development 12
Figura 3 – Login 14
Figura 4 – Funcionalidade Fale Conosco 15
Figura 5 – Arquivo de configuração BDD 16
Figura 6 – Exemplo de caso de teste na ferramenta *Hiptest* 17
Figura 7 – Função derivação em Javascript 18
Figura 8 – Exemplo do script implementado 19
Figura 9 – Visão Geral de execução no Hiptest 20
Figura 10 – Execução Testes Cypress com Interface 21
Figura 11 – Execução dos testes em modo *headless* 22
Figura 12 – Execução dos testes de *Login* com interface gráfica 23
Figura 13 – Execução dos testes de *Login* em modo headless 24
Figura 14 – Resultados da análise - *Login* 28
Figura 15 – Resultados da análise - *Fale Conosco* 29

Lista de tabelas

Tabela 1 – Passos na criação do BDD 12
Tabela 2 – Descrição dos comandos utilizados no Cypress 19
Tabela 3 – Tempo de implementação e execução dos testes de Login 25
Tabela 4 – Tabela de execução dos testes de Login durante uma sprint 26
Tabela 5 – Tempo de implementação e execução testes Fale conosco 26
Tabela 6 – Tabela de execução testes Fale Conosco após correção 27
Tabela 7 – Tabela de execução testes Fale Conosco após alteração 27

Sumário

Lista de ilustrações . . . . . **ix**

Lista de tabelas . . . . . **x**

1 INTRODUÇÃO **1**
1.1 Justificativa **1**
1.2 Estrutura do Trabalho **2**

2 OBJETIVO **3**
2.1 Objetivo Geral **3**
2.2 Objetivos Específicos **3**

3 REVISÃO DE LITERATURA **4**
3.1 Introdução ao Teste de *Software* **4**
3.1.1 Técnicas de Teste de *Software* 5
3.1.1.1 Teste Funcional 5
3.1.1.2 Teste Estrutural 5
3.1.1.3 Teste Baseado em Defeitos 6
3.1.2 Níveis de Teste 6
3.1.3 Fases de Teste 7
3.1.4 Teste de Regressão 8
3.1.5 Tipos de Execução de Teste 8
3.1.5.1 Teste Manual 8
3.1.5.2 Teste Automatizado 9
3.1.5.2.1 Ferramentas de Desenvolvimento de teste de Software Automatizado 9
3.2 Metodologia Ágil **10**

4 METODOLOGIA **14**
4.1 Definição do Ambiente de Aplicação do Estudo **14**
4.2 Criação dos Casos de Teste **16**
4.3 Configuração do Ambiente de Desenvolvimento **17**
4.4 Implementação dos Testes Automatizados **18**
4.5 Execução dos Testes Manuais **20**
4.6 Execução dos testes automatizados **21**

5 RESULTADOS **25**
5.1 Resultados dos Testes na Funcionalidade de *Login* **25**

5.1.1 Primeira execução 25
5.1.2 Segunda execução 25
5.2 Resultados dos Testes da Funcionalidade Fale Conosco 26
5.2.1 Primeira execução 26
5.2.2 Segunda execução 27
5.2.3 Terceira execução 27

6 CONCLUSÃO 28

REFERÊNCIAS 30

APÊNDICES 32

APÊNDICE A – CASOS DE TESTE - BDD 33

APÊNDICE B – CÓDIGO FONTE 41

1 Introdução

O uso de *software* é algo que tem crescido muito no cotidiano das pessoas e empresas. Seja em itens pessoais como *smartphone*, *smart watch*, e *tablets*, ou em geladeiras, televisões ou em ambientes comerciais no caso de bancos, hospitais, aeroportos, estacionamentos, lojas, etc., cada vez mais sistemas são desenvolvidos e utilizados. Com o crescimento do uso de programas para controlar e tornar mais eficientes as atividades do dia a dia, surgem também muitas novidades para desenvolvimento ou atualização dos sistemas em versões mais completas e abrangentes (GRAHAM; VEENENDAAL; EVANS, 2008; MYERS; SANDLER; BADGETT, 2011).

Devido ao aumento na demanda por desenvolvimento, observa-se também a necessidade de controlar a qualidade daquilo que é entregue ao usuário final. Conforme disse Bernardo e Kon (2008), "Os sistemas de *software* devem não só fazer o que o cliente precisa de acordo com o especificado, mas também fazê-lo de forma segura, eficiente e escalável, além de serem flexíveis, de fácil manutenção e evolução".

Um profissional desenvolvedor está sujeito a cometer enganos durante a programação de um sistema, e esse engano pode inserir um defeito no *software* que consequentemente o leve a falhar durante a sua execução. O prejuízo decorrente da falha de um sistema pode ser muito alto, além de impactar a imagem da empresa de forma negativa (IEEE, 1990).

Para alcançar um nível de sistema mais seguro e de fácil manutenção, é necessário que sejam executados testes durante o processo de desenvolvimento. Salvo algumas exceções, a maioria das empresas opta por garantir um nível de qualidade por meio de de testes manuais, onde espera-se encontrar os defeitos e corrigi-los antes que o sistema seja disponibilizado para o uso (BERNARDO; KON, 2008).

Dentro da engenharia de *software* existe uma área responsável por verificar por meio de um conjunto de atividades a qualidade do que é desenvolvido. Esta atividade é conhecida como teste de *software*, que pode ser executada de forma manual ou automatizada. Dessa forma, neste trabalho, temos como objetivo analisar essas duas formas de execução da suíte de teste criada verificando sua eficácia e cobertura.

1.1 Justificativa

Existem diversos testes executados no decorrer do processo de desenvolvimento de *software* que possuem cenários muito longos e repetitivos. Além disso, há a fase de execução de testes de regressão que é realizada após implementação de novas funcionalidades, alterações e/ou correções. Nos testes de regressão, a suíte de teste criada é re-executada com o objetivo de verificar se as alterações realizadas no código-fonte causaram efeito-colateral com a inserção de novos defeitos no *software*.

Conhecidas por sua característica destrutiva, as atividades de teste podem ser desmotivadoras e exaustivas para o ser humano devido às longas horas de trabalho repetitivo. Além disso, as equipes de qualidade costumam ter prazos de entrega pré-determinados e nem sempre suficientes. Consequentemente, diversos cenários de teste planejados podem ficar sem ser executados, causando assim uma cobertura de execução baixa e inadequada. Dessa forma, aumenta-se o risco de se entregar um produto com defeitos críticos que possam gerar prejuízos (MYERS; SANDLER; BADGETT, 2011).

Neste trabalho buscaremos demonstrar em uma experiência prática a eficácia da execução automatizada dos testes para facilitar e tornar menos exaustivo o processo de regressão, aumentar cobertura de execução e consequentemente a eficiência na descoberta de defeitos e antecipar sua correção.

1.2 Estrutura do Trabalho

Este trabalho foi organizado da seguinte forma: no Capítulo 2 são apresentados os objetivos geral e específicos, no Capítulo 3 apresenta-se uma fundamentação teórica sobre teste de software e metodologia de ágil, no Capítulo 4 apresenta-se a metodologia que foi utilizada para o desenvolvimento deste trabalho, no Capítulo 5 descreve-se os resultados obtidos e, por fim, no Capítulo 6 é apresentada a conclusão do trabalho.

2 Objetivo

2.1 Objetivo Geral

Este trabalho tem por objetivo geral a criação, desenvolvimento e execução de testes de forma manual e automatizada, analisando em seguida a cobertura de testes, o tempo de execução e o esforço empregado na implementação das duas abordagens (formas de execução).

Serão consideradas duas funcionalidades de um projeto denominado *Projeto X*[1], sendo elas o *Login* e o *Fale conosco*.

2.2 Objetivos Específicos

São objetivos específicos deste trabalho:

1. Criar casos de teste para as funcionalidades de *Login* e o *Fale conosco*;
2. Configurar o ambiente para o desenvolvimento dos testes automatizados;
3. Implementar os testes automatizados;
4. Executar os testes manuais;
5. Executar os testes automatizados;
6. Analisar e discutir os resultados;

[1] A fim de garantir o anonimato, o projeto analisado será mencionado como *Projeto X*

3 Revisão de Literatura

Esta seção é dedicada à revisão da literatura a respeito dos fundamentos de teste de software, metodologia ágil, e *Behavior Driven Development*, sendo estes assuntos discutidos e utilizados neste trabalho de conclusão de curso.

3.1 Introdução ao Teste de *Software*

As atividades de teste de *software* têm como objetivo encontrar erros causados por interações de subsistemas e problemas com a interface, e também tem objetivo de validar se o sistema cumpre os seus requisitos, para isto é necessário que o mesmo seja usado rigorosamente (SOMMERVILLE, 2011).

Segundo Myers, Sandler e Badgett (2011), o objetivo principal do teste de software é revelar defeitos, e tem como importância aumentar a qualidade e confiabilidade do sistema a fim de evitar prejuízos e danos críticos aos seus usuários. Considerando a crescente competitividade do mercado, é possível afirmar que o fator *qualidade* tem exercido um papel ainda mais fundamental. Para executar os casos de teste, são utilizados um conjunto de dados de entrada específicos que possuem um comportamento esperado conhecido e definido. Após a inserção destas entradas no sistema em teste, é verificado se o comportamento observado está em conformidade com o comportamento esperado, ou seja, neste cenário é observado se o *software* faz o que deve ser feito e se não faz o que não deve ser feito de acordo com o que foi especificado pelo cliente. O papel de decidir se o teste *passou* ou *falhou* é do *oráculo de teste*, que pode ser o próprio testador (ser humano) ou uma ferramenta automatizada (DELAMARO; JINO; MALDONADO, 2017).

Quando o *software* não funciona conforme o esperado, temos as seguintes definições de terminologias que podem ser aplicadas: engano, defeito, erro e falha. Estes termos podem ser utilizados de forma diferente na literatura, logo neste trabalho seguiremos a definição de acordo com IEEE (1990) que define:

- Engano: ação humana como, por exemplo, uma interpretação incorreta do requisito
- Defeito: imperfeição no código inserida (geralmente) por um engano;
- Erro: um estado inconsistente no sistema;
- Falha: quando o defeito é executado gerando um erro que ultrapassa a fronteira do sistema se tornando visível ao usuário. Ou seja, a falha só é perceptível em tempo de execução.

Dessa forma, pode-se sumarizar que um engano gera um defeito que, uma vez executado, gera um erro produzindo, assim, uma falha. Além destes termos, é comum entre os times de

desenvolvimento utilizar o termo *bug*, sendo este sinônimo de defeito (SPILLNER; LINZ; SCHAEFER, 2014).

Dentro deste contexto Myers, Sandler e Badgett (2011) afirma que quanto mais cedo um defeito é encontrado, mais baixo é o custo de sua correção e maior é a probabilidade de que a correção seja feita de forma correta reduzindo assim o retrabalho. Por outro lado, quanto mais próximo do estágio final de entrega do produto o defeito for encontrado, maior será o custo de sua correção. Dessa forma, conforme Graham, Veenendaal e Evans (2008), as atividades de teste devem acontecer durante todo o processo de desenvolvimento, de forma a revelar o quanto antes os defeitos nas unidades de *software* a fim de facilitar a depuração e correção.

Para afirmar que um *software* não possui defeitos, seria necessário a realização do teste exaustivo, o que é irreal. Seguindo essa ideia, grande parte dos sistemas possuem um abrangente número de cenários e combinações para serem testadas, porém testar tudo (realizar o exaustivo) é algo impraticável, pois demoraria muito e demandaria muito recurso humano. Dessa forma, é necessário identificar quais funcionalidades são mais críticas e prioritárias para serem testadas, e qual técnica de teste é mais apropriada para ser aplicada de forma a distribuir os recursos de forma correta e alcançar os resultados esperados (MYERS; SANDLER; BADGETT, 2011).

3.1.1 Técnicas de Teste de *Software*

Embora não seja possível realizar o teste exaustivo, é possível testar o *software* de forma sistemática seguindo um conjunto de regras, também conhecidas como critérios. Essa execução sistemática é realizada através do uso de técnicas de teste. Segundo Myers, Sandler e Badgett (2011), existem três principais técnicas, sendo elas *Teste Estrutural, Teste Funcional*, e *Teste Baseado em Defeitos*. O uso de uma técnica é fundamental para a criação de casos de testes mais eficazes no processo de revelar os defeitos (SPILLNER; LINZ; SCHAEFER, 2014).

3.1.1.1 Teste Funcional

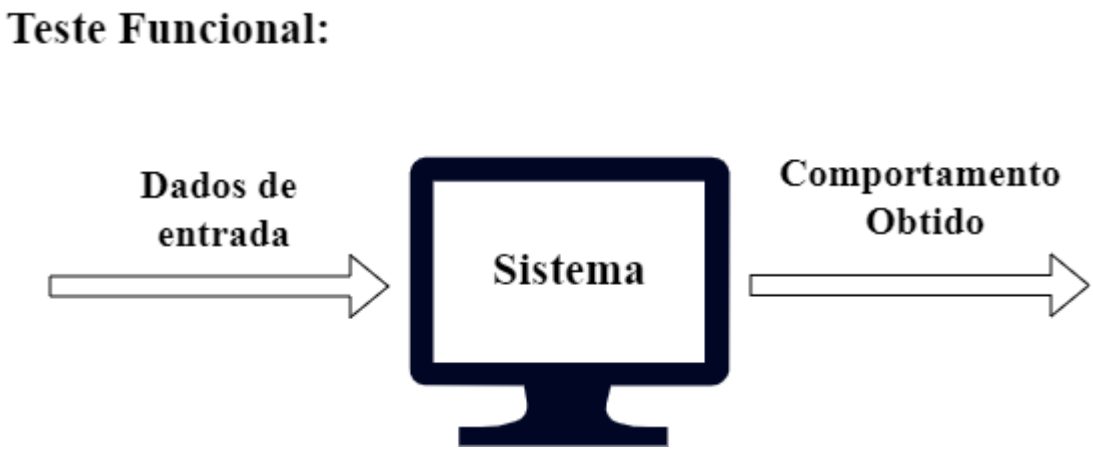
O Teste Funcional, também conhecido como teste caixa preta, é uma técnica na qual se observa o funcionamento do programa sem o conhecimento da sua estrutura interna, ou seja, do código-fonte. Para a criação dos casos de teste utilizando esta técnica, nos baseamos nas especificações do sistema e identificamos circunstâncias onde o comportamento observado está incoerente com o esperado, ou seja, cenários que revelem defeitos inseridos no código (MYERS; SANDLER; BADGETT, 2011).

Para aplicar o teste funcional são planejados casos de teste com um conjunto de dados de entrada e comportamentos esperados do sistema, como mostrado na Figura 1.

3.1.1.2 Teste Estrutural

O Teste Estrutural, também conhecido como teste caixa branca, é utilizado examinando a estrutura interna do sistema. Para isto nos baseamos na lógica de implementação e no código

Figura 1 – Teste Funcional



Fonte: Elaborado pela autora

desenvolvido. Nesta técnica são criados casos de teste específicos para cada componente interno sendo, assim, considerada relevante para manutenção, depuração e desenvolvimento de um *software* confiável.

