RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

AUTOMAÇÃO DE TESTES UTILIZANDO A FERRAMENTA CUCUMBER

TEST AUTOMATION USING CUCUMBER TOOL

Felipe Malheiros Curti[1], Felipe Diniz Dallilo[2]

e321133
https://doi.org/10.47820/recima21.v3i2.1133

RESUMO
Este artigo apresenta um estudo por meio da criação de cenários de testes de automação realizados com a ferramenta *Cucumber*. Adotou-se a metodologia descritiva de cunho quali-quantitativa e teve como ponto de partida um referencial bibliográfico. Na qual foi realizada com auxílio de livros, artigos, teses de mestrado e doutorado e com auxílio da internet. O levantamento dos dados teve base de investigação em bases de consulta como SCIELO (*A Scientific Electronic Library Online*), Revistas online: CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) USP (Universidade de São Paulo) FGV, (Fundação Getúlio Vargas). Em seguida foi realizada descrição detalhada do projeto de automatização, com as ferramentas e tecnologias utilizadas como o *Cucumbe*r Utils.java' e JUnit de modo a compreender a funcionalidade dos testes. Os estágios do *Cucumber* foram executados em uma linguagem comum, clara e objetiva. Onde foi possível concluir dizendo que, o teste de automação com o *Cucumber* oferece uma camada de comunicação real sobre um *framework* de teste e que no desenvolvimento do teste verificou-se que o processo de automação pode reduzir o tempo que um analista de testes empreende em uma tarefa quando comparado com o processo manual.

PALAVRAS- CHAVE: Automação.*Cucumber*.Teste

ABSTRACT
This article presents a study through the creation of automation test scenarios performed with the Cucumber tool. A qualitative-quantitative descriptive methodology was adopted, and the starting point was a bibliographic reference. In which it was carried out with the help of books, articles, master's and doctoral theses and with the help of the internet. Data collection was based on research in databases such as SCIELO (A Scientific Electronic Library Online), Online Journals: CAPES (Coordination for the Improvement of Higher Education Personnel) USP (University of São Paulo) FGV, (Fundación Getúlio Vargas). Then, the automation project was detailed in detail, with the tools and technologies used such as Cucumber Utils.java' and JUnit in order to understand the functionality of the tests. Cucumber's projects were carried out in a common, clear and objective language. Where it was possible for communication to take place, test automation offers a layer of realness over the test and that test development can be evaluated, the process of reducing the time a test analyst undertakes on a task compared to the process manual.

KEYWORDS*: Automation. Cucumber. Test*

INTRODUÇÃO

É fato que o mundo dos negócios está em constantes mudanças, e o *software* deve acompanhar esse movimento. Dessa forma, é fundamental que o produto se adapte a novas regras de negócios, e acumule novas funcionalidades. Sendo que, a realização de manutenção de um

[1] Universidade de Araraquara - UNIARA
[2] Mestrado em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo, Brasil (2014) - Professor da Universidade de Araraquara, Brasil

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

software é o aperfeiçoamento de todo um sistema e ao realizar a manutenção o sistema ficara mais estável.

De modo geral os *softwares* passam pelo processo de teste manual, esse processo resulta no atraso na entrega de uma atividade, por levar mais tempo para a realização de testes mais simples. A busca de medidas para redução de tempo é constante, por isso cada vez mais são utilizados métodos e metodologias que ajudam a minimizar o tempo para realização todas as tarefas no prazo estipulado.

O objetivo geral desse trabalho é desenvolver um programa de automação de teste a partir da ferramenta *Cucumber*, dessa forma evidenciando uso dessa prática para o benefício dos clientes em potencial.

A automação de testes é uma prática que está diretamente ligada garantia de qualidade (GQ) do produto final. A etapa de teste é uma etapa essencial para o sucesso de todo s*oftware*, no seu desenvolvimento e no decorrer da sua vida útil. É relevante que com a automação de testes de *software* tenha redução tempo e de custos, sem perder a qualidade.

Segundo Caetano (2008) a automação de testes é uma área que está em constante expansão, porém ela ainda é muito imatura. Muitas vezes ela acaba sendo pouco utilizada, pois causa um certo receio confiar nas máquinas para realizar um trabalho que exige uma confiança e um certo grau de confiabilidade.

