Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUCAS PETRUCI PENGA

ESTUDO DO FRAMEWORK DE COBERTURA DE TESTE SELENIUM

Assis/SP
2016

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LUCAS PETRUCI PENGA

ESTUDO DO FRAMEWORK DE COBERTURA DE TESTE SELENIUM

Projeto de pesquisa apresentado ao curso de Ciências da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Lucas Petruci Penga
Orientador(a): Esp. Domingos de Carvalho Villela Junior

Assis/SP
2016

FICHA CATALOGRÁFICA

PENGA, Lucas Petruci.
Estudo do framework de cobertura de teste Selenium / Lucas Petruci Penga. Fundação Educacional do Município de Assis –FEMA – Assis, 2016.
Numero de páginas. 35

Orientador: Esp. Domingos de Carvalho Villela Junior
Trabalho de conclusão de curso – Instituto Municipal de Ensino Superior de Assis - IMESA

1. Estudo e Conceitos. 2. Framework. 3. Selenium. 4.Caso de Uso

CDD: 001.6
Biblioteca da FEMA

ESTUDO DO FRAMEWORK DE COBERTURA DE TESTE SELENIUM

LUCAS PETRUCI PENGA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Esp. Domingos de Carvalho Villela Junior

Examinador: Dr. Almir Rogério Camolesi

Assis/SP
2016

RESUMO

A cobertura de teste é uma técnica que auxilia na produção de software de alta qualidade e cada vez mais desenvolvedor estão adotando essa prática. Porém ao fazer uso da cobertura alguns cuidados devem ser tomados, pois a cobertura de teste quando usada de maneira incorreta acarreta em uma falsa métrica de qualidade. Esse trabalho auxilia o desenvolvedor a fazer um melhor uso da cobertura de teste, apontando seus pontos fracos e fortes através de um estudo realizado conceitos de cobertura de teste.

.

Palavras-chave: Cobertura de teste; Engenharia de Software; Teste de software; Selenium.

ABSTRACT

The test coverage is a technique that assists in high production software quality and more and more developers are adopting this practice. But to make use cover some care must be taken because the test coverage when used improperly brings in a alse quality metric. This work helps developers to make better use Test coverage, pointing their strengths and weaknesses through a study conducted test coverage concepts.

.

Keywords: Test coverage; Software engineering; Software Testing; Selenium.

LISTA DE ILUSTRAÇÕES

Figura 1 - Download Selenuim IDE. 17

Figura 2 - Menu Ferramentas..... 18

Figura 3 - Janela Selenium IDE..... 18

Figura 4 – Erro..... 24

Figura 5 – Erro 2..... 24

Figura 6 – Erro 3..... 24

Figura 7 – Caso de Uso Teste1..... 25

Figura 7 – Caso de Uso Teste2..... 25

Figura 9 – Somatematica..... 26

Figura 10 – Novo Comando..... 27

Figura 11 – Teste 1..... 28

Figura 12 – Retorno Teste 1..... 29

Figura 13 – Jogo de somar..... 30

Figura 14 – Teste 2..... 31

Figura 15 – Retorno Teste 2..... 32

LISTA DE TABELAS

Tabela 1 - Exemplo Script Syntax. 20

Tabela 2 - Exemplo Script Syntax 2..... 20

Tabela 3 - Exemplo armazenar variáveis..... 22

Tabela 4 – acessar o valor de uma variável..... 22

Tabela 5 – acessar o valor de uma variável 2..... 22

SUMÁRIO

1. INTRODUÇÃO 9

1.1. OBJETIVOS 9

1.1.1. OBJETIVO GERAL 9

1.1.2. OBJETIVOS ESPECÍFICOS 9

1.2. JUSTIFICATIVA 9

1.3. MOTIVAÇÃO 10

1.4. PERSPECTIVA DE CONTRIBUIÇÃO 10

1.5. MÉTODO DE PESQUISA 10

1.6. ESTRUTURA DE TRABALHO 11

***2.* ESTUDO E CONCEITOS 12**

2.1. ENGENHARIA DE SOFTWARE 12

2.2. TESTE DE SOFTWARE 13

2.3. COBERTURA DE TESTE 14

3. FRAMEWORK 16

3.1. SELENIUM 16

3.2. INSTALAÇÃO SELENIUM IDE 17

3.3. COMANDOS SELENIUM 19

3.4. ARMAZENAR COMANDOS E VARIÁVEIS DE SELENIUM 21

3.5. SOLUÇÃO DE PROBLEMAS 23

4. ESTUDO DE CASO 25

4.1. IDENTIFICANDO E ACESSANDO LINKS 26

4.2. ARMAZENANDO VALORES 29

5. CONCLUSÃO 33

6. REFERÊNCIAS 34

1. INTRODUÇÃO

Das diversas fases associadas a engenharia de software, o teste é a fase indispensável para validar e verificar software em desenvolvimento. No cenário atual da engenharia de software, os testes são tão importantes quanto o processo de desenvolvimento. Uma das questões que deve se levar em consideração na execução de um teste é medir sua eficácia, existem diversas ferramentas que auxiliam nessa tarefa, basicamente cobertura de teste é qualquer medida relacionada a um requisito ou a um critério de implementação do código, como a verificação de casos de uso ou a execução de todas as linhas de código.

Esse estudo busca somar conhecimento relacionado a cobertura de teste, através dos dados coletados.

1.1. OBJETIVOS

1.1.1. OBJETIVO GERAL

Este trabalho tem como objetivo reunir informações sobre o funcionamento de frameworks que auxiliam na cobertura de teste, analisar os dados coletados e através destes dados apresentar um conteúdo de fácil entendimento que agregue conhecimento sobre teste de software, assim contribuindo para a comunidade de Engenharia de software.

1.1.2. OBJETIVOS ESPECÍFICOS

Explicar o funcionamento do Selenium, criar um tutorial de instalação no Sistema operacional Windows. Demonstrar com a pesquisa um trabalho que some conhecimento sobre cobertura de teste utilizando a ferramenta Selenium IDE.

1.2. JUSTIFICATIVA

A importância deste trabalho consiste em apresentar um conteúdo que seja de fácil entendimento sobre o framework de cobertura de teste Selenium, abordando seu funcionamento, instalação, e comandos básicos.

1.3. MOTIVAÇÃO

Atualmente, cobertura de teste é um tema em que se encontra uma grande quantidade de fóruns discutindo sobre o assunto, porém não temos um material que nos permita iniciar com a cobertura de teste de uma forma simples e de fácil entendimento. Sendo assim veio a necessidade de criar um conteúdo de fácil entendimento, permitindo assim uma melhor utilização da ferramenta e de suas funções.

1.4. PERSPECTIVA DE CONTRIBUIÇÃO

Somar conhecimento para aqueles que desejam se aprofundar na engenharia de software, expor o conteúdo sobre o framework Selenium, ensinar como instalar, contribuir diretamente para a comunidade de engenharia de software cobertura de teste.

1.5. MÉTODO DE PESQUISA

Segundo Gil (2010, p.27) esta pesquisa pode se caracterizar como:

> "*Pesquisa básica pura: pesquisas destinadas unicamente a ampliação do conhecimento, sem qualquer preocupação com seus possíveis benefícios.*"

Para dar sustentação conceitual a esta pesquisa, foi feita uma revisão bibliográfica buscando encontrar conceitos mais plausíveis ao tema abordado.