Embora possa ser utilizada em qualquer nível de teste, o teste estrutural é recomendado especialmente para os níveis *Unidade* e *Integração* (MYERS; SANDLER; BADGETT, 2011).

3.1.1.3 Teste Baseado em Defeitos

A técnica de Teste Baseado em Defeitos tem por objetivo verificar a qualidade do conjunto de casos de teste criado, também conhecido como suíte de teste (MYERS; SANDLER; BADGETT, 2011). Esta técnica se baseia em criar uma cópia do programa original e inserir (de forma conhecida e proposital) defeitos no código. Em seguida, executa-se a suíte de testes nesta cópia do programa e espera-se que os testes falhem nas funcionalidades nas quais os defeitos foram inseridos. Se os testes passarem, significa que os casos de testes estão ineficazes em seu papel de revelar os defeitos existentes.

Embora esta seja uma técnica muito útil e importante, é pouco conhecida e utilizada na indústria devido à falta de profissionais que a conheçam, sua complexidade e esforço exigido durante o processo de desenvolvimento.

3.1.2 Níveis de Teste

Como já mencionado, é primordial iniciar as atividades de teste o mais cedo possível dentro do processo de desenvolvimento de *software* (MYERS; SANDLER; BADGETT, 2011; SPILLNER; LINZ; SCHAEFER, 2014). De acordo com Graham, Veenendaal e Evans (2008), há três níveis nos quais os testes podem ser realizados, sendo eles:

- *Unidade*: os testes em nível de unidade são realizados nas menores partes do *software*

de forma independente, podendo ser um método, função, classe, etc. Neste nível de teste é indicado o uso da técnica de Teste Estrutural e geralmente é realizado pelo próprio desenvolvedor devido ao conhecimento do código que desenvolveu;

- *Integração*: os testes em nível de integração são realizados com o intuito de revelar defeitos na comunicação entre as unidades; dependendo do tamanho do *software*, é recomendado que se utilize a técnica de Teste Funcional;

- *Sistema*: os testes nível de sistema buscam verificar o funcionamento do sistema como um todo, ou seja, verifica-se se a integração das unidades foi feita corretamento e se o *software* se comporta de acordo com a especificação. Neste nível de teste é indicado o uso da técnica de Teste Funcional e, se possível, deve ser realizado por uma equipe independente do desenvolvimento. Embora não seja muito utilizado e conhecido, o Teste Baseado em Defeitos também é indicado para este nível de teste.

3.1.3 Fases de Teste

Assim como no desenvolvimento, o teste de *software* possui um conjunto de fases em seu processo Graham, Veenendaal e Evans (2008), sendo elas:

- Planejamento: nesta etapa é criado um plano de testes com o intuito de definir os objetivos a serem alcançados. Este documento contém um conjunto de informações como as técnicas que serão utilizadas, em quais níveis, o time responsável e o papel de cada membro, as ações tomadas em casos de defeitos críticos identificados, etc.

- configuração de dados e ambiente: nesta etapa ocorre a configuração dos dados e do ambiente onde os testes serão implementados e/ou executados;

- Projeto de casos de teste: nesta etapa são criados os casos de teste de acordo com o que foi definido no plano de testes;

- Execução e avaliação: etapa na qual os testes são executados e os resultados são avaliados. A execução pode ser realizada tanto de forma manual quanto automatizada;

- Monitoramento: etapa na qual as atividades são monitoradas e comparadas com as definições feitas no plano de teste;

- Manutenção: esta fase envolve a manutenção da suíte de testes, podendo ser alterações, correções ou criação de novos casos de teste com o objetivo de cobrir as mudanças feitas no *software*;

3.1.4 Teste de Regressão

Quando um *software* é alterado, seja para correção ou implementação de novas funcionalidades, é desejável que não sejam inseridos novos defeitos, ou seja, espera-se que as funcionalidades continuem funcionando como esperado. Do contrário, para qualquer mudança no comportamento do *software* entende-se que um defeito foi inserido mediante a recente alteração. No entanto, efeitos colaterais costumam acontecer com as alterações, ou seja, surgem novos defeitos como consequência das mudanças realizadas no *software*(CARRINGTON, 1997).

Uma possível forma de remediar este efeito colateral é a realização dos testes de regressão, que podem ser realizados em todos os níveis, ou seja, unidade, integração e sistema. (SPILLNER; LINZ; SCHAEFER, 2014). O teste de regressão tem como objetivo identificar se novos defeitos foram inseridos no *software*, e essa identificação é realizada por meio da re-execução de um conjunto de casos de teste. No entanto, quantidade de casos de testes a serem re-executados na regressão pode ser muito grande. Além disso, se novos defeitos forem encontrados durante a regressão, deve-se reiniciar todo o processo de testes, visto que estas correções (durante a regressão) também podem ter inserido novos defeitos. Dessa forma, entende-se que este processo é longo e também pode ser repetitivo. Neste cenário, é recomendável (e fundamental) que seja realizada a sua automação dos casos de teste.

3.1.5 Tipos de Execução de Teste

Existem diversas atividades que são melhor executadas pelo computador, em contrapartida outras atividades são melhor executadas pelo ser humano. Planejar casos de teste é uma tarefa intelectual por natureza e, muitas vezes, é melhor realizado pelo ser humano. Existem duas formas de se executar os casos de teste, sendo o teste manual realizado pelo ser humano, e o automatizado que é desenvolvido pelo homem e executado por alguma ferramenta computacional (VINCENZI et al., 2018).

3.1.5.1 Teste Manual

O teste manual é uma das formas de execução de teste de *software*, no qual os casos de teste criados são executados sem o auxílio de ferramentas automatizadas. Este tipo de execução de teste é altamente indicada quando uma funcionalidade ou sistema possui um curto prazo de desenvolvimento, no qual não será necessário testar repetidas vezes (FREITAS, 2018).

Segundo Gomes (2017), o teste manual é mais indicado para os seguintes tipos de teste:

- Testes Exploratórios: baseado no conhecimento e experiência do testador; testa-se funcionalidades sem grande quantidade de documentações e casos de teste;
- Testes de Usabilidade: utilizado para verificar se o sistema atende os critérios de usabilidade, tais como a facilidade de uso;

3.1.5.2 Teste Automatizado

O teste automatizado consiste em utilizar um *software* que executa os casos de teste criados e compara os resultados obtidos com o esperado automaticamente. Sempre que existir a intenção de utilizar a automação, é necessário avaliar a viabilidade analisando se haverá ganho de tempo (redução de esforço), redução de custos, e se ainda manterá a qualidade dos testes (OLIVEIRA, 2018).

Segundo Gomes (2017), Oliveira (2018), a automação de testes é indicada para projetos que estão em constante mudança, funcionalidades críticas e testes de unidade (idealmente realizado pelos desenvolvedores). Um dos maiores benefícios da automação é facilitar o processo de regressão, conforme mostrado na Seção 3.1.4.

Dentre tudo o que foi apresentado até o momento, é importante também destacar que uma atividade fundamental na avaliação dos testes é a análise da cobertura, na qual é determinado um percentual de elementos requeridos por um critério de teste que foram executados(MALDONADO et al., 1998). De acordo com Bergman (2018) existem diferentes métricas para mensurar a cobertura, sendo algumas delas:

- Cobertura de Linha: observa se cada linha executável foi executada.
- Cobertura de Classe: observa se cada método de uma classe foi invocado.
- Cobertura de Caminho: observa se os caminhos possíveis na execução de uma função foram seguidos.

Além destes, há também métricas de cobertura como a verificação de casos de uso (baseada nos requisitos de *software*), avaliando a totalidade relacionada a um requisito (MDS, 2018). Especialmente quando se opta pela automação, é uma boa prática comparar a cobertura dos testes executados de forma manual e automatizada.

3.1.5.2.1 Ferramentas de Desenvolvimento de teste de Software Automatizado

Existem diversas ferramentas disponíveis para desenvolvimento de testes automatizados, e dentre elas estão Cypress[1], Protractor[2], Selenium[3].

O Cypress é um *framework* de testes no qual é possível configurar, desenvolver, executar e depurar testes diretamente no navegador. Este *framework* se comunica com o *Node.js*, que é uma plataforma construída sobre o motor *JavaScript* do *Google Chrome* para facilmente construir aplicações de rede rápidas e escaláveis"(NODEBR, 2016). Esta ferramenta também opera na camada de rede para controlar o trafego das informações, sendo capaz de modificar

[1] <https://www.cypress.io/> - acessado em novembro de 2019
[2] <https://www.protractortest.org/> - acessado em novembro de 2019
[3] <https://selenium.dev/> - acessado em novembro de 2019

toda informação que entra e sai do navegador, podendo assim automatizar suas funcionalidades (NODEBR, 2016).

O Protractor, por sua vez, é um *framework* de testes para aplicações AngularJs. "O AngularJS é um *framework open-source* de desenvolvimento front-end que possibilita o desenvolvimento de aplicações web, com foco em simplificar tanto a codificação quanto o processo de teste. (SAMPAIO, 2015)"Este framework se comunica com o browser sendo possível configurar, desenvolver e executar testes como um usuário faria de forma automatizada (PROTRACTOR, 2014).

Já o Selenium é uma ferramenta que permite a escrita de *scripts* de testes de forma organizada."O Selenium *WebDriver* faz chamadas diretamente ao navegador utilizando o suporte à automação nativo de cada navegador (GOES, 2017)". Esta ferramenta suporta a maioria dos navegadores de internet disponíveis atualmente, além de aceitar diversas linguens de programação como Phyton, Ruby, Javascript e Php (GOES, 2017).

3.2 Metodologia Ágil

As empresas de tecnologia possuem uma tendência à utilização de metodologias ágeis. O desenvolvimento ágil consiste em expressar os valores e princípios do manifesto para desenvolvimento ágil de *software* por meio de *frameworks* e práticas baseadas nestes princípios. (ALLIANCE, 2019). Ainda segundo o Alliance (2019), o manifesto foi escrito em meados de 2001 com o objetivo de encontrar novas maneiras de desenvolvimento de *software*, e busca valorizar:

1. Indivíduos e interações mais que processos e ferramentas;
2. *software* em funcionamento mais que documentação abrangente;
3. Colaboração com o cliente mais que negociação de contratos;
4. Responder a mudanças mais que seguir um plano;

Com base no manifesto ágil, uma segunda versão foi escrita com foco em testes de *software*, nas quais se busca valorizar (LUCAS, 2017) :

1. Testar por todas as etapas mais que testar no final;
2. Prevenir defeitos mais que encontrar defeitos;
3. Testar o entendimento mais que testar as funcionalidades;
4. Construir o melhor sistema mais que quebrar o sistema;
5. Responsabilidade do time pela qualidade mais que responsabilidade dos testadores;

Diversos *frameworks* surgiram a partir do manifesto ágil, como por exemplo o Scrum, que vem sendo utilizado na gestão e planejamento de projetos de software. Este *framework* possui um conjunto de ritos e papeis desempenhados pelos envolvidos nos projetos. O projeto em desenvolvimento é dividido em iterações chamadas *Sprints*, que representam um período de tempo (geralmente de duas a quatro semanas) e, ao final de cada *Sprint*, é apresentado um entregável ao cliente.

O Scrum é composto por um conjunto de pessoas que desempenham os seguintes papéis:

- *Product Owner* (PO): responsável por definir e dar prioridade aos itens presentes no *product backlog*.
- *Development Team*: responsável por transformar os itens do *product backlog* em entregáveis.
- *Scrum Master*: responsável por assegurar que os ritos do Scrum sejam seguidos pela equipe, e remover obstáculos que atrapalhem o time.

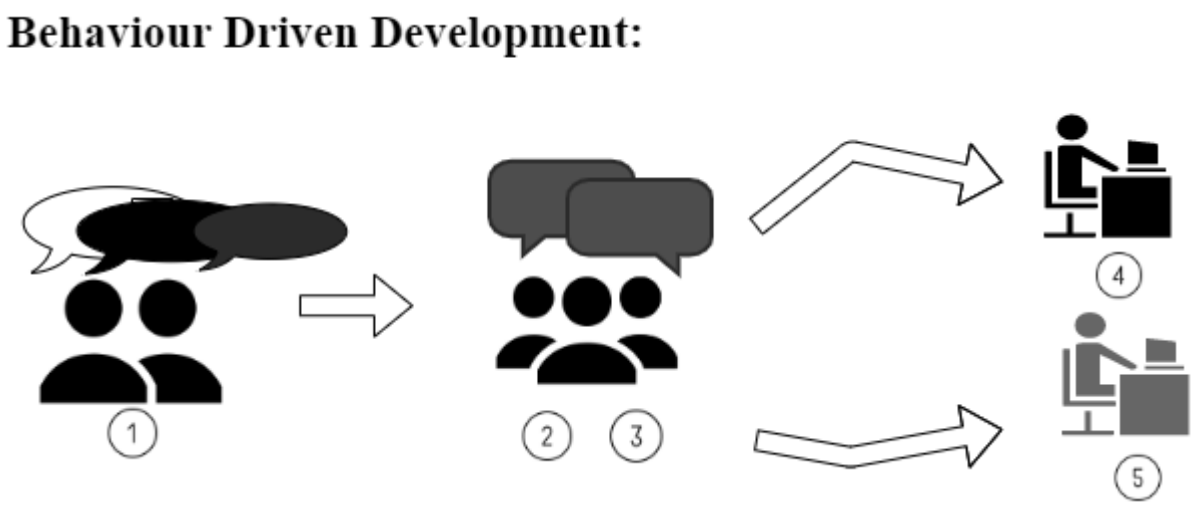
Além dos papeis, o Scrum é composto por um conjunto de ritos que são seguidos ao longo dos *Sprints*, sendo eles:

- *Sprint Planning*: rito no qual é feito um planejamento das atividades a serem desenvolvidas na *sprint* que se inicia.
- *Daily*: é uma reunião diária de no máximo quinze minutos para sincronizar as atividades do time e criar um planejamento para o dia seguinte.
- *Retrospective*: neste rito o time realiza uma auto-análise da *sprint* passada e cria um plano de melhorias para a próxima *sprint*.
- *Sprint Review*: rito onde é apresentado ao *product owner* o resultado do que foi alcançado na *sprint*, e é onde os itens do *product backlog* são repriorizados, caso necessário.

Usualmente as equipes de desenvolvimento adotam práticas que facilitem a comunicação entre as pessoas. Uma dessas práticas é o *Behaviour Driven Development* (BDD), uma técnica de desenvolvimento que foca no comportamento do sistema ao se basear na integração dos requisitos com a linguagem de programação. A linguagem utilizada no BDD é baseada nas especificações do sistema fornecidas pelo cliente durante o levantamento dos requisitos do *software*. Na aplicação do BDD, preocupa-se com o comportamento do sistema ao construir os cenários que serão desenvolvidos e testados (SOARES, 2011)

Na figura 2 são representados os passos para a elaboração do BDD enquanto a descrição pode ser acompanhada na Tabela 1.