Nesse contexto justifica-se esta pesquisa contribuir para o uso da automação de testes visando demonstrar as vantagens que podem ser obtidas por empresas na questão do lucro, redução de custos e tempo em manutenção, aumento na confiabilidade e satisfação do cliente. A importância e a motivação desse trabalho residem na premissa de que, em projetos de sistemas de *software* num ambiente profissional é necessária a utilização de testes automatizados de forma a manter a qualidade do produto final.

Garantir a qualidade de um *software* desenvolvido pelas empresas, é uma necessidade, assim como segurança e aderência entre projeto e execução e a satisfação dos clientes.

A pergunta que norteia este estudo é: Como os testes automatizados podem ajudar a garantir a qualidade de *software*?

A metodologia adotada nesta pesquisa é uma abordagem descritiva de cunho quali-quantitativa e teve como ponto de partida um referencial bibliográfico sobre o objeto de estudo que norteia acerca do teste de automação utilizando o método Ágil. Na qual foi realizada com auxílio de livros, artigos, teses de mestrado e doutorado e com auxílio da internet. O levantamento dos dados teve base de investigação em bases de consulta como SCIELO (*A Scientific Electronic Library Online*), Revistas *online*: CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) USP (Universidade de São Paulo) FGV, (Fundação Getúlio Vargas).

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

REVISÃO BIBLIOGRÁFICA

Para descrever sobre o tema de automação de teste se faz necessário apresentar alguns conceitos que, segundo Molinari (2008 p. 74) o processo de teste "é um conjunto de passos parcialmente ordenados, cujo objetivo é atingir uma meta para entregar um produto de software de maneira eficiente previsível e que vá ao encontro das necessidades do negócio".

AUTOMAÇÃO DE TESTES E CONCEITOS CORRELATOS

Teste automatizado é o uso da inteligência de *software* que controla a execução e a aderência dos resultados obtidos comparando-os com os resultados esperados, tentando alcançar através de ferramentas e técnicas adequadas um processo de qualidade aceitável. Teste é um processo para executar um *software* ou sistema com o objetivo de revelar a presença de defeitos e tem como meta aumentar a confiança sobre o software testado (VELOSO, 2014).

Segundo Molinari (2008) existem duas razões iniciais para a automatizar testes de *software*:

a) argumento tem relação ao crescimento elevado de sistemas e aplicações, onde se torna difícil o controle e elaboração de testes manuais;
b) baseia-se na alta complexidade dos *softwares* que são desenvolvidos atualmente

Considerando os pontos levantados, existem sistemas em que a única solução para a execução dos testes é a utilização da automatização, considerando qual o tipo de teste que será realizado e da função a ser realizada, tornando assim algumas vezes a execução do teste manual inviável (MOLINARI, 2008).

Segundo Wootton (2013) existe certa dificuldade das empresas desenvolvedoras na entrega de qualidade de seus produtos e satisfação dos seus clientes. Entregar *software* em produção é um processo que tem que se tornado cada vez mais difícil nas empresas de T.I.

Para Fowler (2012) a principal vantagem da automatização de testes é a economia de tempo e recursos durante a execução dos testes. Outro fator importante é que o teste automatizado, como qualquer outro *software*, comunica-se com o *software* que está sendo testado através de uma interface

TESTE SOFTWARE

Conforme Fernandes e Fonseca (2020) no contexto do desenvolvimento de um sistema, um *software* ou uma aplicação, existem diversas ferramentas que podem ser utilizadas para realização da automação de testes. Entre elas estão o *Selenium, Cucumber, Apium, TestComplete, Ranorex, Robotium,* entre outras. Cada uma possui elementos e funcionalidades que as tornam específicas para determinados tipos de teste e ambientes.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Segundo Rios e Moreira (2013) o processo de teste deve estar presente durante todo o desenvolvimento do *software,* porém esses testes podem ser divididos em diferentes fases, as quais se diferenciam pela abstração e complexidade dos testes produzidos e executados em cada uma delas. o processo de teste é dividido em quatro (4) fases, sendo Testes de Unidade, Testes de Integração, Testes de Sistema e Testes de Aceitação, a saber:

a) **Testes de Unidade:** Na fase dos Testes de Unidade os esforços estão concentrados nas menores unidades do software produzido. O objetivo é detectar erros de lógica e/ou de implementação em pequenas partes do sistema, independentemente do restante. Dessa forma, cada unidade do sistema é testada isoladamente. Isso contribui para assegurar a correção dos componentes individuais, mas não garante que a integração dessas partes funcione como o esperado.