> "*A pesquisa bibliográfica é desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos. Embora em quase*

> *todos os estudos seja exigido algum tipo de trabalho dessa natureza, há pesquisas desenvolvidas exclusivamente a partir de fontes bibliográficas (GIL, 2002, p. 44).*"

Ao passo inicial da pesquisa fez-se então uma revisão de literatura, em que buscou identificar:

ENGENHARIA DE SOFTWARE

TESTES DE SOFTWARE

COBERTURA DE TESTE

Para que se efetive o objetivo descrito no capítulo "FRAMEWORK", traceja-se o aporte metodológico, apresentando o Framework, sua instalação e Funcionamento deste Framework.

1.6. ESTRUTURA DE TRABALHO

O presente trabalho para melhor organização é subdividido em capítulos sendo eles organizados da seguinte forma:

No primeiro capitulo é possível obter a introdução do trabalho bem como sua finalidade.

No segundo capitulo temos uma breve introdução sobre engenharia de software, conceitos de teste de software e cobertura de teste.

O capítulo terceiro nos mostra os dados coletados sobre o framework Selenium IDE.

O Quarto capítulo nos mostra o funcionamento do framework Selenium IDE.

No Quinto capítulo é apresentada à conclusão que obtive durante toda a pesquisa deste trabalho.

2. ESTUDO E CONCEITOS

2.1. ENGENHARIA DE SOFTWARE

A Engenharia de *Software* visa a criação de *softwares* que atendam às necessidades de pessoas e instituições. Para isso, usa o conhecimento científico, técnico e gerencial, tanto teórico quanto prático.

Segundo Sommerville (2007, p.3):

> "A engenharia de *Software* é um ramo de engenharia cujo foco é desenvolvimento dentro de custos adequados de sistemas de *software* de alta qualidade."

O conceito de engenharia de *software* foi proposto em 1968 em meio da chamada 'Crise de *software'*. A crise resultava da introdução de um novo tipo de hardware baseado em circuitos integrados, seu poder fez das aplicações até então não realizáveis, propostas visíveis. Com isso a experiência inicial na construção e desenvolvimento de softwares se mostrou insuficiente. O custo do *software* superava as previsões, era difícil de manter, não era confiável e seu desempenho insatisfatório.

Surge a necessidade de novos métodos e técnicas para controlar os grandes sistemas de *software*.

Sommerville (2007, p.3):

> "*Software* é abstrato e intangível. Não é limitado por materiais ou controlado por leis da física ou por processos de manufatura."

Essas técnicas tornaram-se parte da engenharia de *software*, no entanto assim como aumentou a habilidade de produção de software, também aumentou a necessidade por sistemas mais complexos.

Em resumo a engenharia de *software* é uma disciplina de engenharia relacionada com todos os aspectos da produção de software, surgiu do aumento da complexidade dos *softwares* e abrange toda sua vida útil desde os estágios iniciais de especificações do sistema até sua manutenção.

2.2. TESTE DE SOFTWARE

Teste de software é o processo que executa um produto para determinar se ele funcionou corretamente no ambiente para o qual foi projetado, e se atingiu suas especificações. O objetivo é identificar falhas em um produto, para que suas causas possam ser corrigidas pela equipe de desenvolvimento antes da entrega final.

Segundo Martins (2007, p.15):

> "Um *software* precisa ser testado para descobrir erros que foram feitos durante o projeto e a construção. Os testes são conduzidos através de uma estratégia, que integra os métodos de teste, os passos e os roteiros. A estratégia de teste deve ser planejada sob medida para cada projeto, considerando o tempo que será investido neste trabalho, a disponibilidade de recursos e a tecnologia utilizada na construção do software."

O conceito de teste de *software* pode ser entendido através de duas visões, uma visão intuitiva ou mesmo de uma maneira formal. De uma forma simples, testar um *software* significa avaliar se o seu comportamento ocorre da forma como foi especificado. O objetivo principal é apontar o número máximo de falhas dispondo do mínimo de esforço, mostrar aos desenvolvedores se os resultados estão de acordo com os padrões estabelecidos. Por conta desta característica das atividades de teste, atribuem a elas uma natureza destrutiva, e não construtiva, pois visa expor seus problemas antes de sua entrega ao usuário final.

Os testes fazem parte do processo de verificação e validação, que segundo Martins (2007, p.15):

> “A verificação busca avaliar se as funções do software foram implantadas corretamente, enquanto que a validação avalia se os requisitos informados pelo cliente foram implementados consistentemente no *software*.”

Os principais níveis de teste de *software* começam pelo processo de verificação que é composto por vários níveis como o teste de unidade, que tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeito de lógica e de implementação em cada módulo, separadamente. O próximo passo é o teste de integração, que visa provocar falhas associadas ás interfaces entre os módulos quando esses são integrados para construir a estrutura do *software* que foi estabelecida na fase de projeto.

Depois vem os testes referentes ao processo de validação. O teste de validação fornece a garantia final de que o *software* satisfaz a todos os requisitos funcionais, comportamentais e de desempenho. A principal ferramenta é o teste de sistema, que avalia o *software* em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. O desenvolvedor é responsável por testar as unidades individuais para garantir que cada parte exiba o comportamento esperado. No teste de aceitação um grupo independente de pessoas simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado. Outros testes executados são os testes de segurança para verificar os mecanismos de proteção e o de estresse colocando o sistema em situações anormais de uso. Finalmente, vem o teste de desempenho que ocorre em todos os passos do processo de teste e que avalia o desempenho do software durante a sua utilização normal.

2.3. COBERTURA DE TESTE

Testes eficientes são essenciais para o controle de qualquer projeto de desenvolvimento de *software*. A cobertura de teste consiste em determinar um valor ou valores que indique qual a abrangência de um teste aplicado em um *software*. Os resultados obtidos representam um retorno de grande importância aos desenvolvedores de software.

As medidas de teste mais utilizadas são as coberturas de teste baseada em requisitos, e a baseada em código que será abordada nesse trabalho. Em suma, a cobertura de teste é qualquer medida significativa relacionada a um requisito ou critério de implementação do código. A cobertura de teste baseada em código mede a quantidade de código executada durante o teste, em comparação com a quantidade de código com execução pendente, essa modalidade de cobertura pode ser baseada em fluxos de controle ou fluxos de dados.

A cobertura sobre fluxo de controle tem por objetivo testar linhas de código, condições de ramificação, caminhos que percorrem o código ou outros elementos do fluxo de controle do *software*.

Na baseada em fluxo de dados, tem como objetivo testar se os estados dos dados permanecem válidos durante a operação do software, por exemplo, se um elemento de dados é definido antes de ser usado, ou se está recebendo um tipo de dado correspondente a ele.

A cobertura de teste baseada em códigos é calculada pela seguinte equação:

Cobertura de Teste = I / TIic

Onde I é o número de itens executados expressos como instruções, ramificações de código, caminhos de código, pontos de decisão do estado de dados ou nomes de elementos de dados. TIic é o número de itens no código.

A transformação dessa relação em uma porcentagem nos permite declarar que x% dos casos de teste (I) obtiveram uma taxa de êxito de y%.

Essa é uma explicação importante de cobertura de teste que pode ser comparada a critérios de êxito definidos. Se os critérios não forem alcançados, a explicação fornecerá uma base para prever a quantidade de esforço de teste restante.

Hoje em dia, existem diversos *frameworks* que nos permitem efetuar a cobertura de teste e dentre estes *frameworks* está o Selenium IDE o qual será abordado nesse estudo.

3. FRAMEWORK

3.1. SELENIUM

O Selenium é um framework de teste de software para aplicações Web. O Selenium roda no Windows, Linux e Macintosh. Os testes podem ser escritos codificados em diversas linguagens de programação populares e podem ser executados diretamente na maioria dos browsers modernos da Web ou como tabelas HTML.