Figura 2 – Behaviour Driven Development



Fonte: Elaborado pela autora

Tabela 1 – Passos na criação do BDD

Passo 1	Passo 2	Passo 3
O PO se reúne com o cliente para levantar informações.	O desenvolvedores, testadores e o PO elaboram os requisitos e cenários do sistema.	Desenvolvedores utilizam os cenários como base para desenvolver.
O PO realiza a definição e priorização do product backlog	Os cenários são escritos em formato BDD.	Testadores utilizam os cenários como base para seus testes.

Fonte: Elaborado pela autora

Segundo (SOARES, 2011) o uso do BDD simplifica a comunicação entre todos os membros envolvidos no projeto devido à sua linguagem simples e fluida. ALém disso, utiliza-se cenários conhecidos e descritos pelo próprio cliente e/ou PO. Na estrutura do BDD, cada cenário é descrito utilizando a linguagem *Gherkin*, que possui uma linguagem natural reconhecida por diversas ferramentas de execução de testes automatizados. O *Gherkin* contém um conjunto de palabras-chave, sendo elas:

- Given - [Dado] um contexto ou conjunto de pré-condições
- When - [Quando] acontecer um evento
- Then - [Então] se espera que aconteça algo
- And - [E] Opcionalmente utilizado após o *Given, When* ou *Then* quando houver mais de um resultado ou evento.

Dessa forma, seguindo esta estrutura como padrão, todos os interessados são capazes de compreender a documentação gerada para o projeto sem que a comunicação entre o time e o cliente fique prejudicada pelo excesso de jargões técnicos muitas vezes utilizados e incompreensíveis por pessoas leigas.

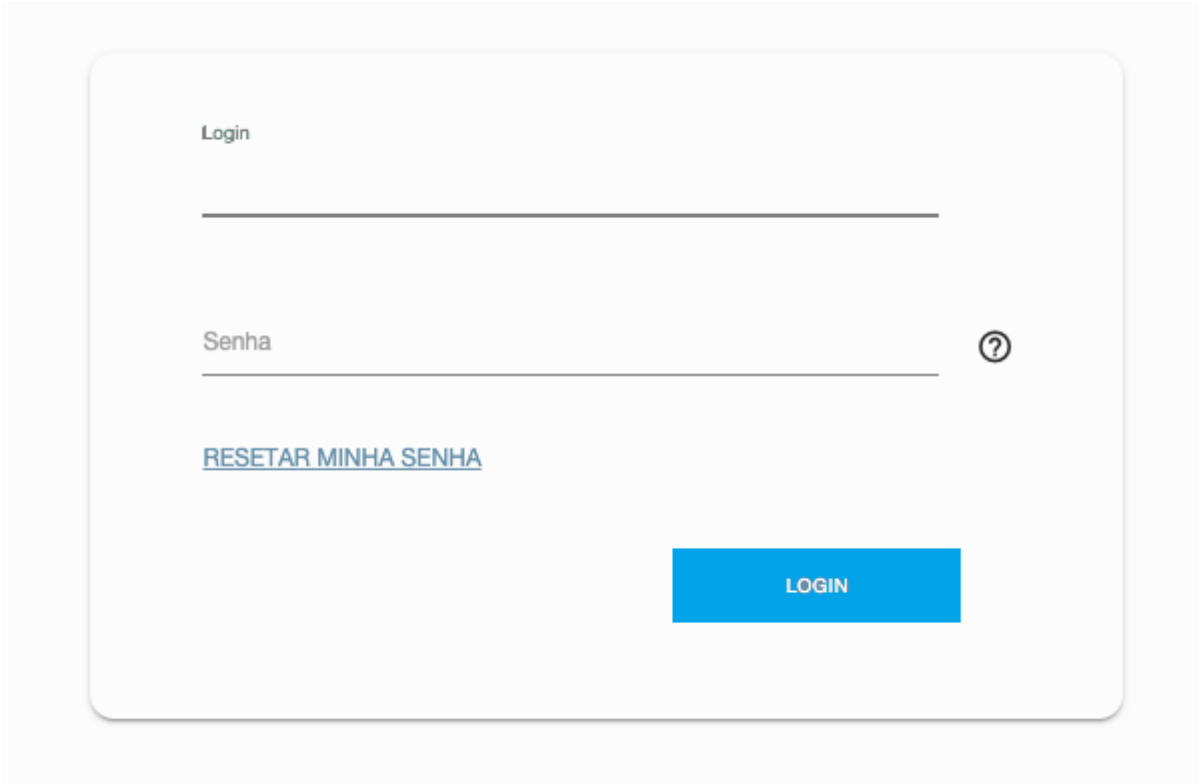
4 Metodologia

4.1 Definição do Ambiente de Aplicação do Estudo

A Tokenlab é uma empresa que desenvolve soluções digitais para diversas áreas de negócio. Dentre essas soluções, há um projeto ao qual iremos nos referir como *Projeto X*. Este projeto é desenvolvido na plataforma *web* e é responsável por gerenciar um sistema de cooperativas médicas operadoras de planos de saúde na área de contratação de seguros. As funcionalidades alvo deste estudo são o *Login* e *Fale Conosco*. Para melhor entendimento, o termo *Cliente* será utilizado para se referir à empresa para qual o Projeto X é desenvolvido, e o termo *Usuário* fará referência ao público alvo do sistema que irá utilizar o mesmo.

Para acessar algumas funcionalidades do sistema, o usuário deve estar logado. Na funcionalidade de *Login*, inserimos o nome de usuário (login) e a senha, como pode ser observado na Figura 12. Em seguida, o serviço checa se o usuário informado está cadastrado devidamente e, então, permite ou não o acesso ao sistema. Além disso, também é possível resetar a senha atual e criar uma nova senha, assim como deslogar o usuário logado.

Figura 3 – Login

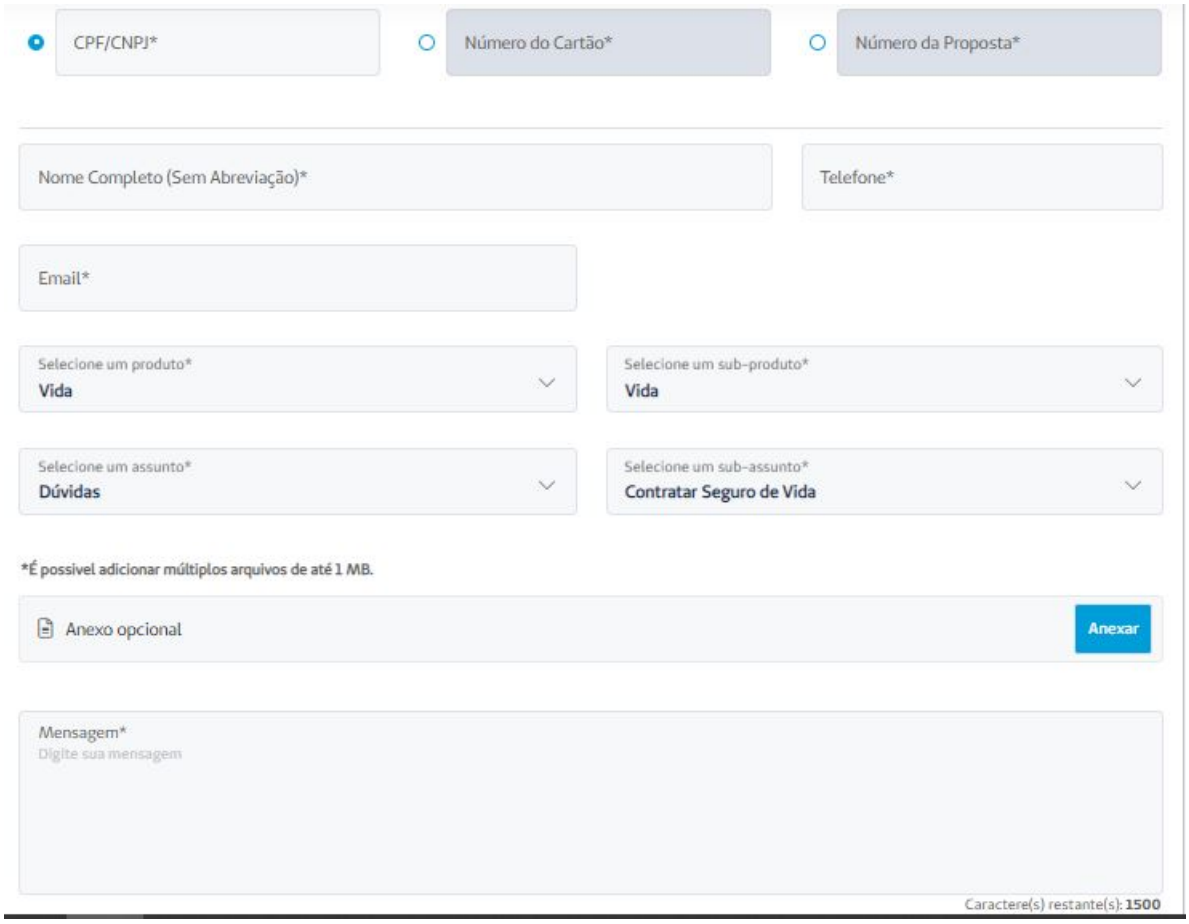


Fonte: Elaborado pela autora

A funcionalidade *Fale Conosco* é um serviço de apoio ao usuário, por meio do qual é

possível tirar dúvidas, solicitar alterações no produto e também registrar reclamações, sugestões e elogios. Dentro das competência do projeto, há quatro produtos disponibilizados, sendo eles: *vida, saúde, previdência, ramos elementares.* Nesta funcionalidade, para cada um dos quatro produtos do projeto existem sub-produtos, assuntos e sub-assuntos possíveis como pode ser observado na Figura 4. Quando o usuário insere seus dados corretamente e realiza a solicitação, o sistema a encaminha a um serviço especifico dependendo da combinação de informações inseridas, retornando um número de protocolo. Foram identificadas noventa e duas possíveis combinações de dados que podem ser informados, de acordo com a solicitação do cliente.

Figura 4 – Funcionalidade Fale Conosco



Fonte: Elaborado pela autora

A Metodologia utilizada no desenvolvimento do Projeto X é o Scrum, conforme apresentado na seção 3.2, e as *sprints* para cada entrega possuem um período de tempo de duas semanas. Para ter um controle das mudanças e observar se esta ocorrendo alguma inserção de defeitos no sistema, é realizado um teste de regressão semanal no sistema (Seção 3.1.4).

Para atingir o objetivo deste trabalho foram utilizados os seguintes passos de acordo com a seção 2.2: (a) criar casos de teste para as funcionalidades de *Login* e o *Fale Conosco*, (b) configurar o ambiente para o desenvolvimento dos testes automatizados, (c) implementar os testes automatizados, (d) executar os testes manuais, (e) executar os testes automatizados.

4.2 Criação dos Casos de Teste

Para a criação dos casos de teste utilizou-se o *Visual Studio Code* (VS Code)[1] que é um editor de código-fonte desenvolvido pela empresa Microsoft, e a ferramenta *Hiptest*[2], que é uma plataforma em nuvem para a criação, execução e refatoração de testes. (HIPTEST, 2019).

No editor VS Code escreveram-se os cenários para os BDDs juntamente com a tabela de dados de entrada possíveis dos sistemas. Esta tabela de dados é utilizada pela ferramenta, para isto os BDDs foram salvos em um arquivo do tipo *.feature* como pode ser observado na Figura 5. Este arquivo foi posteriormente importado pela ferramenta *Hiptest*.

Figura 5 – Arquivo de configuração BDD

```
Scenario Outline: Preencher Formulário Fale Conosco

  Given [Dado que] Estou na pagina Fale conosco
  When [Quando] Eu informar o nome "userName"
   And [E] Eu informar o CPF "userCpf"
   And [E] Eu informar o telefone "userTel"
   And [E] Eu informar uma mensagem "faleMensagem"
   And [E] Eu informar um email "userEmail"
   And [E] Eu selecionar a opção "<ProdutoInput>"
   And [E] Eu selecionar a opção "<SubProdutoInput>"
   And [E] Eu selecionar a opção "<AssuntoInput>"
   And [E] Eu selecionar a opção "<SubAssuntoInput>"
   And [E] clicar no botão "Concluir"
  Then [Então] O sistema deverá gerar um Protocolo de atendimento

Examples:
|Dataset name|userName|userCpf|userTel|userEmail|  iAmInput  |ProdutoInput| SubProdutoInput |    AssuntoInput   |
|    CT01    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   | Clube de Vantagens|
|    CT02    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |  Outros assuntos  |
|    CT03    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |        PGBL       |
|    CT04    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |        PGBL       |
|    CT05    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |        PGBL       |
|    CT06    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |        PGBL       |
|    CT07    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |        PGBL       |
|    CT08    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |    Previdência    |
|    CT09    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |   Previdência   |Sugestões e Elogios|
|    CT10    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |Ramos Elementares| Clube de Vantagens|
|    CT11    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |Ramos Elementares|        D&O        |
|    CT12    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |Ramos Elementares|        D&O        |
|    CT13    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |Ramos Elementares|        D&O        |
|    CT14    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |Ramos Elementares|        D&O        |
|    CT15    |userName|userCpf|userTel|userEmail|faleMensagem|  Segurado  |Ramos Elementares|        D&O        |
```

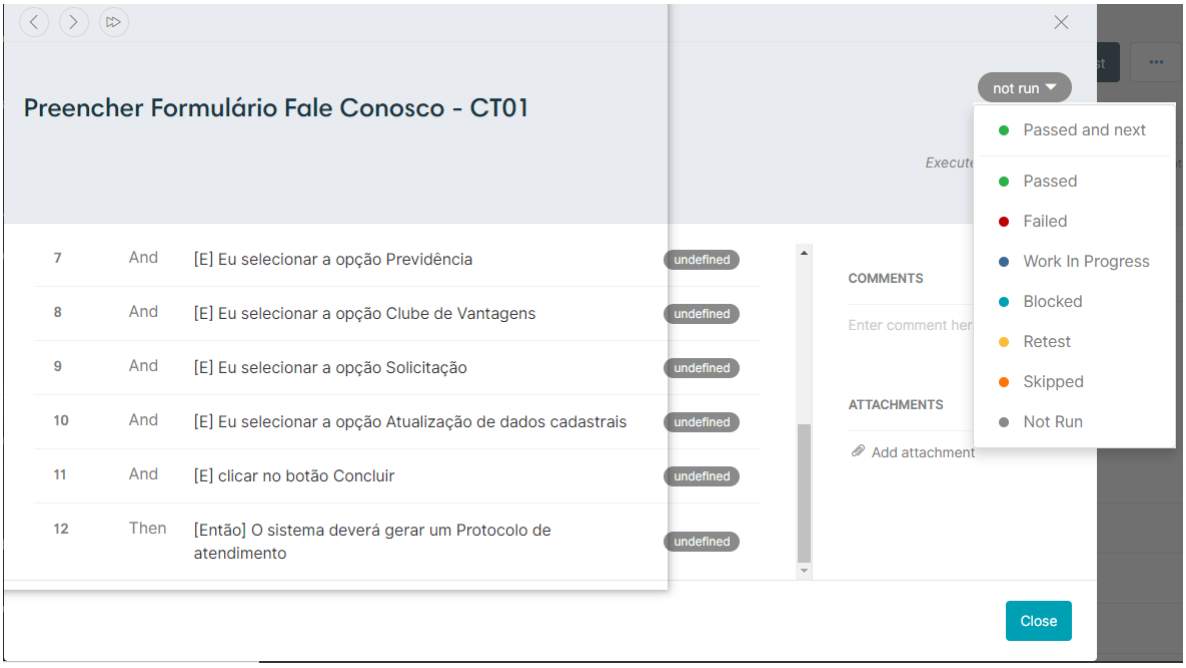
Fonte: Elaborado pela autora

Após a importação do arquivo no *Hiptest*, os passos inseridos no BDD foram organizados em uma sequencia de testes com as combinações possíveis para cada caso de teste. Na Figura 6 é apresentado como são exibidos os testes gerados no *Hiptest*, onde temos cada passo para execução e as opções nativas para informar o resultado obtido sendo elas: passou, falhou, retestar e etc.