b) Testes de Integração: A fase do Testes de Integração o foco está voltado para a detecção de falhas provenientes da integração interna dos componentes do sistema. Para isso os módulos são combinados e testados em conjunto. Esta fase vem logo após os testes de unidade e antecede os testes de sistema, tendo como resultado o sistema integrado e preparado para o teste de sistema. Não faz parte do escopo dessa fase testar a interação do sistema produzido com outros sistemas, que porventura venham a se comunicar.

c) Testes de Sistema: Teste de sistemas é a fase na qual o *software* já está completamente integrado. Sendo assim, os testes visam identificar falhas em relação aos requisitos do sistema, no que diz respeito à comunicação com outros sistemas. Os testes são realizados em condições bastantes parecidas (de ambiente, massa de dados etc.) com as que o usuário utilizará em produção. Os testes de sistema não se limitam aos requisitos funcionais, mas também objetiva testar os requisitos não-funcionais.

d) **Testes de Aceitação:** Os testes de aceitação são, em geral, uma extensão dos testes de sistema. Durante esta fase, o objetivo é verificar se o *software* está pronto e pode ser usado pelo usuário final. Para isso é verificado se o sistema realiza as funções para as quais ele foi criado, satisfazendo as necessidades do cliente. Os testes são planejados e projetados com o mesmo cuidado e nível de detalhe do teste do sistema. Nesta fase, os critérios de aceitação são conhecidos, o que permite a automação dos testes de aceitação, além da monitoração e medição.

De acordo com a ISTQB (INTERNATIONAL SOFTWARE TESTING QUALIFICATION BOARD, 2018) as ferramentas de automação de testes funcionam com a execução de um conjunto de instruções redigidas em uma linguagem de programação, que na maioria das vezes é chamada de linguagem *script*. As instruções para as ferramentas são muito detalhadas e entradas específicas, ordem das entradas, valores específicos utilizados nas entradas e saídas esperadas. Isto pode deixar *scripts* detalhados suscetíveis a alterações no *software* em teste SUT (*System Under Test)*

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

particularmente quando a ferramenta interage com a interface gráfica de usuário GUI (*Graphical user interface)*.

Para Fowler (2012) os testes automatizados mais próximos da interface devem funcionar como uma segunda linha de defesa e investimento, assim a camada intermediária ou camada de serviços e camada de baixo nível (unitária) pode fornecer muitas vantagens e rápidas respostas de execução ao validar serviços (*WebServices* ou APIs) e unidades de *software* (funções e métodos).

De acordo Meszaros e Wesley (2007) uma forma de tornar os testes de *software* mais independentes comparados com a intervenção humana, é o uso de testes automatizados como uma boa prática que consiste na criação de *scripts* que verificam um sistema em teste, desta forma é possível capturar comportamentos e retornos de maneira automática e dinâmica.

PROCESSO DE TESTE MÉTODOS ÁGEIS

Conforme Sommerville (2011), o processo de teste em métodos Ágeis leva muito em consideração o teste de aceitação como um parâmetro de qualidade. O cliente participando do processo ajuda também na elaboração dos critérios para aceitação e o nível de qualidade desejado.

Mendonça e Silva (2014) salientam que os métodos Ágeis compartilham valores como comunicação, *feedback* constante, colaboração com o cliente e constante adaptação são baseados no manifesto Ágil. Os quatro princípios básicos do manifesto ágil mostram o que se espera de qualquer método de desenvolvimento desta categoria:

1) Indivíduos e interações sobre processos e ferramentas.
2) *Software* funcionando sobre documentação extensiva.
3) Colaboração com o cliente sobre negociação de contrato.
4) Responder às mudanças sobre seguir um planejamento.

De acordo com Mendonça e Silva (2014, p. 24) “o papel do cliente é crucial para participar das validações quando se trata projetos Ágeis, também determina em conjunto com a equipe de desenvolvimento o que será prioridade para se desenvolver”.