Selenium surgiu em 2004, quando Jason Huggins estava testando um aplicativo interno na ThoughtWorks. Ele percebeu que havia melhor uso para a ferramenta, sendo assim desenvolveu uma biblioteca JavaScript que poderia conduzir interações com a página, permitindo que ele volte a executar automaticamente testes contra vários navegadores. Essa biblioteca se tornou Selenium Core, que traz toda funcionalidade de controle remoto Selenium e Selenium IDE. Selenium foi inovador porque nenhum outro produto permitia controlar um navegador a partir de um idioma de sua escolha.

Enquanto Selenium foi uma tremenda ferramenta, por outro lado não foi isento de suas desvantagens. Devido ao seu mecanismo de automação baseado em Javascript e as limitações de segurança que navegadores aplicam ao Javascript, coisas diferentes tornaram-se impossível de fazer. Para piorar as coisas, webapps ficaram mais e mais poderosos ao longo do tempo, usando todos os tipos de características especiais e novos navegadores surgindo.

Em 2006, um engenheiro da Google chamado Simon Stewart começou a trabalhar em um projeto que chamou WebDriver. Google tinha sido por muito tempo um usuário pesado de selenium, mas testadores tiveram de contornar as limitações do produto. Simon queria uma ferramenta de teste que fosse diretamente para o navegador usando o método "nativo" para o navegador e sistema operacional, evitando assim as restrições de um ambiente Javascript modo seguro. O projeto WebDriver começou com o objetivo de resolver as desvantagens do Selenium.

No ano de 2008 foi realizada a fusão de Selenium e WebDriver. Selenium teve enorme comunidade e suporte comercial, mas WebDriver era claramente a ferramenta do futuro.

A junção das duas ferramentas fornecia um conjunto comum de recursos para todos os usuários.

3.2. INSTALAÇÃO SELENIUM IDE

Utilizando o navegador FireFox, faça o download do selenium IDE conforme imagem:

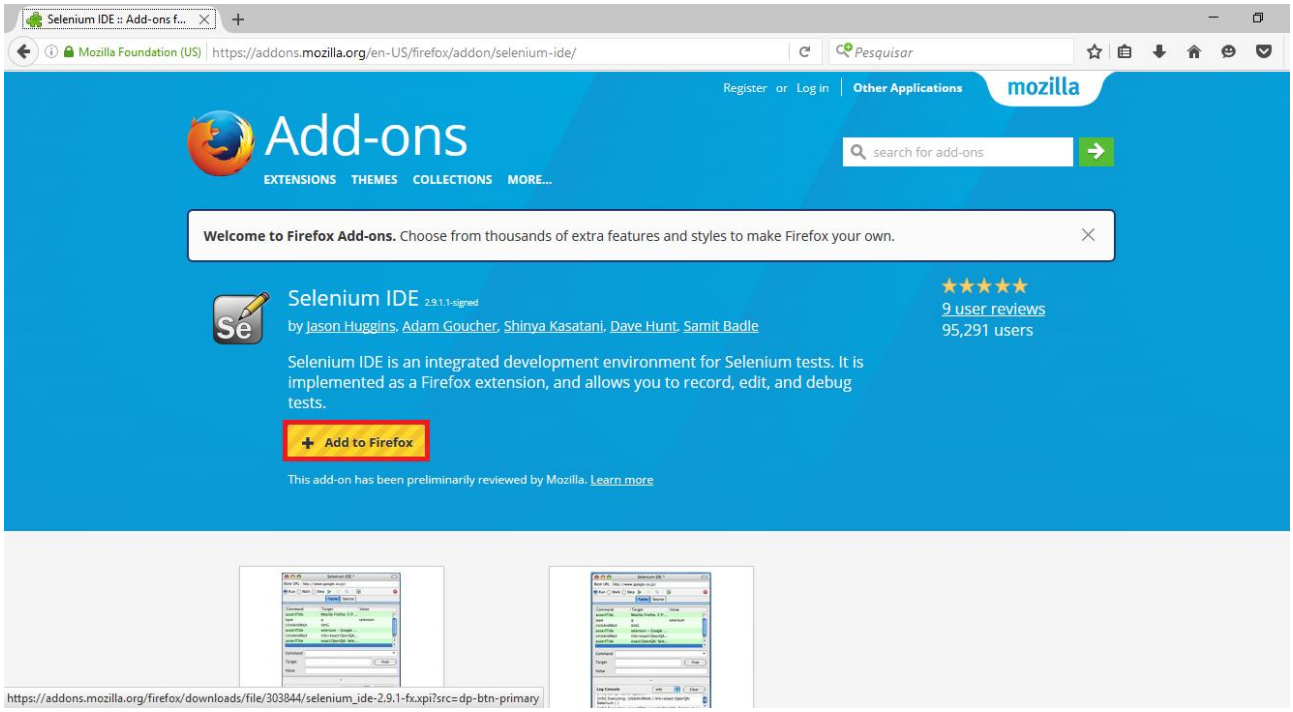

Figura 1 – Download Selenuim IDE

Após o download e instalação reinicie o navegador, você encontrará o selenium IDE no menu ferramentas.

Figura 2 – Menu Ferramentas

Para executar o Selenium, abra o menu ferramentas e selecione Selenium IDE, será exibida uma janela de edição vazia e um menu para carregamento, ou criação de casos de teste.

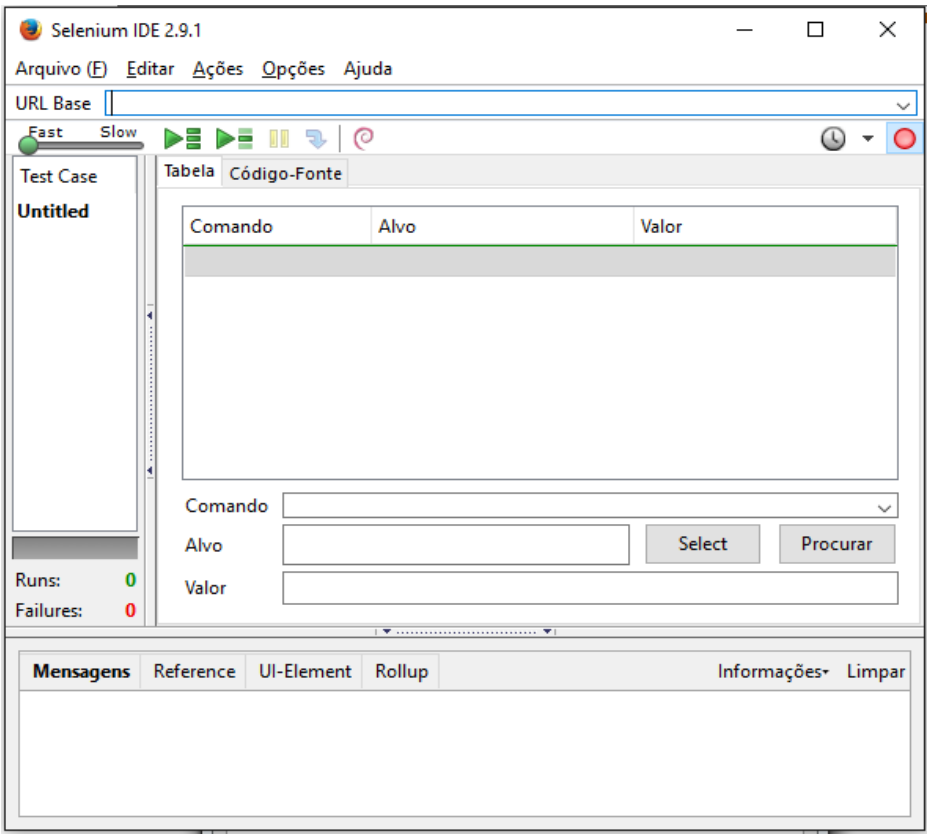

Figura 3 – Janela Selenium IDE

A barra de ferramentas possui botões de controle como:

Controle de velocidade;

Executar tudo;

Executar;

Pause/Resume

Passo;

Modo TestRunner;

Aplicar regras de rollup;

Record.