O objetivo do conjunto de testes desenvolvido para a funcionalidade *Fale Conosco* foi testar se os serviços retornavam corretamente ou não os protocolos esperados após o preenchimento do formulário. Para isto utilizou-se dados pessoais fictícios como nome, CPF, e-mail, telefone, que são obrigatórios, porém não interferem no resultado esperado do serviço. Para a funcionalidade *Login* os dados foram disponibilizados pelo cliente para os testes. técnica de teste utilizada foi o teste funcional conforme visto na sessão 3.1.1.1, esta técnica foi escolhida devido ao fato de não termos acesso ao código dos serviços disponibilizados.

[1] <https://code.visualstudio.com/>- acessado em outubro, 2019
[2] <https://hiptest.com/>- acessado em outubro, 2019

Figura 6 – Exemplo de caso de teste na ferramenta *Hiptest*



Fonte: Elaborado pela autora

4.3 Configuração do Ambiente de Desenvolvimento

A ferramenta utilizada para a implementação dos testes automatizados foi o *Cypress*[3], onde foi levado em consideração a facilidade de uso e o fato de já ser utilizada pela equipe de qualidade da empresa Tokenlab.

Para a utilização do *Cypress* há como pré-requisito a necessidade de um gerenciador de pacotes *npm* instalado. Para a instalação deste pacote, seguimos os seguintes passos:

1. Criar uma pasta para o projeto;
2. Acessar a pasta criada (comando "cd nomeDoDiretorio")
3. Instalar a ferramenta *NodeJS*;
4. Abrir o terminal de comandos e executar o comando "npm init"(sem as aspas);

Com o comando *npm init* é gerada a pasta *node_modules*, onde ficam armazenadas as bibliotecas necessárias do projeto e o arquivo *package.json* que utilizamos para listar as dependências do projeto.

Com esta configuração finalizada, passamos a instalação do *Cypress* executando o seguinte comando no terminal: "npm install cypress –save-dev"

Uma vez instalado e já localizado dentro da pasta do projeto, deve-se executar o comando "./node_modules/.bin/cypress open"para inicializar o *Cypress*.

[3] <https://www.cypress.io/> - acessado em novembro, 2019

Com o *Cypress* já configurado no computador, o próximo passo foi a criação de um arquivo dentro da pasta *cypress* com a extensão "spec.js ". A partir deste ponto foi finalizada a configuração de ambiente e no próximo passo realizou-se a implementação dos *Scripts* de teste.

4.4 Implementação dos Testes Automatizados

Para realizar a implementação dos testes da funcionalidade *Fale Conosco* foram seguidos os seguintes passos:

1. Realizar a leitura do arquivo de derivações
2. Implementar testes para preencher as informações
3. Configurar os testes com os dados de entrada

Para a criação dos *scripts* de teste foi utilizado o padrão de projeto de automação *Page Objects*. neste padrão cria-se uma classe para cada página web e, com a utilização do paradigma de orientação a objetos, obtém-se encapsulamento nas classes, atributos e métodos. Vicente (2019).

Foi disponibilizado pelo cliente um arquivo JSON no qual estavam presentes as combinações de produto, subproduto, assunto e sub-assunto com um total de noventa combinações possíveis. Para a leitura do arquivo foi utilizada uma função chamada "derivação(Perfil)", escrita na linguagem Javascript e disponibilizada no repositório do projeto, como pode ser observado na Figura 7.

Figura 7 – Função derivação em Javascript

```
export function derivacao(perfil) {

    getDerivacaoPerfil(perfil, getDerivacoes())
    console.log(derivacoesMontadas);
    return derivacoesMontadas;
}
```

Fonte: Elaborado pela autora

A função recebe um perfil de usuário, e retorna um array chamado "derivacoesMontadas"que foi utilizado para acesso aos dados de entrada dos testes.

A cada iteração, o *script* executado preenche os campos definidos e seleciona uma combinação de produto, subproduto, assunto e sub-assunto presente no caso de teste.

Um exemplo do *script* implementado no Cypress pode ser observado na Figura 8, onde a estrutura para criação dos casos de teste é implementada. As principais funções utilizadas e suas descrições - segundo Cypress (2019) - são representadas na Tabela 2.

Figura 8 – Exemplo do script implementado

```
JS faleconosco.spec.js ●   JS FalePage.js   JS loginPage.js   JS PreenchePage.js   JS login.spec.js
cypress > integration > JS preencheFale.spec.js > ...
10  describe('Testes: Campos de dados Usuário Fale Conosco', () => {
11      // runs once before all tests in the block
12      before(() => {
13          // Acessar os dados do arquivo dataFile na pasta fixtures
14          cy.fixture('preencheData').then((data) => {
15              preencheData = data;
16          })
17
18      })
19
20
21      it('Todos os dados válidos', () => {
22
23          cy.visit('/')
24
25          falePage2.getInput().select(preencheData.userTipo);
26          falePage2.getCpf().type(preencheData.userCpfvalido);
27          falePage2.getName().type(preencheData.userName);
28          falePage2.getEmail().type(preencheData.userEmail);
29          falePage2.getTel().type(preencheData.userTel);
30          preDeriva();
31          falePage2.getMsg().type(preencheData.faleMensagem);
32          cy.get('.pl-2 > .btn').click()
33          cy.wait(20000)
34          cy.get('.description').should('be.visible')
35
36      })
```

Fonte: Elaborado pela autora

Tabela 2 – Descrição dos comandos utilizados no Cypress

Comando	Descrição
describe( )	Utilizada para criação de um conjunto de teste
it ( )	Utilizada para criar um caso de teste
cy.get ( )	Obtem um elemento da página
cy.visit ( )	Visita a Url informada
cy.wait ( )	Aguarda por um tempo ou elemento

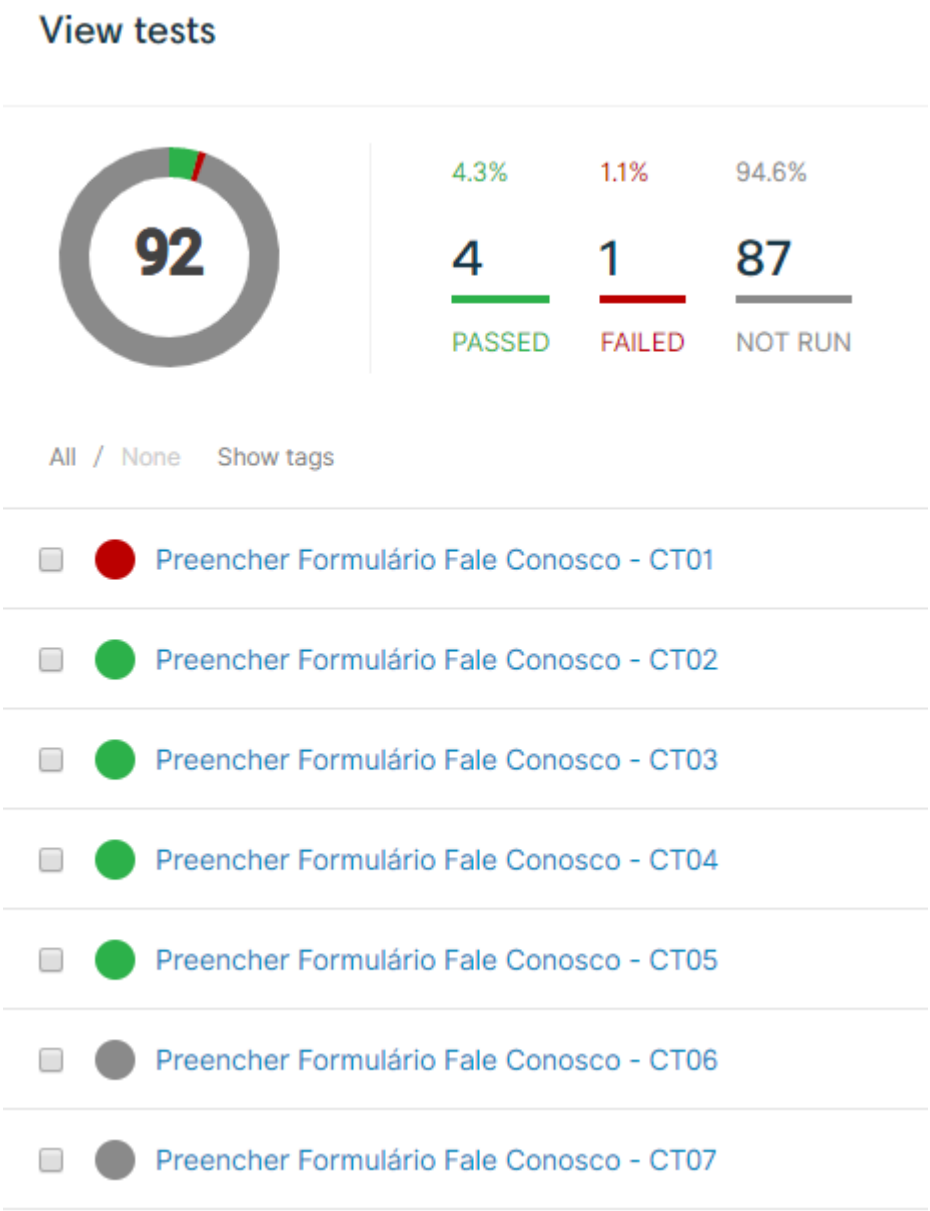
Fonte: Elaborado pela autora

O Projeto X ainda possui diversas funcionalidades que se encontram em desenvolvimento e manutenção paralelamente. A implementação dos casos de teste foi feita em intervalos de tempo com menor quantidade de tarefas demandadas.

4.5 Execução dos Testes Manuais

Para a execução dos testes manuais conforme a seção 3.1.5, criou-se um conjunto de testes nos quais são selecionados os casos de teste a serem executados dentre os disponíveis na ferramenta Hiptest. Os testes foram executados segundo os passos definidos, conforme visto na Figura 6. Para cada execução foram preenchidos todos os campos manualmente no sistema e sequencialmente foi marcado o resultado do teste na própria ferramenta. Na Figura 9 é ilustrada uma visão geral do andamento da execução em que quatro testes passaram, um teste falhou e oitenta e sete testes ainda não foram executados, além de ser possível selecionar um caso de teste e visualizar os passos seguidos no mesmo.

Figura 9 – Visão Geral de execução no Hiptest



Fonte: Elaborado pela autora

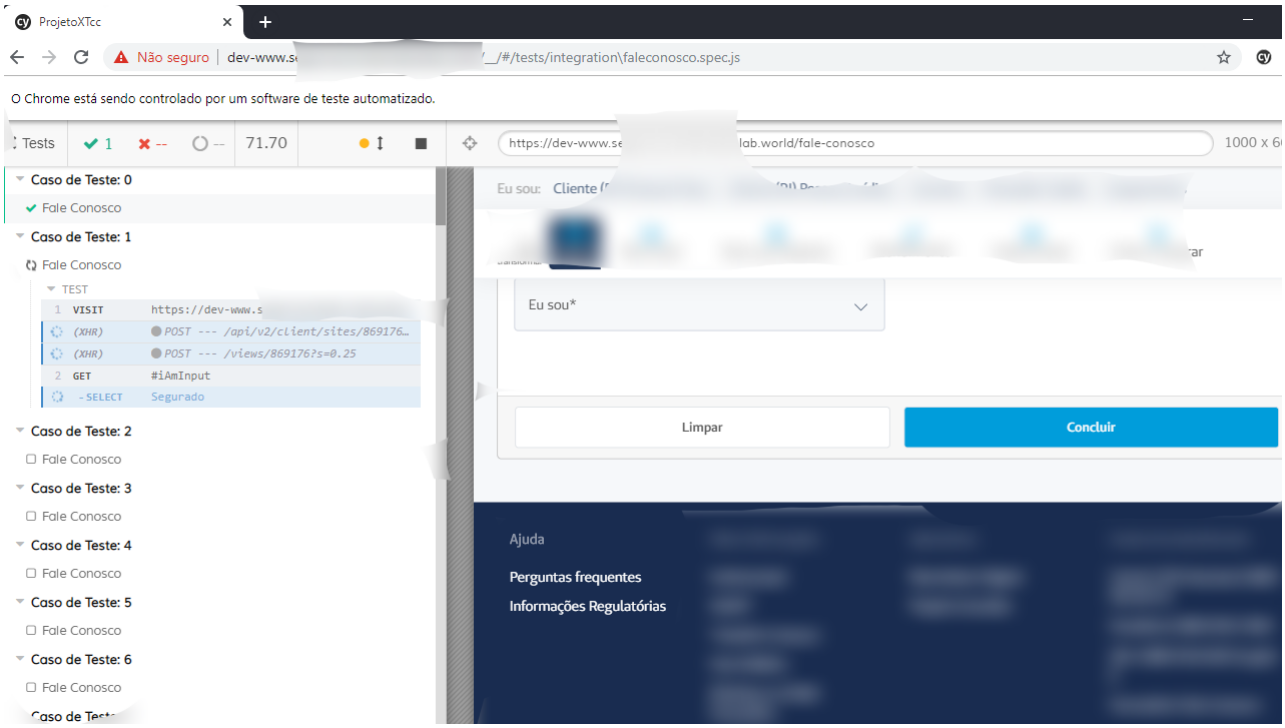
Como a ferramenta Hiptest não possui um marcador de tempo, um cronômetro foi utilizado para medir o tempo gasto nas execuções.

4.6 Execução dos testes automatizados

Utilizando a ferramenta Cypress é possível executar os testes automatizados, como visto na Seção 3.1.5, utilizando sua interface gráfica e acompanhando todos os passos em tempo de execução, ou em modo *headless*, no qual a execução ocorre em linha de comando sem utilização de elementos gráficos.

Para iniciar a execução com interface gráfica, inseriu-se o comando "npx cypress run". Com este comando, a ferramenta inicia uma instância do navegador exibindo os casos de teste em execução.