A utilização de métodos Ágeis no desenvolvimento de *software* tem como características intrínsecas a flexibilidade e rapidez nas respostas a mudanças. A agilidade, para uma organização de desenvolvimento de *software*, é a habilidade de adotar e reagir rapidamente e apropriadamente a mudanças no seu ambiente e por exigências impostas pelos clientes (NEGRELLO, 2013).

Segundo Bernardo (2011) a prática intrínseca ao desenvolvimento surgiu na década de 1970, onde surgiu a necessidade de criar programas para testar cenários específicos, a fim de obter melhorias na qualidade do *software*.

Para Negrello (2013) a qualidade do próprio processo de desenvolvimento adotado contribui para a qualidade do produto, resistência da utilização das metodologias Ágeis existe por diversos

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

fatores, primeiramente porque ainda existe empresas que esperam os produtos ficarem prontos para posteriormente iniciar o processo de qualidade.

DESENVOLVIMENTO

Esta pesquisa se baseou em um estudo da documentação citada na revisão de literatura, sobre a importância da utilização de métodos Ágeis para automação de testes visando a qualidade, foi desenvolvido um programa com a ferramenta JUnit, para apresentar provas e exemplos de que a automação de testes pode ser uma grande aliada para a equipe de QA (Garantia de qualidade).

Seguindo as etapas de codificação e desenvolvimento do programa com ferramentas e técnicas que auxiliaram no processo de manutenção do *software*, evitando erros gerais durante o desenvolvimento, sustentação e evolução do sistema.

O teste de automação foi desenvolvido no laboratório da Universidade de Araraquara (UNIARA) onde foi utilizada as ferramenta e tempo para configuração de cada teste, assim como análise dos resultados obtidos.

O desenvolvimento do programa de automação de teste método Ágil, seguindo modelo apresentado *Sommerville* (2011) onde o cliente participa no processo de teste e na elaboração dos critérios para aceitação visando o nível de qualidade. O teste foi executado com ferramenta JUnit. Após avaliação foi considerada mais adequada a ferramenta *Cucumber*, primeiramente pela afinidade dos frameworks complementares ao Cucumber, por também não possuir a necessidade de criar um ambiente próprio só para automatizar e por ser compatível.

Inicialmente gerou-se código de classe '*Cadastro.Feature'*, o campo é preenchido com informações do usuário, onde faz o cadastro de e-mail, nome, data, endereço, número de telefone, e geração de senha de acesso. Conforme Figura 1 da tela do usuário:

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

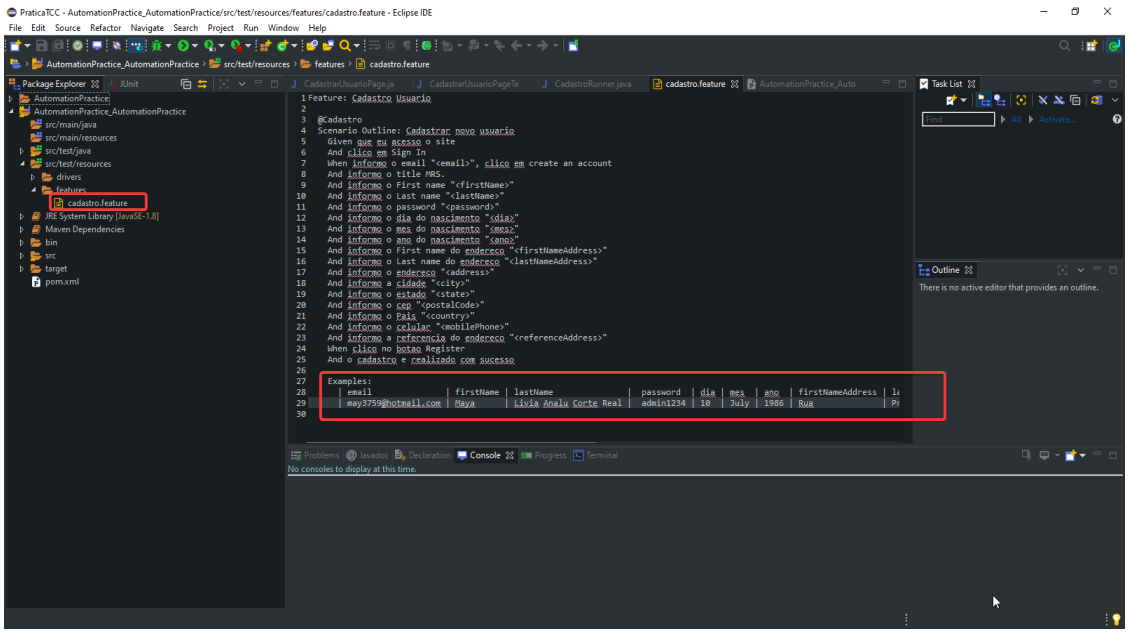


Figura 1- Cadastro do usuário- Fonte: Autor

A práticas de codificação auxilia a legibilidade e o ciclo de evolução e fica mais ágil e fácil. Endentação e comentários no código são relevantes, pois auxiliam no entendimento do código-fonte. O uso de pacotes, orientação a objetos e padrões de projeto também contribuem na organização e divisão das responsabilidades para reaproveitamento de código, aumentando a produtividade das equipes de desenvolvimento

A classe 'Utils.java' possui os elementos para acessar o navegador, abrir o site, e por sua vez direcionar os respectivos comandos para cada campo, verificando o cadastro e no final do programa fechar o navegador. Conforme Figura 2 e 3:

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

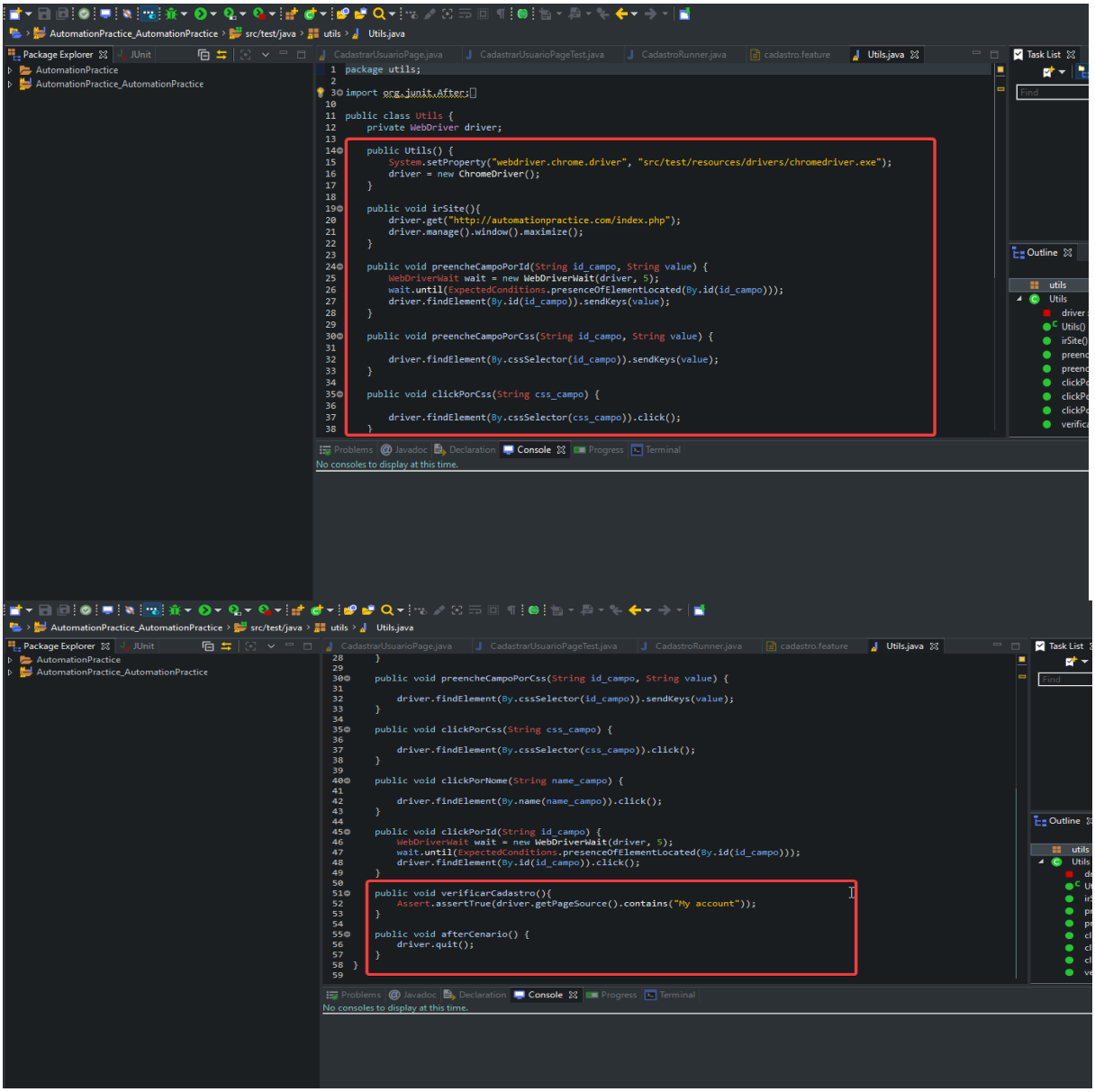
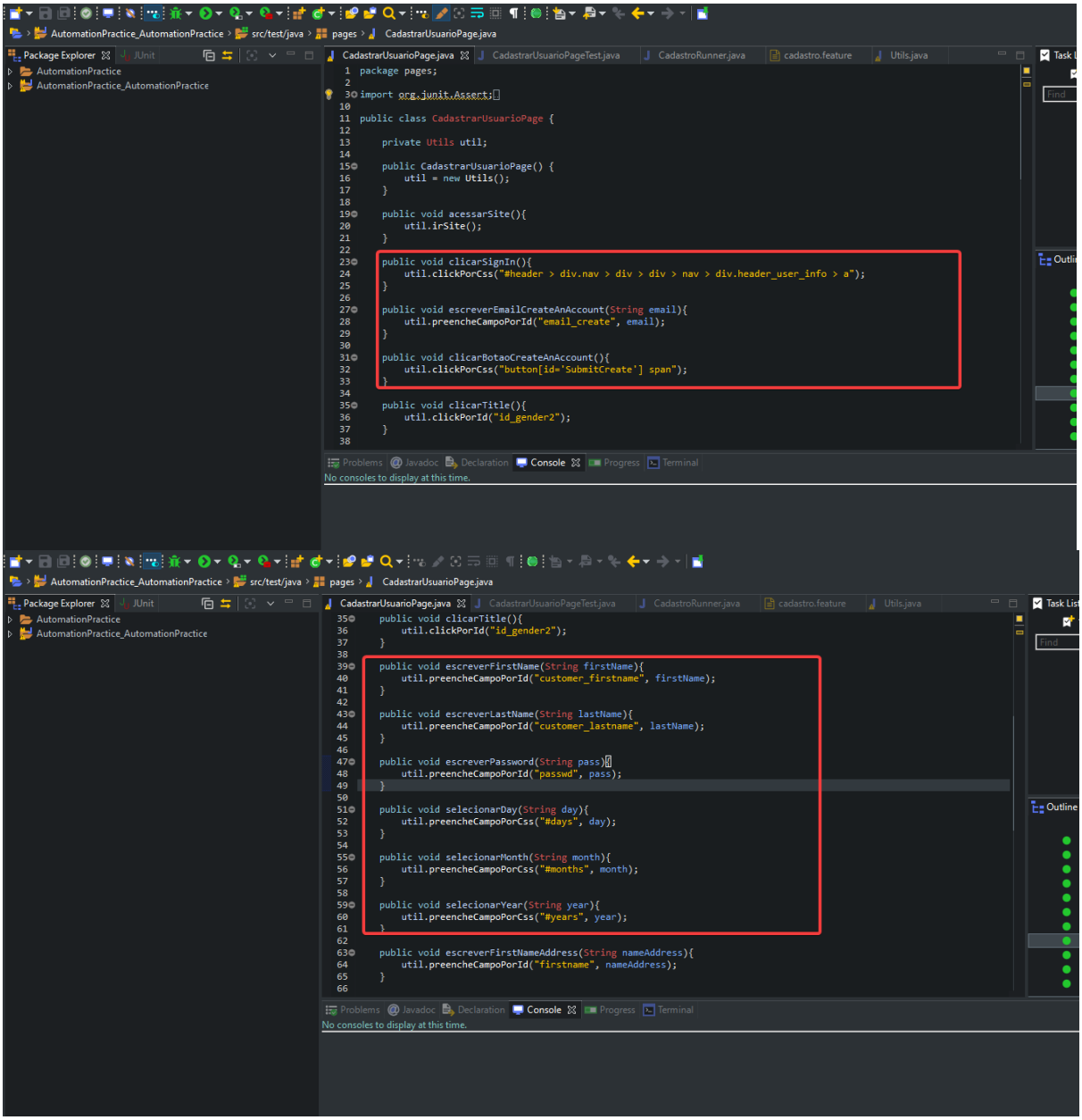