3.3. COMANDOS SELENIUM

Comandos de selenium são conjuntos de comandos que executam o teste. A sequencia destes comandos formam um script de teste. Selenium fornece um amplo conjunto de comandos para testes de aplicações web.

Um comando diz ao Selenium o que fazer. Comandos de selenium vêm em três tipos: Actions , Accessors , e Assertions .

- **Actions** são comandos que manipulam o estado do aplicativo. Estes comandos fazem coisas como “clique neste link” e “Selecionar esta opção”. Se uma ação falha a execução do teste para.
- **Accessors** examina o estado da aplicação e armazenar os resultados em variáveis, por exemplo, "storeTitle". Eles também são utilizados para gerar automaticamente afirmações.
- **Assertions** são como Accessors, mas verificam se o estado da aplicação está de acordo com o que se espera. Exemplos como "certifique-se o título da página é X" e "verifique se esta opção for assinalada".

  Todas as afirmações de selenium podem ser utilizados em 3 modos: “assert”, “verify”, e “waitFor” Por exemplo, você pode utilizar “assertText”, “verifyText” e “waitForText”. Quando um “assert” falhar, o teste para. Quando um “verify” falha, o

teste continua a execução, registrando a falha. Isso permite que um “assert” garanta que a aplicação está na página correta, seguido por um conjunto de “verify” afirmações para testar valores de campo de formulário, etiquetas, etc.

Comandos de selenium são simples, eles consistem de comando e dois parâmetros. Por exemplo:

verifyText	//div//a[2]	Login

Tabela 1 - Exemplo Script Syntax

Os parâmetros nem sempre são informados, eles dependem do comando. Em alguns casos, ambos são necessários, em outros é necessário um parâmetro, e em outros ainda o comando pode ficar sem parâmetros em tudo. Outros exemplos:

goBackAndWait		
verifyTextPresent		Bem-Vindo à minha página inicial
Digitar	id = telefone	(555) 666-7066
Digitar	id = address1	$ {} MyVariableAddress

Tabela 2 - Exemplo Script Syntax 2

A referência de comando descreve os requisitos de parâmetro para cada comando. Parâmetros variam em:

- um localizador para identificar um elemento de interface do usuário dentro de uma página.
- um padrão de texto para a verificação ou afirmação do conteúdo da página esperado.
- um padrão de texto ou uma variável de selenium para a inserção de texto em um campo de entrada ou para selecionar uma opção a partir de uma lista de opções.

Localizadores, padrões de texto variáveis de selenium, e os comandos em si são descritos em detalhes considerados na seção de Comandos selenium.

Comandos típicos de Selenium. Estes são provavelmente os comandos mais usados para testes de construção.

open

abre uma página usando uma URL.

click/clickAndWait

executa uma operação de clique, e opcionalmente aguarda uma nova página para carregar.

verifyTitle / assertTitle

verifica um título de página esperada.

verifyTextPresent

verifica se o texto esperado está em algum lugar na página.

verifyElementPresent

verifica um elemento IU esperado, tal como definido pelo seu tag de HTML, está presente na página.

verifyText

verifica o texto esperado e se sua tag HTML correspondente estão presentes na página.

verifyTable

verifica o conteúdo esperado de uma tabela.

waitForPageToLoad

para a execução até que uma novas cargas de página. Chamado automaticamente quando clickAndWait é usado.

waitForElementPresent

para a execução até que um elemento UI esperado, conforme definido pelo seu tag HTML, esteja presente na página.

3.4. ARMAZENAR COMANDOS E VARIÁVEIS DE SELENIUM

Você pode usar variáveis de selenium para armazenar constantes no início de um script. Além disso, quando combinado com um design de teste orientado a dados, as variáveis de selenium podem ser usadas para armazenar valores passados para o seu programa de teste a partir da linha de comando, a partir de outro programa, ou a partir de um arquivo.

A simples store de comandos é o mais básico dos muitos comandos de armazenamento e pode ser usada apenas para armazenar um valor constante em uma variável de selenium. Leva dois parâmetros o valor de texto a ser armazenado e uma variável selenium. Use as convenções de nomenclatura padrão variável de apenas caracteres alfanuméricos na escolha de um nome para a variável.

Comando	**Alvo**	**Valor**
Store	lucas@meusite.com	userName

Tabela 3 – Exemplo armazenar variáveis

Para acessar o valor de uma variável, coloque a variável entre chaves {} e antes coloque o sinal de $.

Comando	**Alvo**	**Valor**
verifyText	// Div / p	$ {Username}

Tabela 4 - acessar o valor de uma variável

Um uso comum das variáveis é para o armazenamento de entrada para um campo de entrada.

Comando	**Alvo**	**Valor**
Digitar	id = login	$ {Username}

Tabela 5 - acessar o valor de uma variável 2

Variáveis de selenium podem ser usadas tanto no primeiro ou segundo parâmetro e são interpretadas pelo selenium antes de quaisquer outras operações realizadas pelo comando. Uma variável de selenium pode também ser usada dentro de uma expressão localizadora.

Um comando equivalente store existe para cada comando verify e assert. Aqui estão alguns mais usados da store.

storeElementPresent

Corresponde a verifyElementPresent. Ele simplesmente armazena um valor booleano "verdadeiro" ou "falso" dependendo se o elemento UI é encontrado.

storeText

StoreText corresponde a verifyText. Ele usa um localizador para identificar textos expecificos dentro de uma pagina. O texto se for encontrado é armazenado na variável. StoreText pode ser usado para extrair o texto a partir da página que está sendo testada.

storeEval

Este comando pega um script como seu primeiro parâmetro. StoreEval permite o teste para armazenar o resultado da execução do script em uma variável.

3.5. SOLUÇÃO DE PROBLEMAS

Segue uma lista com alguns erros e explicações que descrevem fontes frequentes de problemas com Selenium-IDE:

“Table view is not available with this format.”

Esta mensagem pode ser exibida na guia Tabela quando Selenium IDE é lançado. A solução é fechar e reabrir Selenium IDE.

“error loading test case: no command found.”

Você usou **Arquivo => Abrir** para tentar abrir um arquivo de conjunto de testes. Use **Arquivo => Open Test Suite** em vez.

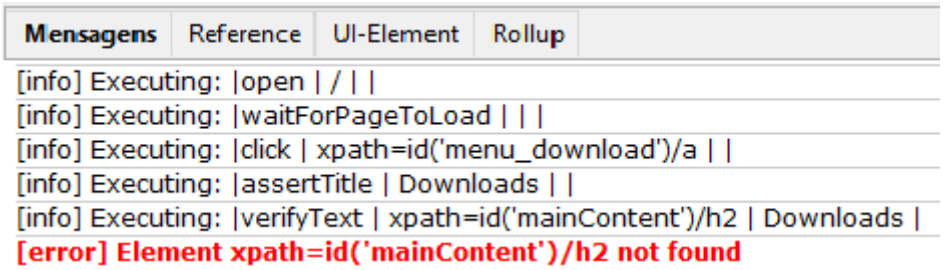

Figura 4 – erro

Este tipo de erro pode indicar um problema de tempo, ou seja, o elemento especificado por um localizador em seu comando não foi totalmente carregado quando o comando foi executado.