Figura 10 – Execução Testes Cypress com Interface



Fonte: Elaborado pela autora

Como pode ser observado na Figura 10, os testes foram executados diretamente no navegador. A ferramenta exibe na cor verde os testes aprovados, e na cor vermelha os testes que falharam. Também foi utilizada a lista de casos de teste implementados em fila para execução, para acompanhar qual cenário foi aplicado em cada teste. Após a execução, coletaram-se o tempo total de execução (em segundos) e a quantidade de defeitos encontrados.

Na execução em o modo headless, a interface gráfica não foi carregada para visualização, e os testes foram executados em linha de comando. Para a execução no modo *headless* utilizou-se o comando "./node_modules/.bin/cypress run cypress"

Figura 11 – Execução dos testes em modo *headless*

```
JS faleconosco.spec.js ×
cypress > integration > JS faleconosco.spec.js > ...
14    cy.get('#telefoneInput').type('2899999999')
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                  1: node

(Run Starting)

Cypress:    3.6.1
Browser:    Electron 73 (headless)
Specs:      1 found (faleconosco.spec.js)


Running:  faleconosco.spec.js                                        (1 of 1)


Caso de Teste: 0
  √ Fale Conosco (39220ms)

Caso de Teste: 1
  √ Fale Conosco (41926ms)

Caso de Teste: 2
  √ Fale Conosco (38421ms)

Caso de Teste: 3
  √ Fale Conosco (37157ms)

Caso de Teste: 4
  √ Fale Conosco (39929ms)

Caso de Teste: 5
  √ Fale Conosco (37533ms)
```

Fonte: Elaborado pela autora

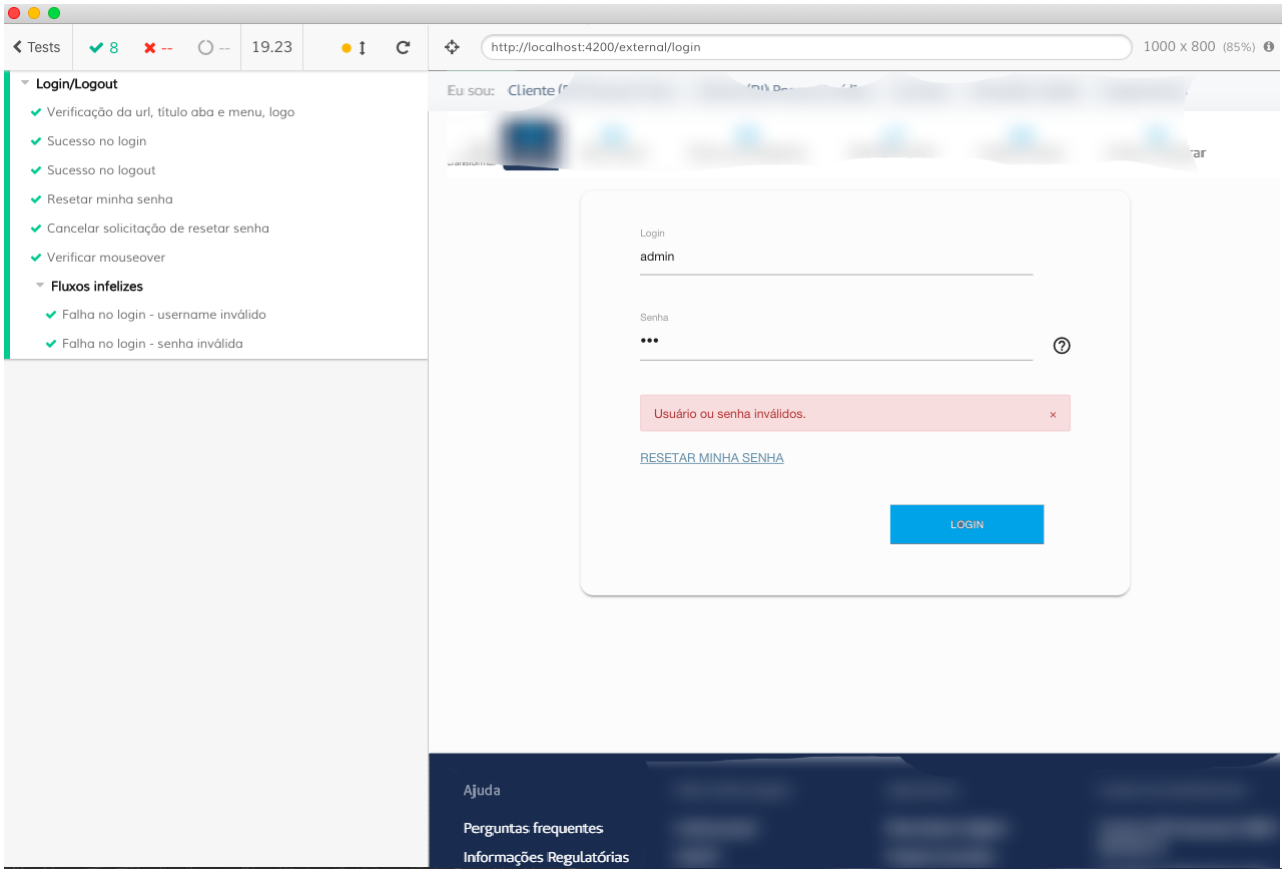
Na Figura 11 é apresentada a execução no modo *headless* onde, semelhante ao modo de execução com interface, a cor verde indica sucesso no teste, a cor vermelha demarca falha, e o tempo de execução é exibido em segundos. A desvantagem do modo *headless* é a maior dificuldade de analisar o cenário que levou o sistema a falhar, pois o mesmo não exibe em qual passo do teste ocorreu a falha.

Semelhantemente foram executados os testes automatizados para a funcionalidade *Login*. A execução com interface gráfica é representada na Figura 12, onde são exibidos na lateral os conjuntos de testes, status da execução e tempo total gasto na execução.

Na Figura 13 é apresentada a execução dos casos de teste de *Login* no modo *headless*.

Antes da finalização desta análise, o cliente informou a necessidade de inserir um novo tipo usuário na solicitação, sendo este o tipo *Corretor*. A necessidade veio do fato de que os usuários *corretores* estavam realizando contato com a empresa se identificando como usuário *Segurado* e, consequentemente solicitações importantes estavam sendo direcionadas ao setor errado da empresa, gerando assim uma demora na resposta e perda de vendas. Um novo arquivo de derivações foi passado para a equipe de desenvolvimento com a mesma estrutura do arquivo

Figura 12 – Execução dos testes de *Login* com interface gráfica



Fonte: Elaborado pela autora

inicial, porém com novas derivações.

De acordo com o cliente, o procedimento seria acrescentar o novo arquivo de derivações e novos endereços do serviço, e não haveria influência no sistema. A alteração foi tratada como "hot Fix" , termo utilizado internamente pela equipe de desenvolvimento para definir uma alteração urgente, geralmente com prazo de entrega de quarenta e oito horas, refletindo em um tempo curto e insuficiente dedicado à execução dos teste.

A estrutura da função utilizada na leitura, como apresentado na Seção 4.4, teve poucas mudanças, sendo alterada somente para receber tratar o perfil "*Corretor*" recebido como parâmetro. Após esta alteração foram realizadas novas execuções, cujos resultados serão descritos na Seção 5.2.

Figura 13 – Execução dos testes de *Login* em modo headless

```
Running: login/login.spec.js...                                    (1 of 1)

Login/Logout
  ✓ Verificação da url, título aba e menu, logo (4707ms)
  ✓ Sucesso no login (3725ms)
  ✓ Sucesso no logout (4092ms)
  ✓ Resetar minha senha (2212ms)
  ✓ Cancelar solicitação de resetar senha (1074ms)
  ✓ Verificar mouseover (791ms)
  Fluxos infelizes
    ✓ Falha no login - username inválido (2078ms)
    ✓ Falha no login - senha inválida (1531ms)

8 passing (21s)

(Results)

┌────────────────────────────────────────┐
│ Tests:        8                        │
│ Passing:      8                        │
│ Failing:      0                        │
│ Pending:      0                        │
│ Skipped:      0                        │
│ Screenshots:  0                        │
│ Video:        true                     │
│ Duration:     20 seconds               │
│ Spec Ran:     login/login.spec.js      │
└────────────────────────────────────────┘
```

Fonte: Elaborado pela autora

5 Resultados

Os critérios que serão analisados nesta seção são a cobertura de execução dos testes, o tempo de execução e o esforço empregado na implementação dos testes para as duas abordagens. Os resultados apresentados para as execuções automatizadas foram baseados nos dados das execuções em modo *headless* como descrito na seção 4.6 , visto que os mesmos demandaram menor tempo de execução.

5.1 Resultados dos Testes na Funcionalidade de *Login*

5.1.1 Primeira execução

Conforme podemos observar na Tabela 3 o tempo de implementação e execução do teste automatizado para esta funcionalidade foi maior que o tempo empreendido na execução manual. A quantidade de defeitos e casos de teste executados foi equivalente e concluímos que como visto na seção 3.1, os testes manuais são recomentados quando a funcionalidade será testada uma ou duas vezes e o sistema tem um curto prazo de desenvolvimento. Porém o Login é um requisito de alta importância no Projeto x, visto que se o usuário não estiver autenticado, ele não consegue utilizar a maioria das demais funcionalidades. Quando são realizadas alterações no software, é desejável que seja verificado se não houve a inserção de defeitos. Nessa situação a tarefa se torna repetitiva e devido ao prazo e demanda de tarefas não é possível refazer todos os testes.

Tabela 3 – Tempo de implementação e execução dos testes de Login

Tipo de execução	Automatizada	Manual
Esforço de implementação	2h 40m	0h
Esforço de execução	19s	5m 23s
Quantidade de testes executados	8	8
Quantidade de defeitos encontrados	1	1

Fonte: Elaborado pela autora

5.1.2 Segunda execução

Após a primeira execução testes automatizados de login foram executados automaticamente a cada conjunto de alterações inseridas no ambiente no qual são realizados os testes, o que ocorreu duas vezes ao dia durante a *sprint*. Os resultados dos testes ao final da *sprint*estão representados na tabela 4

Tabela 4 – Tabela de execução dos testes de Login durante uma sprint

Tipo de execução	Automatizada	Manual
Quantidade de execuções	20	2
Quantidade de testes executados	160	16
Quantidade de defeitos encontrados	1	0

Fonte: Elaborado pela autora

Como podemos observar na tabela 4 a cobertura de execução dos testes automatizados foi muito maior que a cobertura de execução dos testes manuais, e encontrou-se um defeito no sexto dia de execução. Não podemos afirmar que o defeito não seria revelado na execução manual, mas é certo que o mesmo ocorreria ao final da *sprint* e consequentemente a sua correção implicaria na possibilidade de inserção de defeitos. Com curto prazo disponível para testes o sistema poderia falhar no ambiente de produção. E como visto na seção 3.1, quanto antes um defeito é encontrado menor é o custo com sua correção.

5.2 Resultados dos Testes da Funcionalidade Fale Conosco

A funcionalidade fale conosco possui a maior quantidade de cenários de teste e em regressão nem sempre há tempo disponível para execução de todos, sendo priorizadas tarefas recém desenvolvidas. por este motivo em alguns resultados não há dados relevantes de execução manual.

5.2.1 Primeira execução

Os resultados da execução dos testes nesta funcionalidade estão representados na tabela 5. Como pode ser observado, novamente o tempo de implementação da automação de testes tem um grande esforço que, não existe na execução manual dos testes. Porém o esforço utilizado para a execução dos testes automatizados é aproximadamente quatro vezes menor que o manual. A quantidade de execuções e defeitos encontrados nesta execução foi a semelhante, fato que era esperado na primeira execução visto que foram utilizados os mesmos cenários, e com este dado entendemos que os testes automatizados estão com funcionamento de acordo com o esperado.

Tabela 5 – Tempo de implementação e execução testes Fale conosco

Tipo de execução	Automatizada	Manual
Esforço de implementação	6h 15m	0h
Esforço de execução	1h 3m	4h 23m 23s
Quantidade de testes executados	92	92
Quantidade de defeitos encontrados	5	5

Fonte: Elaborado pela autora

5.2.2 Segunda execução

Os defeitos encontrados foram repassados para correção pela equipe de desenvolvimento, neste ponto ainda restavam cinco dias para o fechamento da *sprint*, as tarefas de correção foram finalizadas restando dois dias para a entrega, com o tempo disponível foi realizada a regressão no sistema. Os resultados da segunda execução dos testes estão representados na tabela 6.

Tabela 6 – Tabela de execução testes Fale Conosco após correção

Tipo de execução	Automatizada	Manual
Quantidade de execuções	1	1
Quantidade de testes executados	92	29
Quantidade de defeitos encontrados	0	0

Fonte: Elaborado pela autora

A segunda execução dos testes automatizados foi realizada em paralelo com a regressão do sistema. Com o tempo disponível, foram cobertos vinte e seis cenários de teste executados de forma manual, enquanto com a execução automatizada foi possível cobrir 100% dos cenários, não havendo novos defeitos revelados em nenhuma das abordagens.

5.2.3 Terceira execução

Após a alteração "Hot fix" realizada no sistema como mencionado na seção 4, o conjunto de cenários de teste passou de noventa e dois, para cento e noventa e cinco. os dados estão representados na Tabela 7. Com poucas horas disponíveis, optou-se por executar os testes automatizados e reexecutar manualmente alguns cenários de forma exploratória, a execução revelou dois novos defeitos e levou cerca de duas horas e dezesseis minutos para ser finalizada.

Tabela 7 – Tabela de execução testes Fale Conosco após alteração

Tipo de execução	Automatizada	Manual
Quantidade de execuções	1	1
Quantidade de testes executados	195	10
Quantidade de defeitos encontrados	2	0

Fonte: Elaborado pela autora

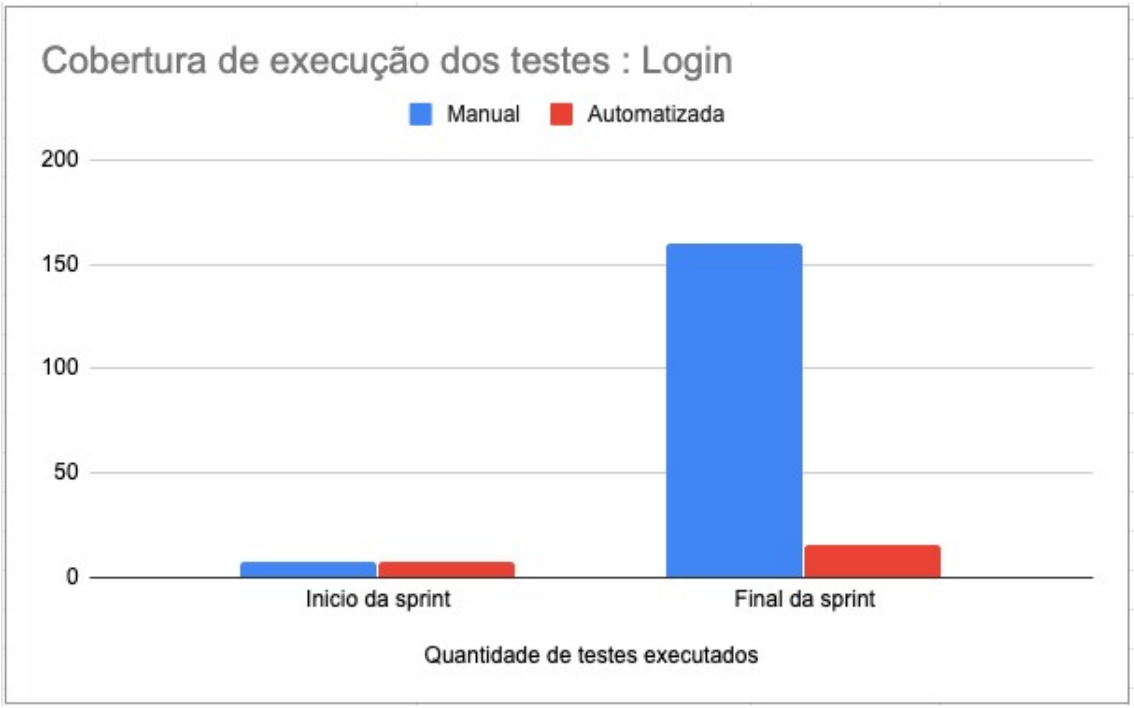
Os cenários que falharam foram passados para correção pela equipe de desenvolvimento. não houveram novas execuções dentro do período de desenvolvimento deste trabalho.