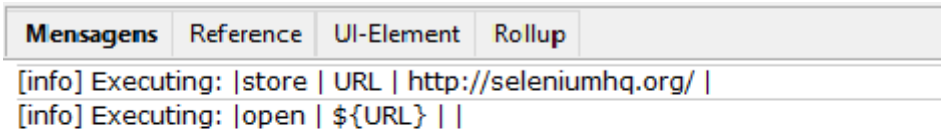

Figura 5 – erro 2

Este é devido variável no campo valor estar no campo alvo ou vice-versa. No exemplo acima, os dois parâmetros estão colocados na ordem inversa.

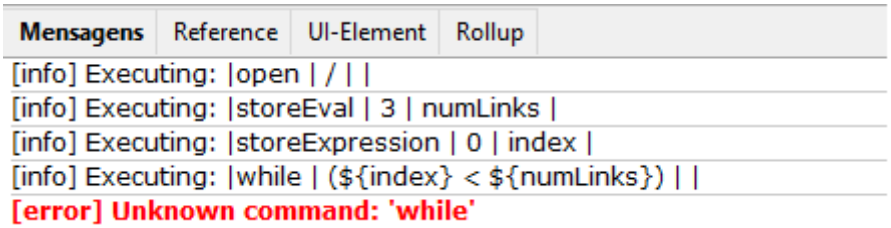

Figura 6 – erro 3

O conteúdo de seu arquivo de extensão não foi encontrado pelo Selenium-IDE. Certifique-se de que você especificou o caminho correto para o arquivo extensões através de Opções => Opções => Geral no extensões Selenium Core de campo.

4. ESTUDO DE CASO

Este capítulo tem por objetivo apresentar como se utiliza o framework de forma prática. O principal objetivo desse framework é simplificar e otimizar a criação de testes funcionais, para a demonstração será utilizado um site www.somatamatica.com.br\matkids.php como pode ser visto na Figura - 9.

Para realização dos testes foram criados dois casos de uso conforme figura 7 e 8.

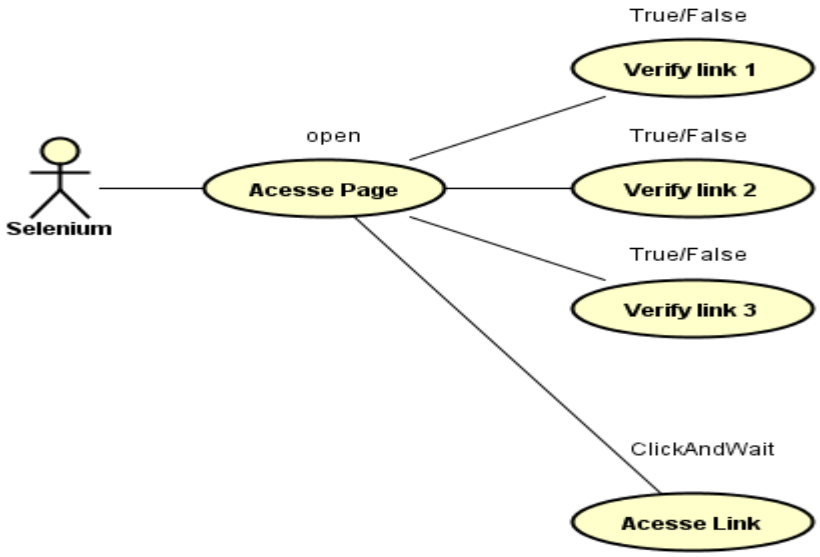

Figura 7 – Caso de Uso Teste1

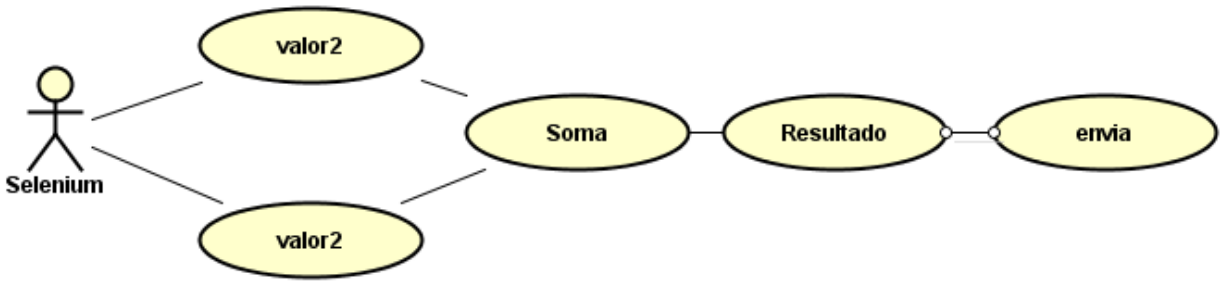

Figura 8 – Caso de Uso Teste2

Após definido a aplicação web a ser testada é preciso identificar os PageObject que são basicamente a separação de funcionalidades elementos do HTML da página onde o teste irá interagir.

Figura 9 – Somatematica

4.1. IDENTIFICANDO E ACESSANDO LINKS

Inicie o Selenium IDE, ele se encontra na barra de menu Ferramentas. Vamos começar pelo campo URL base que vamos testar, no caso, www.somatamatica.com.br\matkids.php. Desabilite a gravação automática dos testes, e apertando o botão direito no campo do meio, teremos uma opção para inserir um novo comando. A imagem a seguir mostra a tela do Selenium com o botão de gravação e a opção de novo comando.

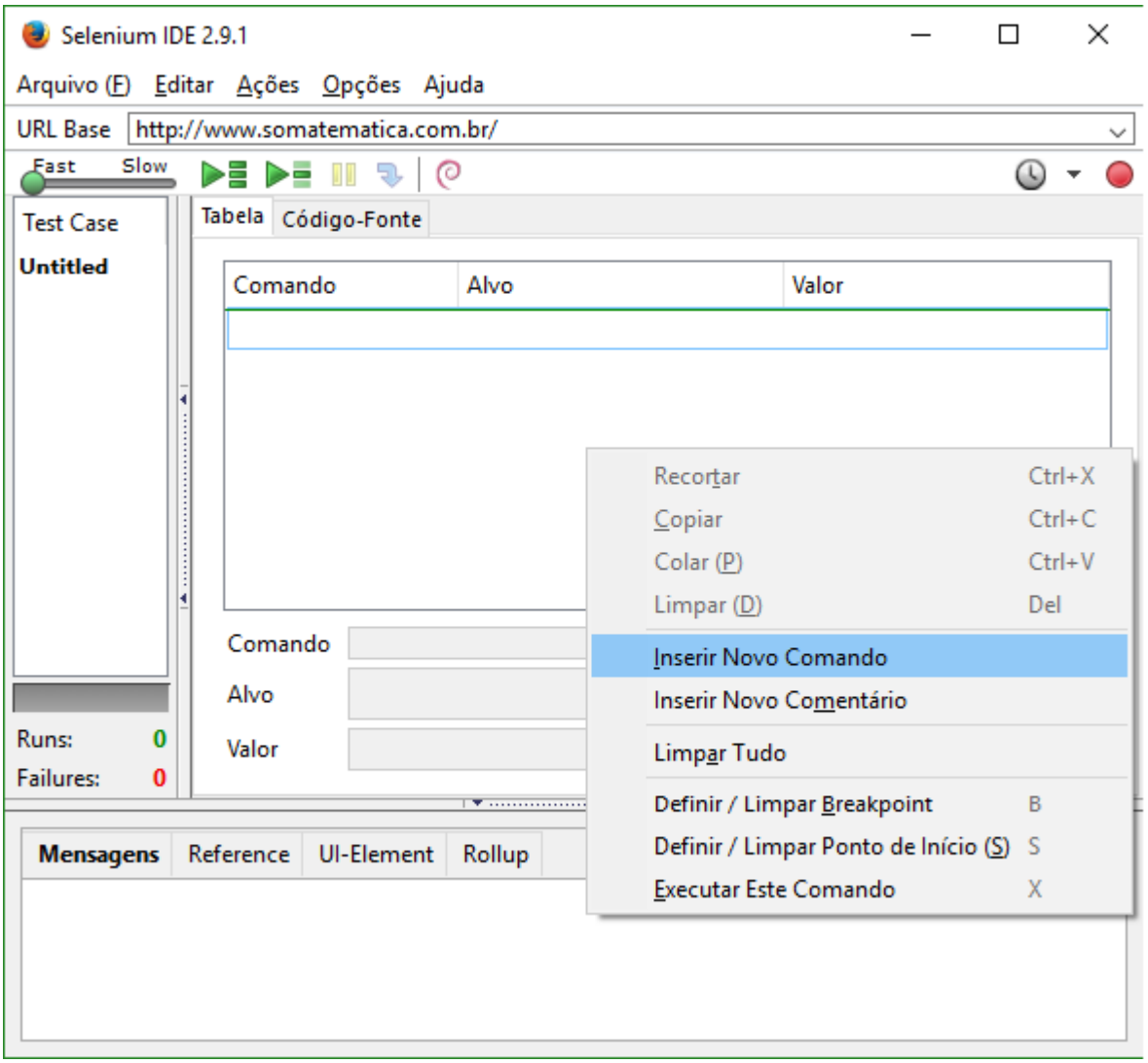

Figura 10 – Novo comando

Os campos “Comando”, “Alvo” e “Valor” serão habilitados, insira o comando “open” seguido pelo seu alvo que corresponde a url da página que será testada.

Insira novos comandos para verificar se alguns links do menu lateral estão presentes, utilizando o comando “VerifyElementPresent”, que retorna true se o parâmetro “alvo” está presente e false, caso contrário, utilize a função Inspecionar elementos do navegador para melhor identificação dos PageObject.

Com o comando “clickAndWait” faça o Selenium acessar um dos links. Os comandos devem ficar como mostra a figura 11.

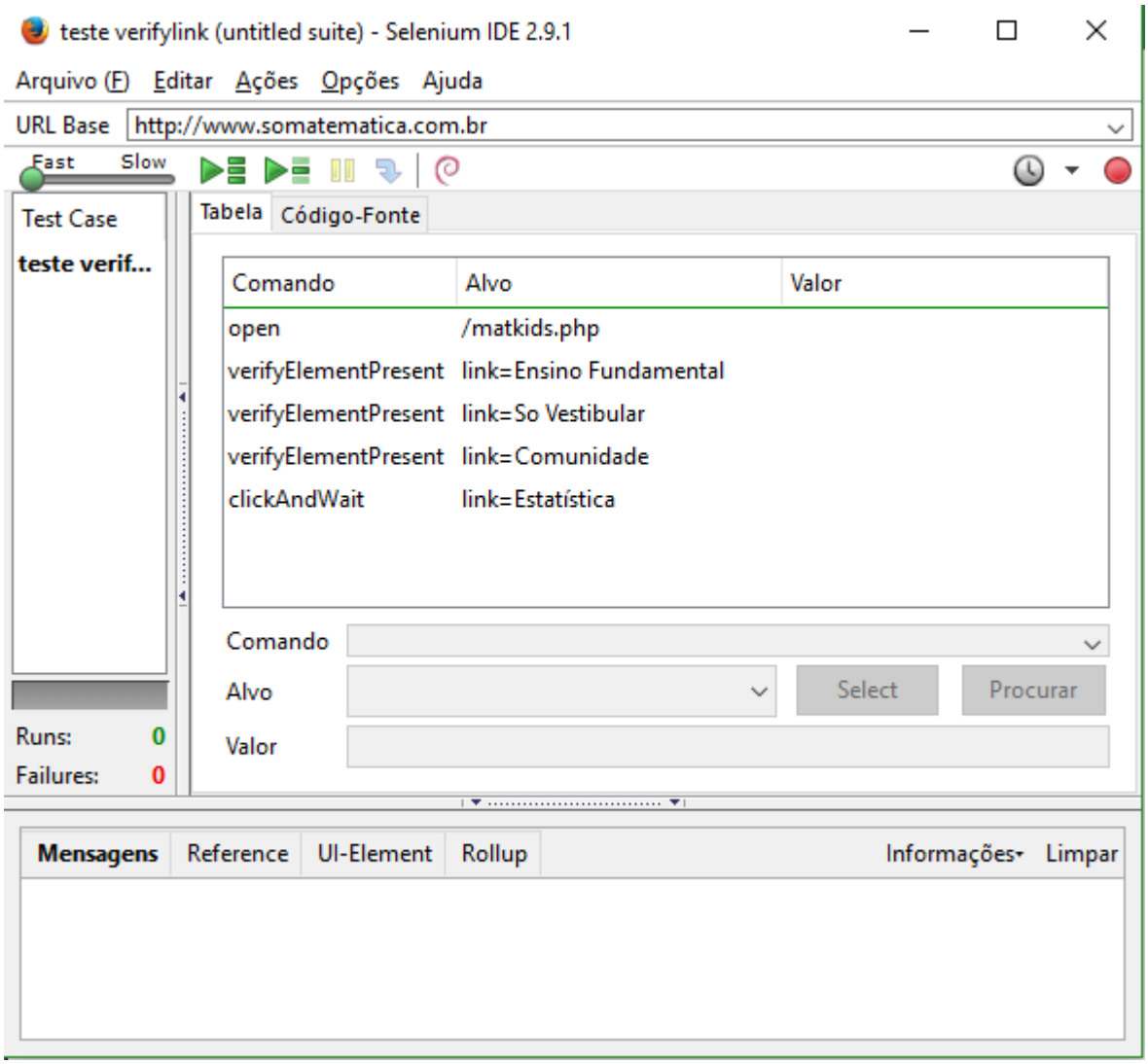

Figura 11 – Teste 1

Execute o teste e verifique o retorno do Selenium na aba “Mensagens” na parte inferior da Janela.

Com a execução do teste nota se que uma das verificações de link apresentou erro conforme a figura 10. O erro ocorreu devido ao informar no campo alvo um link inexistente ou escrito de forma incorreta.