Após esta alteração não foi possível executar todos os testes automatizados em uma única execução sequencial devido ao tamanho da memória disponível no computador.

6 Conclusão

Após a analise dos dados, os resultados para o testes das funcionalidade de *Login* e *Fale Conosco* são ilustrados nas Figuras 14 e 15.

Figura 14 – Resultados da análise - *Login*

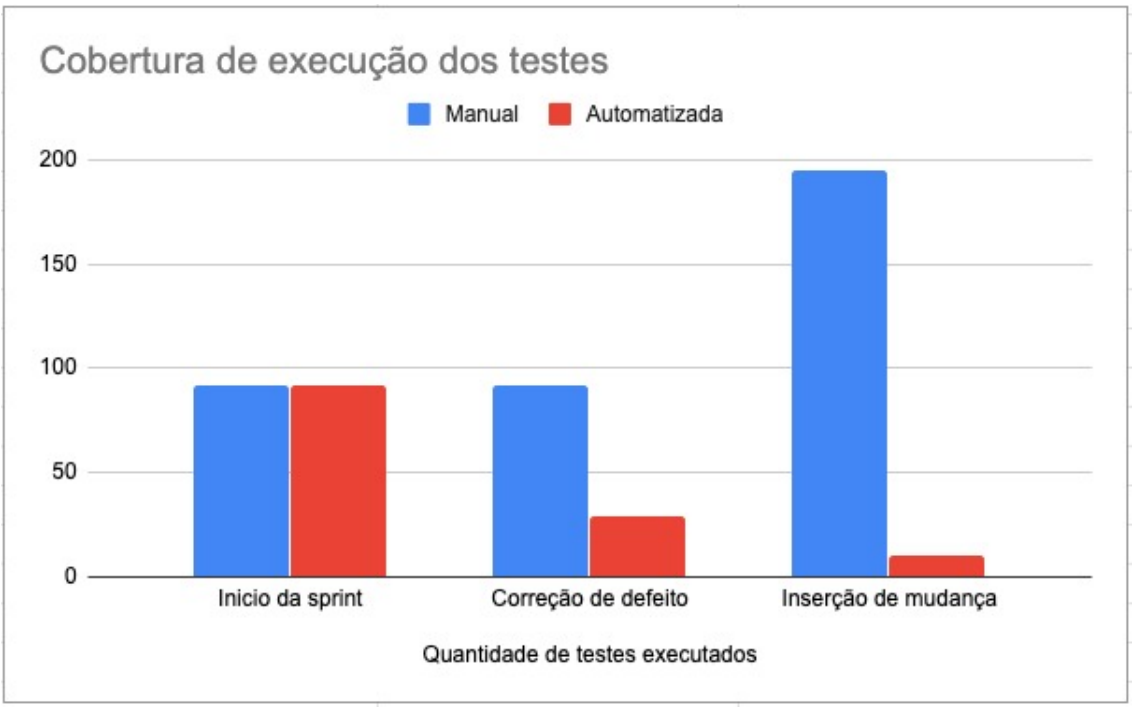


Fonte: Elaborado pela autora

Os dados nas figuras estão baseados no total de testes executados. Como podemos observar no gráfico da Figura 14 e nos resultados demonstrados na seção 5, com o mesmo tempo empreendido foi possível ter uma cobertura maior de execução de testes. Dessa forma, mesmo em funcionalidades simples a automação demonstra uma melhoria na cobertura e diminuição no esforço empreendido na execução dos testes. Além disso, observamos uma maior velocidade nas execuções e, consequentemente, os defeitos foram revelados mais cedo, diminuindo o custo na correção. Portanto, automatizar os testes é uma prática vantajosa sempre que possível, mesmo em funcionalidades simples. Apesar do tempo necessário para a implementação dos testes automatizados para algumas funcionalidades mais complexas como o *Fale Conosco* ser um ponto negativo, o tempo ganho na regressão do sistema torna possível manter uma cobertura de execução alta, como podemos observar no gráfico da Figura 15. Para um projeto com um número elevado de casos de teste e de atividades repetitivas, a automação não só reduz o tempo de execução, como aumenta a quantidade de casos de teste executados e revelando mais rapidamente as inconformidades. Dessa forma temos uma maior possibilidade de que o defeito seja identificado, corrigido e validado dentro do prazo de entrega do projeto. Uma observação importante no

desenvolvimento deste trabalho é em relação ao problema na execução causado pela quantidade de memória RAM disponível, conforme citado na Seção 5.2.3. Assim, destacamos que pode existir a necessidade de desenvolver conjuntos de teste com menor quantidade de cenários, evitando assim falhas de execução.

Figura 15 – Resultados da análise - *Fale Conosco*



Fonte: Elaborado pela autora

Referências

ALLIANCE, A. Agile 101.(2018). *Disponível em <https://www.agilealliance.org/agile101/> - acessado em novembro de 2019*, 2019.

BERGMAN, S. *Análise de Cobertura de Código*. 2018. Disponível em <https://phpunit.readthedocs.io/pt_BR/latest/code-coverage-analysis.html> - acessado em novembro, 2019.

BERNARDO, P. C.; KON, F. A importância dos testes automatizados. *Engenharia de Software Magazine*, v. 1, n. 3, p. 54–57, 2008.

CARRINGTON, D. Teaching software testing. In: ACM. *Proceedings of the 2nd Australasian conference on Computer science education*. [S.l.], 1997. p. 59–64.

CYPRESS. *Writing Your First Test*. 2019. Disponível em <https://docs.cypress.io/guides/getting-started/writing-your-first-test.html#Add-a-test-file> - acessado em outubro, 2019.

DELAMARO, M.; JINO, M.; MALDONADO, J. *Introdução ao teste de software (2ª Edição)*. [S.l.]: Elsevier Brasil, 2017.

FREITAS, D. A. *Testes de software: manuais X automatizados*. 2018. Disponível em <https://blog.cedrotech.com/testes-de-software-manuais-x-automatizados/> - acessado em novembro, 2019.

GOES, R. *Introdução ao Selenium WebDriver*. 2017. Disponível em <http://www.testesautomatizados.com.br/introducao-selenium-webdriver/> - acessado em novembro, 2019.

GOMES, M. V. M. *Teste Manual Vs. Automação de Teste*. 2017. Disponível em <https://medium.com/@mvmgmg/teste-manual-vs-automaç~ao-de-teste-8fcf4f9832b1> - acessado em novembro, 2019.

GRAHAM, D.; VEENENDAAL, E. V.; EVANS, I. *Foundations of software testing: ISTQB certification*. [S.l.]: Cengage Learning EMEA, 2008.

HIPTEST. *Collaboration Platform for Behavior Driven Development*. 2019. Disponível em <https://hiptest.com/> - acessado em outubro, 2019.

IEEE. *IEEE 829 Standard for Software and System Test Documentation*. New York/NY - USA, 1990.

LUCAS, S. *Manifesto do Teste Ágil*. 2017. Disponível em <https://agilizei.com/2017/03/20/manifesto-dos-testes/> - acessado em novembro, 2019.

MALDONADO, J. C. et al. *Aspectos teóricos e empíricos de teste de cobertura de software*. [S.l.]: ICMSC-USP, 1998.

MDS. *Conceito: Principais Medidas de Teste*. 2018. Disponível em <http://mds.cultura.gov.br/core.base_rup/guidances/concepts/key_measures_of_test_62253EE4.html> - acessado em novembro, 2019.

MYERS, G.; SANDLER, C.; BADGETT, T. *The Art of Software Testing*. Wiley, 2011. (ITPro collection). ISBN 9781118133156. Disponível em: <https://books.google.com.br/books?id=GjyEFPkMCwcC>.

NODEBR. *O que é Node.js?* 2016. Disponível em <http://nodebr.com/o-que-e-node-js/> - acessado em novembro, 2019.

OLIVEIRA, t. *Automação de testes: o que é, quando e por que automatizar*. 2018. Disponível em <https://medium.com/venturus/quais-as-raz~oes-para-automaç~ao-de-testes-c177cbd9397> - acessado em novembro, 2019.

PROTRACTOR. *Protractor*. 2014. Disponível em <https://www.protractortest.org/#/> - acessado em novembro, 2019.

SAMPAIO, J. *Implementando serviços com AngularJS*. 2015. Disponível em <https://www.devmedia.com.br/implementando-servicos-com-angularjs/32715> - acessado em novembro, 2019.

SOARES, I. *Desenvolvimento orientado por comportamento (BDD)*. 2011. Disponível em <https://www.devmedia.com.br/desenvolvimento-orientado-por-comportamento-bdd/21127> - acessado em novembro, 2019.

SOMMERVILLE, I. *Software Engineering*. 9th. ed. Boston/MA - USA: Addison-Wesley, 2011. ISBN 0-13-703515-2.

SPILLNER, A.; LINZ, T.; SCHAEFER, H. *Software testing foundations: a study guide for the certified tester exam*. [S.l.]: Rocky Nook, Inc., 2014.

VICENTE, G. *Page Objects – Padrão de projeto para organização de testes funcionais*. 2019. Disponível em <https://dextra.com.br/pt/page-objects-padrao-de-projeto-para-organizacao-de-testes-funcionais/> - acessado em outubro, 2019.

VINCENZI, A. M. R. et al. Automatização de teste de software com ferramentas de software livre. 2018.

Apêndices

APÊNDICE A – Casos de Teste - BDD

Neste apêndice estão os BDDs escritos para utilização na execução manual dos testes e como referência na escrita dos scripts de teste automatizados.

```
 Feature: derivação

   // BDD conjunto de testes para preenchimento dos dados do usuário Fale Conosco
   Scenario Outline: Preencher Formulário Fale Conosco

     Given [Dado que] Estou na pagina Fale conosco
      When [Quando] Eu informar o nome "userName"
      And [E] Eu informar o CPF "userCpf"
      And [E] Eu informar o telefone "userTel"
      And [E] Eu informar uma mensagem "faleMensagem"
      And [E] Eu informar um email "userEmail"
      And [E] Eu selecionar a opção "<ProdutoInput>"
      And [E] Eu selecionar a opção "<SubProdutoInput>"
      And [E] Eu selecionar a opção "<AssuntoInput>"
      And [E] Eu selecionar a opção "<SubAssuntoInput>"
      And [E] clicar no botão "Concluir"
      Then [Então] O sistema deverá gerar o "<Resultado>"

      Examples:
         | Dataset name                | userName | userCpf         | userTel         |
         | Todos os dados válidos      | userName | userCpf         | userTel         |
         | CPF inválido                | userName | userCpfinvalido | userTel         |
         | Nome sem preenchimento      | '  '     | userCpf         | userTel         |
         | Telefone sem preenchimento  | userName | userCpf         | '  '            |
         | Telefone inválido           | userName | userCpf         | userTelinvalido |
         | Email sem preenchimento     | userName | userCpf         | userTel         |
         | Email inválido              | userName | userCpf         | userTel         |
         | Mensagem sem preenchimento  | userName | userCpf         | userTel         |

         | userEmail         | faleMensagem | iAmInput | ProdutoInput |
         | userEmail         | faleMensagem | Segurado | Previdência  |
         | userEmail         | faleMensagem | Segurado | Previdência  |
         | userEmail         | faleMensagem | Segurado | Previdência  |
         | userEmail         | faleMensagem | Segurado | Previdência  |
         | userEmail         | faleMensagem | Segurado | Previdência  |
         | '  '              | faleMensagem | Segurado | Previdência  |
         | userEmailinvalido | faleMensagem | Segurado | Previdência  |
         | userEmail         | '  '         | Segurado | Previdência  |

         | SubProdutoInput    | SubAssuntoInput                 | Resultado |
         | Clube de Vantagens | Atualização de dados cadastrais | Protocolo |
         | Clube de Vantagens | Atualização de dados cadastrais | Falha     |
         | Clube de Vantagens | Atualização de dados cadastrais | Falha     |
         | Clube de Vantagens | Atualização de dados cadastrais | Falha     |
         | Clube de Vantagens | Atualização de dados cadastrais | Falha     |
         | Clube de Vantagens | Atualização de dados cadastrais | Falha     |
         | Clube de Vantagens | Atualização de dados cadastrais | Falha     |
         | Clube de Vantagens | Atualização de dados cadastrais | Falha     |
```

```
74
75
76
77      // BDD conjunto de testes para os serviços Fale Conosco
78
79      Scenario Outline: Preencher Formulário Fale Conosco
80
81         Given [Dado que] Estou na pagina Fale conosco
82         When [Quando] Eu informar o nome "userName"
83         And [E] Eu informar o CPF "userCpf"
84         And [E] Eu informar o telefone "userTel"
85         And [E] Eu informar uma mensagem "faleMensagem"
86         And [E] Eu informar um email "userEmail"
87         And [E] Eu selecionar a opção "<ProdutoInput>"
88         And [E] Eu selecionar a opção "<SubProdutoInput>"
89         And [E] Eu selecionar a opção "<AssuntoInput>"
90         And [E] Eu selecionar a opção "<SubAssuntoInput>"
91         And [E] clicar no botão "Concluir"
92         Then [Então] O sistema deverá gerar um Protocolo de atendimento
93
94         Examples:
95            | Dataset name | userName | userCpf | userTel |
96            | CT01         | userName | userCpf | userTel |
97            | CT02         | userName | userCpf | userTel |
98            | CT03         | userName | userCpf | userTel |
99            | CT04         | userName | userCpf | userTel |
100           | CT05         | userName | userCpf | userTel |
101           | CT06         | userName | userCpf | userTel |
102           | CT07         | userName | userCpf | userTel |
103           | CT08         | userName | userCpf | userTel |
104           | CT09         | userName | userCpf | userTel |
105           | CT10         | userName | userCpf | userTel |
106           | CT11         | userName | userCpf | userTel |
107           | CT12         | userName | userCpf | userTel |
108           | CT13         | userName | userCpf | userTel |
109           | CT14         | userName | userCpf | userTel |
110           | CT15         | userName | userCpf | userTel |
111           | CT16         | userName | userCpf | userTel |
112           | CT17         | userName | userCpf | userTel |
113           | CT18         | userName | userCpf | userTel |
114           | CT19         | userName | userCpf | userTel |
115           | CT20         | userName | userCpf | userTel |
116           | CT21         | userName | userCpf | userTel |
117           | CT22         | userName | userCpf | userTel |
118           | CT23         | userName | userCpf | userTel |
119           | CT24         | userName | userCpf | userTel |
120           | CT25         | userName | userCpf | userTel |
121           | CT26         | userName | userCpf | userTel |
122           | CT27         | userName | userCpf | userTel |
123           | CT28         | userName | userCpf | userTel |
124           | CT29         | userName | userCpf | userTel |
125           | CT30         | userName | userCpf | userTel |
126           | CT31         | userName | userCpf | userTel |
127           | CT32         | userName | userCpf | userTel |
128           | CT33         | userName | userCpf | userTel |
129           | CT34         | userName | userCpf | userTel |
130           | CT35         | userName | userCpf | userTel |
131           | CT36         | userName | userCpf | userTel |
132           | CT37         | userName | userCpf | userTel |
133           | CT38         | userName | userCpf | userTel |
134           | CT39         | userName | userCpf | userTel |
135           | CT40         | userName | userCpf | userTel |
136           | CT41         | userName | userCpf | userTel |
137           | CT42         | userName | userCpf | userTel |
138           | CT43         | userName | userCpf | userTel |
139           | CT44         | userName | userCpf | userTel |
140           | CT45         | userName | userCpf | userTel |
141           | CT46         | userName | userCpf | userTel |
142           | CT47         | userName | userCpf | userTel |
143           | CT48         | userName | userCpf | userTel |
144           | CT49         | userName | userCpf | userTel |
145           | CT50         | userName | userCpf | userTel |
146           | CT51         | userName | userCpf | userTel |
```

```
| CT52         | userName | userCpf | userTel |
| CT53         | userName | userCpf | userTel |
| CT54         | userName | userCpf | userTel |
| CT55         | userName | userCpf | userTel |
| CT56         | userName | userCpf | userTel |
| CT57         | userName | userCpf | userTel |
| CT58         | userName | userCpf | userTel |
| CT59         | userName | userCpf | userTel |
| CT60         | userName | userCpf | userTel |
| CT61         | userName | userCpf | userTel |
| CT62         | userName | userCpf | userTel |
| CT63         | userName | userCpf | userTel |
| CT64         | userName | userCpf | userTel |
| CT65         | userName | userCpf | userTel |
| CT66         | userName | userCpf | userTel |
| CT67         | userName | userCpf | userTel |
| CT68         | userName | userCpf | userTel |
| CT69         | userName | userCpf | userTel |
| CT70         | userName | userCpf | userTel |
| CT71         | userName | userCpf | userTel |
| CT72         | userName | userCpf | userTel |
| CT73         | userName | userCpf | userTel |
| CT74         | userName | userCpf | userTel |
| CT75         | userName | userCpf | userTel |
| CT76         | userName | userCpf | userTel |
| CT77         | userName | userCpf | userTel |
| CT78         | userName | userCpf | userTel |
| CT79         | userName | userCpf | userTel |
| CT80         | userName | userCpf | userTel |
| CT81         | userName | userCpf | userTel |
| CT82         | userName | userCpf | userTel |
| CT83         | userName | userCpf | userTel |
| CT84         | userName | userCpf | userTel |
| CT85         | userName | userCpf | userTel |
| CT86         | userName | userCpf | userTel |
| CT87         | userName | userCpf | userTel |
| CT88         | userName | userCpf | userTel |
| CT89         | userName | userCpf | userTel |
| CT90         | userName | userCpf | userTel |
| CT91         | userName | userCpf | userTel |
| CT92         | userName | userCpf | userTel |

| userEmail | faleMensagem | iAmInput | ProdutoInput      |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Previdência       |
| userEmail | faleMensagem | Segurado | Ramos Elementares |
| userEmail | faleMensagem | Segurado | Ramos Elementares |
| userEmail | faleMensagem | Segurado | Ramos Elementares
```

userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Ramos Elementares
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Saúde
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida
userEmail	faleMensagem	Segurado	Vida

SubProdutoInput	SubAssuntoInput
Clube de Vantagens	Atualização de dados cadastrais
Outros assuntos	null
PGBL	Informação do plano
PGBL	Parcelas pendentes
PGBL	Resgate
PGBL	2ª Via de Extrato
PGBL	2ª Via de
Previdência	null
Sugestões e Elogios	null

Clube de Vantagens	Atualização de dados cadastrais	
D&O	Informar no quadro de mensagem	
D&O	Informar no quadro de mensagem	
D&O	Informar no quadro de mensagem	
D&O	Informar no quadro de mensagem	
D&O	Informar no quadro de mensagem	
D&O	Informar no quadro de mensagem	
D&O	2ª Via de Apólice	
D&O	Reprogramação Boleto / Débito	
D&O	Informar no quadro de mensagem	
Empresarial	Informar no quadro de mensagem	
Empresarial	Informar no quadro de mensagem	
Empresarial	2ª Via de Apólice	
Empresarial	Reprogramação Boleto / Débito	
Empresarial	Informar no quadro de mensagem	
RCP Coletivo	Informar no quadro de mensagem	
RCP Coletivo	Informar no quadro de mensagem	
RCP Coletivo	2ª Via de Apólice	
RCP Coletivo	Reprogramação Boleto / Débito	
RCP Coletivo	Informar no quadro de mensagem	
RCP Individual	Informar no quadro de mensagem	
RCP Individual	Informar no quadro de mensagem	
RCP Individual	2ª Via de Apólice	
RCP Individual	Reprogramação Boleto / Débito	
RCP Individual	Informar no quadro de mensagem	
RCP Instituições	Informar no quadro de mensagem	
RCP Instituições	Informar no quadro de mensagem	
RCP Instituições	2ª Via de Apólice	
RCP Instituições	Reprogramação Boleto / Débito	
RCP Instituições	Informar no quadro de mensagem	
Residencial	Informar no quadro de mensagem	
Residencial	Informar no quadro de mensagem	
Residencial	2ª Via de Apólice	
Residencial	Reprogramação Boleto / Débito	
Residencial	Informar no quadro de mensagem	
AD	Assistência Domiciliar	
Clube de Vantagens	Atualização de dados cadastrais	
Saúde	Coparticipação	
Saúde	Dificuldades no App	
Saúde	Dúvidas de Atendimento	
Saúde	Elogio	
Saúde	Mau atendimento do Prestador	
Saúde	Autorização/Pedido de Medicação	
Saúde	Exames	
Saúde	Outros Assuntos	
Saúde	Terapias_Psicoterapia_	
Saúde	2ª via de Cartão	
Saúde	Alteração de dados cadastrais	
Saúde	Inclusão de segurado/dependente	
Saúde	Outros Assuntos	
Saúde	null	
Saúde	Comprovante / Baixa de Fatura	
Saúde	Outros Assuntos	
Saúde	null	
Saúde	null	
Saúde	Consultas/Avaliação	
Saúde	Exames	
Saúde	Procedimento Cirúrgico	
Saúde	Terapias	
Saúde	Devolução de documento	
Saúde	Dúvidas sobre valores pagos	
Saúde	Envio de Documentação Solicitada	
Saúde	Outros Assuntos	
Saúde	Reanálise de Reembolso	
Saúde	null	
Saúde	null	
Clube de Vantagens	Atualização de dados cadastrais	
Outros assuntos	null	
SERIT	Andamento seguro SERIT	
SERIT	Recusa sinistro seguro de VIDA	
Sugestões e Elogios	null	
Vida	Contratar Seguro de Vida	
Vida	null	

```
          | Vida                | Autorização de boleto por e-mail   |
          | Vida                | Carta de quitação anual de débitos |
          | Vida em Grupo       | Alteração de forma de pagamento    |
          | Vida em Grupo       | Andamento seguro VIDA              |
          | Vida em Grupo       | Cancelamento de proposta           |
          | Vida em Grupo       | Reabilitação de proposta           |
          | Vida em Grupo       | Recusa sinistro SERIT              |
          | Vida em Grupo       | Situação da proposta               |
          | Vida em Grupo       | 2ª Via de certificado              |
          | Vida em Grupo       | Alteração dos dados cadastrais     |
```

```
// BDD conjuntos de  Testes de Login

  Scenario Outline: Sucesso no login
     Given [Dado] que um usuário esta na tela de login
     When [Quando] informa seus dados de acesso
     And [E] clica em LOGIN
     Then [Então] deverá ser logado
     And [E] a tela de Inicial deverá ser exibida


  Scenario Outline: Sucesso no logout
     Given [Dado] que um usuário está logado
     When [Quando] clicar em logout
     Then [Então] deverá ser deslogado
     And [E] a tela de Login deverá ser exibida

  Scenario Outline: Resetar minha senha
     Given [Dado] que o usuário clica em resetar senha
     When [Quando] informar seu "<UserName>"
     And [E] clicar em recuperar senha
     Then [Então] uma mensagem de sucesso deverá ser exibida

  Scenario Outline: Cancelar solicitação de resetar senha
     Given [Dado] que o usuário clica em resetar senha
     When [Quando] informar seu "<UserName>"
     And [E] clicar em "Voltar"
     Then [Então] a tela de Login deverá ser exibida

  Scenario Outline: Verificar mouseover
     Given [Dado] que um usuário esta na tela de login
     When [Quando] mantiver o mouse sobre o icone de interrogação
     Then [Então] Uma mensagemd e ajuda deverá ser exibida


  Scenario Outline: Falha no login - username inválido
     Given [Dado] que um usuário esta na tela de login
     When [Quando] informa um "<UserName>" inválido
     And [E] clica em LOGIN
     Then [Então] Então uma mensagem de erro deverá ser exibida



  Scenario Outline: Falha no login - senha inválida
     Given [Dado] que um usuário esta na tela de login
     When [Quando] informa um "<UserPassword>" inválido
     And [E] clica em LOGIN
     Then [Então] Então uma mensagem de erro deverá ser exibida
  // BDD conjuntos de  Testes de Login


  Scenario : Sucesso no login
     Given [Dado] que um usuário esta na tela de login
     When [Quando] informa seus dados de acesso
     And [E] clica em LOGIN
     Then [Então] deverá ser logado
     And [E] a tela de Inicial deverá ser exibida
```

```
439
440
441     Scenario : Sucesso no logout
442        Given [Dado] que um usuário está logado
443        When [Quando] clicar em logout
444        Then [Então] deverá ser deslogado
445        And [E] a tela de Login deverá ser exibida
446
447     Scenario : Resetar minha senha
448        Given [Dado] que o usuário clica em resetar senha
449        When [Quando] informar seu "<UserName>"
450        And [E] clicar em recuperar senha
451        Then [Então] uma mensagem de sucesso deverá ser exibida
452
453     Scenario : Cancelar solicitação de resetar senha
454        Given [Dado] que o usuário clica em resetar senha
455        When [Quando] informar seu "<UserName>"
456        And [E] clicar em "Voltar"
457        Then [Então] a tela de Login deverá ser exibida
458
459     Scenario : Verificar mouseover
460        Given [Dado] que um usuário esta na tela de login
461        When [Quando] mantiver o mouse sobre o icone de interrogação
462        Then [Então] Uma mensagemd e ajuda deverá ser exibida
463
464
465     Scenario : Falha no login - username inválido
466        Given [Dado] que um usuário esta na tela de login
467        When [Quando] informa um "<UserName>" inválido
468        And [E] clica em LOGIN
469        Then [Então] Então uma mensagem de erro deverá ser exibida
470
471
472
473     Scenario : Falha no login - senha inválida
474        Given [Dado] que um usuário esta na tela de login
475        When [Quando] informa um "<UserPassword>" inválido
476        And [E] clica em LOGIN
477        Then [Então] Então uma mensagem de erro deverá ser exibida
```

APÊNDICE B – Código Fonte

Neste Apendice é apresentado o código utilizado na produção deste trabalho:

Os dados referenciados como Data: ex: "loginData.userPassord"
Estão nas paginas:

- faleData.json
- preencheData.json

As Mesmas não estão presentes neste documento por conterem Dados pessoais.

A estrutura das páginas de dados json é a seguinte:

```
{
    "userCpf": "000.000.000-00",
    "userName": "Usuário nome",
    "userPassword": "senha do Usuario",
    "userTel": "16999999999",
    "userEmail": "Teste@teste.com",
    "faleMensagem": "Mensagem Teste "

}
```