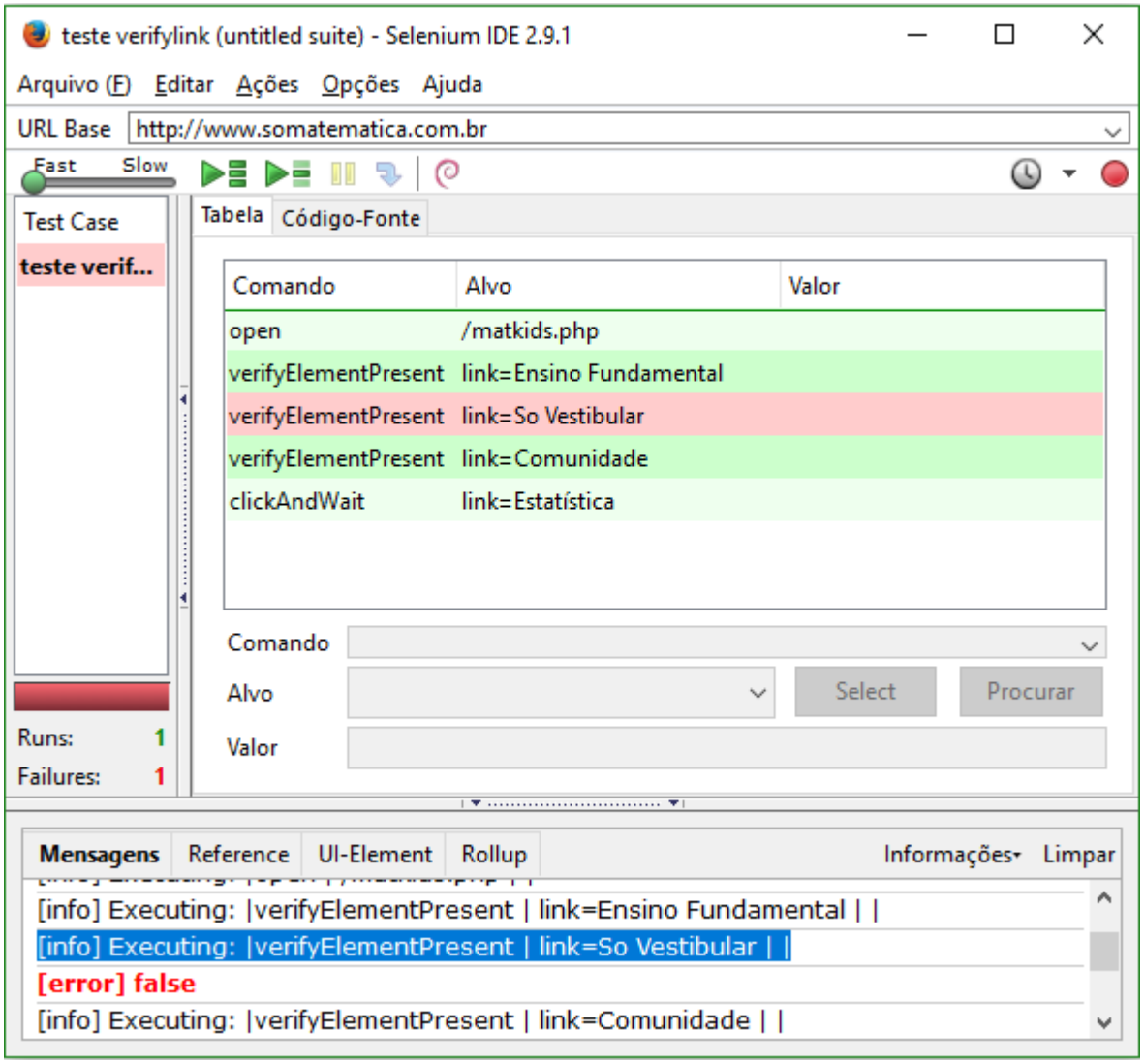

Figura 12 – Retorno Teste 1

4.2. ARMAZENANDO VALORES

Acesse a URL http://www.somatematica.com.br/matkids/game.php, insira um Nome e escolha uma operação, no caso será utilizada Adição, click em Quero praticar para exibir a tela onde será realizado o teste conforme figura 11. Inicie o Selenium IDE.

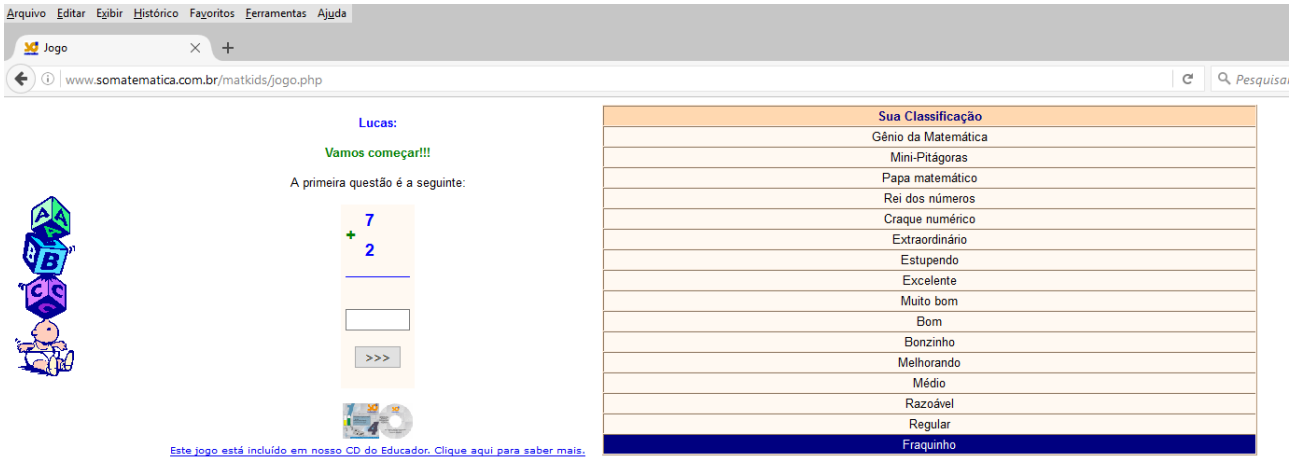

Figura 13 – Jogo de somar

Click com o botão direito do mouse sobre o primeiro valor, vá em Exibir todos os comandos e escolha a opção “storeText”, este comando irá localizar e armazenar o valor selecionado um uma variável, nomeie a variável como valor1, realize o mesmo procedimento com o segundo valor mas o nomeie como valor2. No Selenium os comandos store estarão criados, insira um novo comando “storeEval” no campo alvo informe: ${valor1}+${valor2}, e o nomeie como resultado, assim somamos os valores das variáveis e armazenamos em outra variável o resultado. Identifique os PageObjects do campo onde deve ser informado o resultado e do botão para enviar o resultado.

Insira um novo comando com a PageObject para informar o resultado, no campo valor informe: ${resultado}, para chamar a variável que está armazenando a soma dos dois valores. Crie um novo comando para realizar o evento de envio do resultado.

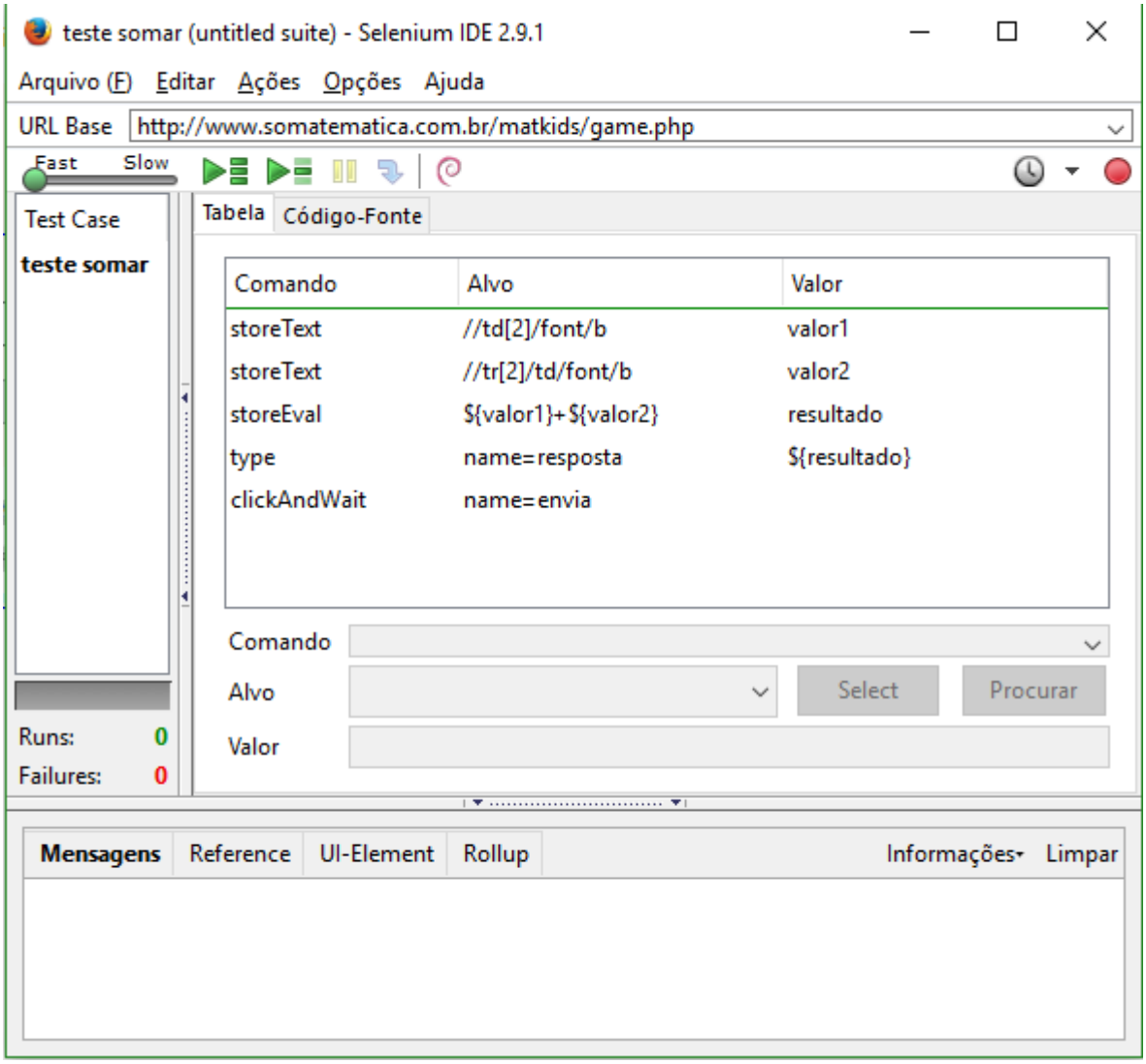

Figura 14 – Teste 2

Execute o teste os valores deveram ser identificados, somados, o resultado informado no campo e enviado para validação da página como é apresentado na figura 15.

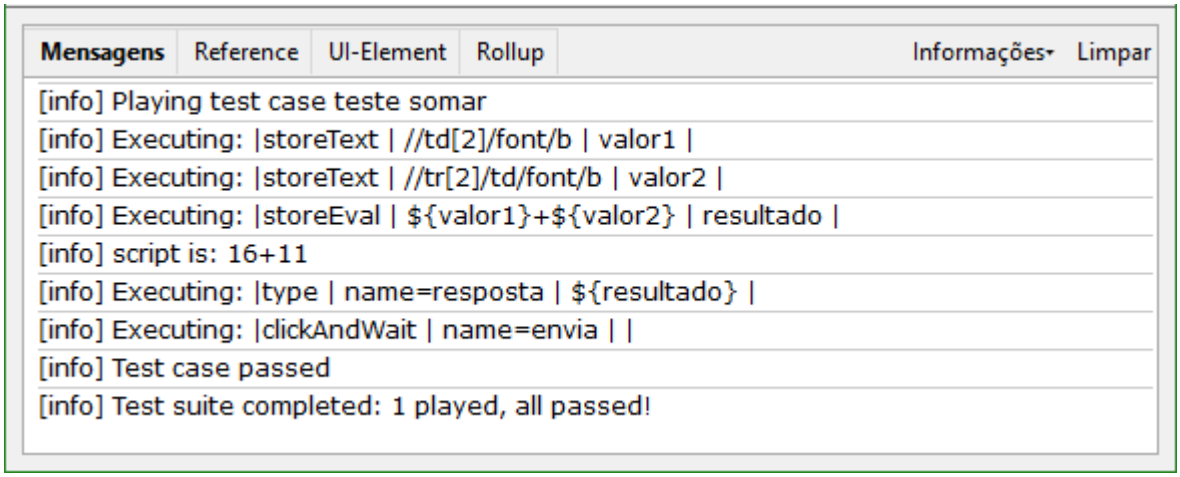

Figura 15 – Retorno Teste 2

5. CONCLUSÃO

Selenium IDE é uma ferramenta de prototipagem para a construção de scripts de teste. É um plugin do Firefox e fornece uma interface fácil de usar para o desenvolvimento de testes automatizados. Selenium IDE tem um recurso de gravação, que registra as ações do usuário como elas são realizadas e depois exporta como um script reutilizável em uma das muitas linguagens de programação que pode ser executado.

Este trabalho mostrou ter uma visão geral do que é o Selenium IDE, como realizar a instalação, a utilização e as principais ações que necessitamos para utiliza-lo de uma forma mais correta.

O intuito desta pesquisa foi reunir uma quantidade de informações relevantes sobre o Selenium, e assim agregar conhecimento para os interessados em cobertura de teste.

Para uma futura atualização do framework, destaca-se a possibilidade de se utilizar tipos de massa de dados como banco de dados e arquivos texto, também a possibilidade de disponibiliza-lo como código open-source e a criação de funções e métodos que dê suporte a criação de casos de teste.

6. REFERÊNCIAS

Conceitos: Principais Medidas de teste Disponível em:< http://www.wthreex.com/rup/portugues/process/workflow/test/co_keyme.htm>. Acesso em: 13 Fev. 2016.

GIL, Carlos Antônio. **Como Elaborar Projetos de Pesquisa.** São Paulo: Atlas, 2010. 200p.

GIL, Carlos Antônio. **Como Elaborar Projetos de Pesquisa.** São Paulo: Atlas, 2002. 176p.

MARIANO, Pedro. **Cobertura de teste e a falsa impressão de segurança.** Disponível em:< http://www.infoq.com/br/news/2010/06/cobertura-testes-falsa-impressao>. Acesso em: 13 Fev. 2016.

MARTINS, José Carlos Cordeiro. **Técnicas para gerenciamento de projetos de software.** Rio de Janeiro: Brasport, 2007. 456 p.

NetBeans. **Testando com o PHPUnit e Seleium**. Disponível em:< https://netbeans.org/kb/docs/php/phpunit_pt_BR.html>. Acesso em: 05 Mar.2016.

REIS, Christian. **Cobertura e Adequação do Teste de Unidade.** Disponível em:< http://www.async.com.br/~kiko/papers/testcriteria/>. Acesso em: 13 Fev. 2016.

Revista Engenharia de Software. **Introdução a Teste de Software**. Disponível em:< http://www.devmedia.com.br/revista-engenharia-de-software/8028>. Acesso em: 13 Fev. 2016.

SOMMERVILLE, Ian. **Engenharia de software.** São Paulo: Pearson, 2007. 568p.

Testingbot. **Primeiro Teste PHP**. Disponível em:< http://testingbot.com/support/getting-started/php.html>. Acesso em: 14 Fev.2016.

SeleniumHQ. **Introduction**. Disponível em:< http://docs.seleniumhq.org/docs/01_introducing_selenium.jsp#introducing-selenium>. Acesso em: 26 jul.2016.

SeleniumGuideBook. **Introduction**. Disponível em:< https://seleniumguidebook.com/ >. Acesso em: 28 jul.2016.