Os cenários implementados aqui estão presentes no Apêndice A.

```
// Código Script Para Testar a funcionalidade Login #script1
//Script dos Casos de Teste

/// <reference types="Cypress" />
import { LoginPage } from "../../support/pageObjects/LoginPage";

let loginData;
let loginPage = new LoginPage();

describe('Login/Logout', () => {
    // runs once before all tests in the block
    before(() => {
        // Acessar os dados do arquivo dataFile na pasta fixtures
        cy.fixture('loginData').then((data) => {
            loginData = data;
        })
    })


    it('Verificação da url, título aba e menu, logo', () => {
        cy.visit('/');
        cy.location('pathname').should('contain', loginData.loginURL)
        cy.title().should('contain', loginData.tabTitle);
        loginPage.getLoginPageTitle().should('contain', loginData.loginTitle);
        loginPage.getLogo().should('be.visible');
    });

    it('Sucesso no login', () => {
        cy.visit('/');
        loginPage.getUsername().type(loginData.loginAutomation);
        loginPage.getPassword().type(loginData.userPassword);
        loginPage.getLoginBtn().click()
    //verifica se o sistema redireciona para a tela de Usuários após logar

    });

    it('Sucesso no logout', () => {
        // LOGIN
        cy.visit('/');
        loginPage.getUsername().type(loginData.adminUsername);
        loginPage.getPassword().type(loginData.userPassword);
        loginPage.getLoginBtn().click()


        // LOGOUT
        loginPage.getMenuDropdown().click();
        loginPage.getLogoutBtn().click({force:true})
```

```
74          cy.location('pathname').should('eq', '/external/login')
75      });
76
77
78      it('Resetar minha senha', () => {
79          cy.visit('/');
80          // RESETAR SENHA
81          loginPage.getResetPasswordLink().click();
82          cy.location('pathname').should('eq', '/external/password-recovery')
83          loginPage.getResetPasswordSubtitle().should('contain', loginData.
            resetPasswordSubtitle)
84          loginPage.getLoginBtn().should('not.be.visible')
85          loginPage.getUsername().type(loginData.managerUsername);
86          loginPage.getResetPasswordBtn().click();
87          loginPage.getResetPasswordSucessMsg().should('contain', loginData.
            resetPasswordSucessMsg)
88      })
89
90
91      it('Cancelar solicitação de resetar senha', () => {
92          cy.visit('/');
93          loginPage.getResetPasswordLink().click();
94          loginPage.getCancelResetPasswordBtn().click();
95          cy.location('pathname').should('eq', '/external/login')
96          loginPage.getLoginBtn().should('be.visible')
97      })
98
99
100     it('Verificar mouseover', () => {
101         cy.visit('/');
102         loginPage.getHelpBtn().trigger('mouseover')
103         loginPage.getMgmBoxHelp().should('be.visible')
104     })
105
106
107     // FLUXOS INFELIZES
108     describe('Fluxos infelizes', () => {
109         it('Falha no login - username inválido', () => {
110             cy.visit('/');
111             loginPage.getUsername().type(loginData.invalidUsername);
112             loginPage.getPassword().type(loginData.userPassword);
113             loginPage.getLoginBtn().click()
114             loginPage.getAlertDangerMsg().should('contain', loginData.
                invalidUsernameErrorMsg)
115                 .and('be.visible');
116         });
117
118
119         it('Falha no login - senha inválida', () => {
120             cy.visit('/');
121             loginPage.getUsername().type(loginData.adminUsername);
122             loginPage.getPassword().type(loginData.invalidPassword);
123             loginPage.getLoginBtn().click()
124             loginPage.getAlertDangerMsg().should('contain', loginData.
                invalidPassowordErrorMsg)
125                 .and('be.visible');
126
127         });
128     })
129
130 })
131
132
133 // Pagina de encapsulamento de métodos PageObjects referente ao #script1
134
135 export class LoginPage {
136
137     getLogo() {
138         return cy.get('.b-logo')
139     }
140
141     getLoginPageTitle() {
142         return cy.get('.mm-text > a')
```

```
143         }
144
145         getUsername() {
146             return cy.get('#username')
147         }
148
149         getPassword() {
150             return cy.get('#password')
151         }
152
153         getLoginBtn() {
154             return cy.get('#login-button')
155         }
156
157         getKeepLoggedCheckbox() {
158             return cy.get('#keepLogged')
159         }
160
161         getHelpBtn() {
162             return cy.get('.help-btn')
163         }
164
165         getMgmBoxHelp() {
166             return cy.get('.mm-box')
167         }
168
169         getMenuDropdown() {
170             return cy.get('#nav-user-dropdown')
171         }
172
173         getLogoutBtn() {
174             return cy.get('#nav-logout')
175
176         }
177
178         getAlertDangerMsg() {
179             return cy.get('.alert-danger')
180         }
181
182         getUsersPageTitle() {
183             return cy.get('.title-page')
184         }
185
186         getResetPasswordLink() {
187             return cy.get(':nth-child(4) > a')
188
189         }
190
191         getResetPasswordTitle() {
192             return cy.get('body > div.row.center-align > span.title-page')
193         }
194
195         getResetPasswordSubtitle() {
196             return cy.get('.subtitle-page')
197         }
198         getCancelResetPasswordBtn() {
199             return cy.get('.left > a > .btn')
200         }
201         getResetPasswordBtn() {
202             return cy.get('.col.right > .btn')
203             cy.get('.left > a > .btn')
204         }
205         getResetPasswordSucessMsg() {
206             return cy.get('.alert-success')
207         }
208
209     }
210
211     ==========================================================================================
212     // Código Script Para preenchimento das combinações de segmentos #script2
213     //Script dos Casos de Teste
214
215     /// <reference types = "Cypress" />
```

```
216
217   //importação dos PageObjects
218   import { FalePage } from '../support/pageObjects/FalePage.js'
219   import { PreenchePage } from '../support/pageObjects/PreenchePage.js'
220
221   let preencheData;
222   let falePage2 = new FalePage();
223
224   describe('Testes: Campos de dados Usuário Fale Conosco', () => {
225       // runs once before all tests in the block
226       before(() => {
227           // Acessar os dados do arquivo dataFile na pasta fixtures
228           cy.fixture('preencheData').then((data) => {
229               preencheData = data;
230           })
231
232       })
233
234
235       it('Todos os dados válidos', () => {
236
237           cy.visit('/')
238
239           falePage2.getInput().select(preencheData.userTipo);
240           falePage2.getCpf().type(preencheData.userCpfvalido);
241           falePage2.getName().type(preencheData.userName);
242           falePage2.getEmail().type(preencheData.userEmail);
243           falePage2.getTel().type(preencheData.userTel);
244           preDeriva();
245           falePage2.getMsg().type(preencheData.faleMensagem);
246           cy.get('.pl-2 > .btn').click()
247           cy.wait(20000)
248           cy.get('.description').should('be.visible')
249
250       })
251
252
253       it('CPF inválido', () => {
254
255           cy.visit('/')
256           falePage2.getInput().select(preencheData.userTipo);
257           falePage2.getCpf().type(preencheData.userCpfinvalido);
258           falePage2.getName().type(preencheData.userName);
259           cy.get('.error-message').should('be.visible');
260       })
261
262
263       it('Nome sem preenchimento', () => {
264           cy.visit('/')
265           falePage2.getInput().select(preencheData.userTipo);
266           falePage2.getCpf().type(preencheData.userCpfvalido);
267           falePage2.getEmail().type(preencheData.userEmail);
268           falePage2.getTel().type(preencheData.userTel);
269           preDeriva();
270           falePage2.getMsg().type(preencheData.faleMensagem);
271           cy.get('.pl-2 > .btn').click()
272           cy.get('.toast-title').should('be.visible');
273
274       })
275
276
277       it('telefone sem preenchimento', () => {
278           cy.visit('/')
279           falePage2.getInput().select(preencheData.userTipo);
280           falePage2.getCpf().type(preencheData.userCpfvalido);
281           falePage2.getName().type(preencheData.userName);
282           falePage2.getEmail().type(preencheData.userEmail);
283           preDeriva();
284           falePage2.getMsg().type(preencheData.faleMensagem);
285           cy.get('.pl-2 > .btn').click()
286           cy.get('.toast-title').should('be.visible');
287           })
288
```

```
289
290         it('telefone inválido', () => {
291
292             cy.visit('/')
293             falePage2.getInput().select(preencheData.userTipo);
294             falePage2.getCpf().type(preencheData.userCpfvalido);
295             falePage2.getName().type(preencheData.userName);
296             falePage2.getEmail().type(preencheData.userEmail);
297             falePage2.getTel().type(preencheData.userTelinvalido);
298             preDeriva();
299             falePage2.getMsg().type(preencheData.faleMensagem);
300             cy.get('.pl-2 > .btn').click()
301             cy.wait(20000)
302             cy.get('.description').should('not.be.visible')
303         })
304
305
306         it('Email sem preenchimento', () => {
307
308             cy.visit('/')
309             falePage2.getInput().select(preencheData.userTipo);
310             falePage2.getCpf().type(preencheData.userCpfvalido);
311             falePage2.getName().type(preencheData.userName);
312             falePage2.getTel().type(preencheData.userTel);
313             preDeriva();
314             falePage2.getMsg().type(preencheData.faleMensagem);
315             cy.get('.pl-2 > .btn').click()
316             cy.get('.toast-title').should('be.visible');
317         })
318
319
320         it('Email inválido', () => {
321             cy.visit('/')
322             falePage2.getInput().select(preencheData.userTipo);
323             falePage2.getCpf().type(preencheData.userCpfvalido);
324             falePage2.getName().type(preencheData.userName);
325             falePage2.getEmail().type(preencheData.userEmailinvalido);
326             falePage2.getTel().type(preencheData.userTel);
327             cy.get('.error-message').should('be.visible');
328         })
329
330
331         it('Mensagem sem preenchimento', () => {
332
333             cy.visit('/')
334             falePage2.getInput().select(preencheData.userTipo);
335             falePage2.getCpf().type(preencheData.userCpfvalido);
336             falePage2.getName().type(preencheData.userName);
337             falePage2.getEmail().type(preencheData.userEmail);
338             falePage2.getTel().type(preencheData.userTel);
339             preDeriva();
340             cy.get('.pl-2 > .btn').click()
341             cy.get('.toast-title').should('be.visible');
342         })
343     })
344
345
346
347
348     function preDeriva() {
349
350         //Função para preencher os segmentos nos seletores
351         falePage2.getProduto().select(preencheData.produto)
352         falePage2.getSubProduto().select(preencheData.subproduto)
353         falePage2.getAssunto().select(preencheData.assunto)
354         falePage2.getSubAssunto().select(preencheData.subAssunto)
355     }
356
357
358     // Pagina de encapsulamento de métodos PageObjects referente ao #script2
359
360     export class PreenchePage {
361
```

```
362     getCpfInvalidoMsg() {
363       return cy.get('u-input-com-radio-directive > .input-box > .error-message')
364     }
365
366     getNameVazio() {
367       return cy.get('.col-md-8 > u-input-field > .input-box > .error-message')
368     }
369
370     getTelvazio() {
371
372       return cy.get('u-input-field-directive > .input-box > .error-message')
373     }
374
375     getEmailinvalidoVazio() {
376       return cy.get(':nth-child(6) > .col-sm-12 > u-input-field > .input-box >
        .error-message')
377     }
378
379     getMsgVazio() {
380       return cy.get('.flex-column > .error-message')
381
382     }
383
384
385   }
386   ==========================================================================================
387
388   // Código Script Para preenchimento das combinações de segmentos #script3
389   //Script dos Casos de Teste
390
391   /// <reference types = "Cypress" />
392
393   import { FalePage } from '../support/pageObjects/FalePage.js'
394   import { derivacao } from '../support/utils'
395
396   let faleData;
397   let falePage = new FalePage();
398   let teste = derivacao('Cliente');
399
400
401
402   for (let i = 0; i < 5; i++) {
403
404     describe('Testes: Tipo Usuário Segurado', () => {
405       // runs once before all tests in the block
406       before(() => {
407         // Acessar os dados do arquivo dataFile na pasta fixtures
408         cy.fixture('faleData').then((data) => {
409           faleData = data;
410         })
411       })
412
413       it('Fale Conosco', () => {
414
415         cy.visit('/')
416
417         falePage.getInput().select(faleData.userTipo)
418
419         prebase();
420
421         falePage.getProduto().select(teste[i][0])
422         falePage.getSubProduto().select(teste[i][1])
423         cy.log(teste[i][2])
424
425         if ((teste[i][2]) != '') {
426           falePage.getAssunto().select(teste[i][2])
427         }
428
429         if ((teste[i][3]) != '') {
430           falePage.getSubAssunto().select(teste[i][3])
431         }
432
433         falePage.getMsg().type(faleData.faleMensagem)
```

```
434         cy.get('.pl-2 > .btn').click()
435         cy.wait(20000)
436         cy.get('.description').should('be.visible')
437
438
439       })
440
441
442     })
443
444   }
445
446   function prebase() {
447     // more statements
448     falePage.getCpf().type(faleData.userCpf)
449     falePage.getName().type(faleData.userName)
450     falePage.getTel().type(faleData.userTel)
451     falePage.getEmail().type(faleData.userEmail)
452   }
453
454   // Pagina de encapsulamento de métodos PageObjects referente ao #script3
455
456   export class FalePage {
457
458     getCpf() {
459       return cy.get('#cpfCnpjInput')
460     }
461     getName() {
462       return cy.get('#nomeCompletoInput')
463     }
464     getTel() {
465       return cy.get('#telefoneInput')
466     }
467
468     getEmail() {
469       return cy.get('#emailInput')
470     }
471
472     getMsg() {
473       return cy.get('#mensagem-fale-conoscoInput')
474     }
475
476     getInput() {
477       return cy.get('#iAmInput')
478       }
479
480     getProduto() {
481       return cy.get('#ProdutoInput')
482     }
483
484     getSubProduto() {
485       return cy.get('#SubProdutoInput')
486     }
487
488     getAssunto() {
489       return cy.get('#AssuntoInput')
490     }
491
492     getSubAssunto() {
493       return cy.get('#SubAssuntoInput')
494     }
495
496   }
497
498   ==========================================================================
499
500   // Código em javascript utilizado para leitura do arquivo de derivações
501
502   var produto = ""
503   var subProduto = ""
504   var assunto = ""
505   var subAssunto = ""
506   var derivacoesMontadas = []
```

```
507
508   function  montaDerivacao() {
509       derivacoesMontadas.push([produto, subProduto, assunto, subAssunto])
510   }
511
512   function getDerivacoes() {
513       var json = require("D:/Nova pasta/ProjectX/derivacoes.json")
514       return JSON.parse(JSON.stringify(json));
515   }
516
517   function getTipoEListaParamDerivacao(derivacao) {
518       var tipo = derivacao["tipoDerivacao"]
519
520       if (tipo == "Template") {
521           return tipo, ""
522       }
523
524       return [tipo, derivacao["listaParamDerivacao"].split("|")]
525   }
526
527
528   function getProdutos(derivacao) {
529       var values = getTipoEListaParamDerivacao(derivacao)
530       var tipo = values[0]
531       var listaParam = values[1]
532
533       if (tipo === "SubProduto") {
534           for (var param in listaParam) {
535               subProduto = listaParam[param]
536               getSubProdutos(derivacao[subProduto.replace("/", "-")])
537           }
538       } else if (tipo == "Assunto") {
539           for (var param in listaParam) {
540               assunto = listaParam[param]
541               subProduto = ""
542               getAssunto(derivacao[assunto.replace("/", "-")])
543           }
544       } else {
545           subProduto = ""
546           assunto = ""
547           subAssunto = ""
548           montaDerivacao()
549       }
550   }
551
552   function getSubProdutos(derivacao) {
553       var values = getTipoEListaParamDerivacao(derivacao)
554       var tipo = values[0]
555       var listaParam = values[1]
556
557       if (tipo == "Assunto") {
558           for (var param in listaParam) {
559               assunto = listaParam[param]
560               getAssunto(derivacao[assunto.replace("/", "-")])
561           }
562       } else {
563           assunto = ""
564           subAssunto = ""
565           montaDerivacao()
566       }
567   }
568
569   function getAssunto(derivacao) {
570       var values = getTipoEListaParamDerivacao(derivacao)
571       var tipo = values[0]
572       var listaParam = values[1]
573
574       if (tipo == "SubAssunto") {
575           for (var param in listaParam) {
576               subAssunto = listaParam[param]
577               montaDerivacao()
578           }
579       } else {
```

```
580             subAssunto = ""
581             montaDerivacao()
582         }
583     }
584 
585     function getDerivacaoPerfil(perfil, derivacao) {
586         derivacoesMontadas = []
587         var values = getTipoEListaParamDerivacao(derivacao[perfil])
588         var tipo = values[0]
589         var listaParam = values[1]
590        // console.log(listaParam)
591 
592         // console.log(derivacao[perfil]["Previdência"])
593 
594         if (tipo == "Produto") {
595             for (var param in listaParam) {
596                 produto = listaParam[param]
597                 getProdutos(derivacao[perfil][listaParam[param].replace("/", "-")])
598             }
599         }
600     }
601 
602     export function derivacao(perfil) {
603         getDerivacaoPerfil(perfil, getDerivacoes())
604         console.log(derivacoesMontadas);
605         return derivacoesMontadas;
606     }
607 
608 
609 
```