Figura 2-3 - Acesso ao navegador-Fonte: Autor

Na Figuras 4 e 5 é demonstrada a classe ‘CadastrarUsuarioPage.java’, nessa classe são adicionadas as informações contidas nas classes ‘cadastro.feature’ e ‘Utils.java’, que irá indicar ao teste automatizado os botões que precisa clicar para realizar os cadastros e os campos devidamente preenchido, com data de nascimento(dia, mês e ano), Número de telefone, Nome, Endereço, Cidade, etc.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218




```
package pages;

import org.junit.Assert;

public class CadastrarUsuarioPage {

    private Utils util;

    public CadastrarUsuarioPage() {
        util = new Utils();
    }

    public void acessarSite(){
        util.irSite();
    }

    public void clicarSignIn(){
        util.clickPorCss("#header > div.nav > div > div > nav > div.header_user_info > a");
    }

    public void escreverEmailCreateAnAccount(String email){
        util.preencheCampoPorId("email_create", email);
    }

    public void clicarBotaoCreateAnAccount(){
        util.clickPorCss("button[id='SubmitCreate'] span");
    }

    public void clicarTitle(){
        util.clickPorId("id_gender2");
    }

    public void escreverFirstName(String firstName){
        util.preencheCampoPorId("customer_firstname", firstName);
    }

    public void escreverLastName(String lastName){
        util.preencheCampoPorId("customer_lastname", lastName);
    }

    public void escreverPassword(String pass){
        util.preencheCampoPorId("passwd", pass);
    }

    public void selecionarDay(String day){
        util.preencheCampoPorCss("#days", day);
    }

    public void selecionarMonth(String month){
        util.preencheCampoPorCss("#months", month);
    }

    public void selecionarYear(String year){
        util.preencheCampoPorCss("#years", year);
    }

    public void escreverFirstNameAddress(String nameAddress){
        util.preencheCampoPorId("firstname", nameAddress);
    }
```

Figura 4-5 - Classe ‘CadastrarUsuarioPage.java’ - Fonte: Autor

A classe ‘CadastrarUsuarioPageTest.java’ é onde serão realizados os comandos para que o teste automatizado ocorra, no caso desse programa esses comandos vão indicar quando o usuário deverá realizar algum procedimento especifico, baseado nos códigos colocados na classe ‘CadastrarUsuarioPage.java’ como por exemplo na imagem a baixo, que fala para o programa acessar o site, após acessar o site clicar no botão *‘Sign In’*, e após clicar no botão ele deve informar o e-mail e clicar no botão ‘*Create an account’.* Conforme Figura 6:

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

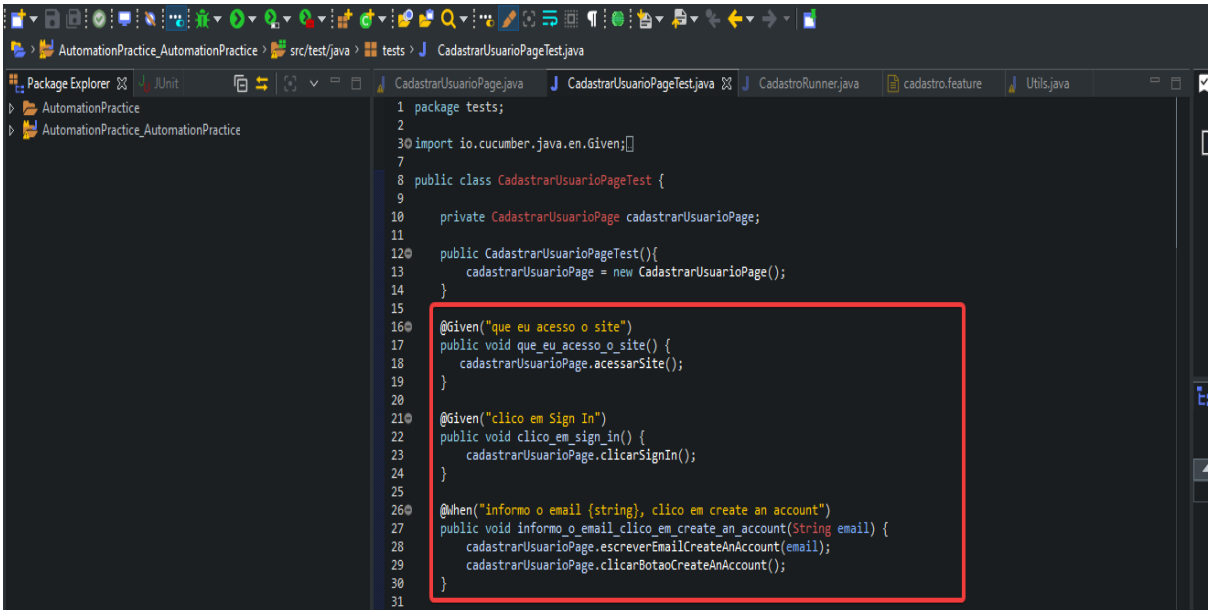


Figura 6: '*Create an account'- Fonte*: Autor

A Figura 7 refere-se a classe 'CadastroRunner.java', que é a classe responsável para compilar o programa, onde é realizado o apontamento das outras classes, informando para rodar o programa trazendo complementos necessários das outras classes.

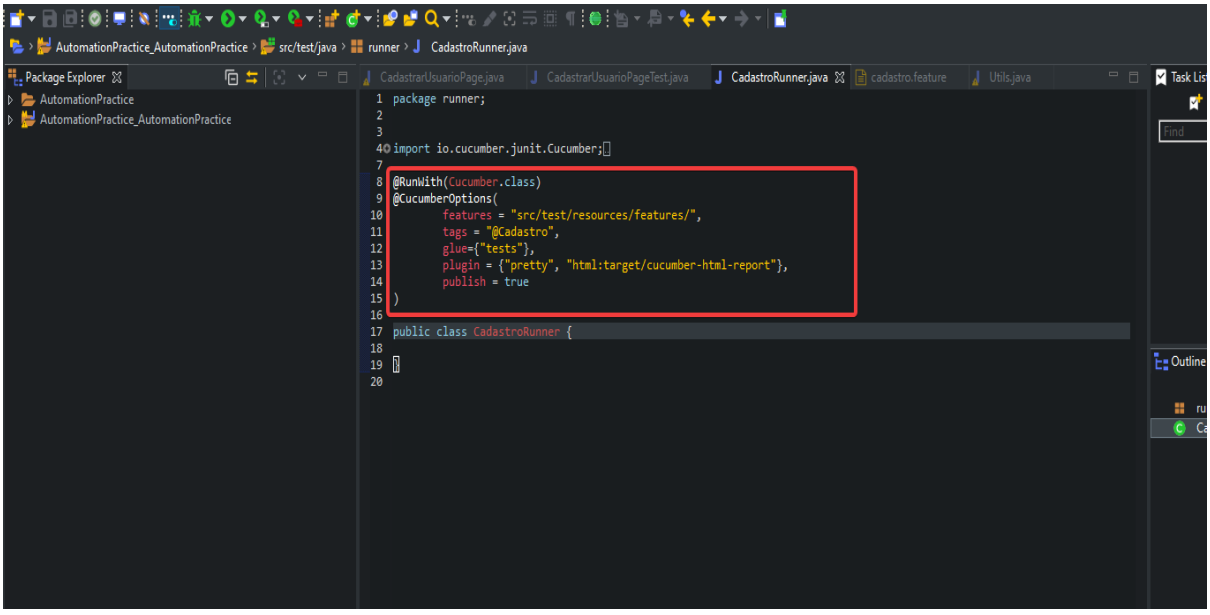


Figura 7- 'CadastroRunner.java'- Fonte: Autor

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

RESULTADOS

Nesta seção são apresentados alguns dados relacionados aos testes que foram automatizados após a geração de os códigos de acesso. Para a execução do teste deve-se clicar no botão '*Sign In*, conforme indicativo na Figura 8:

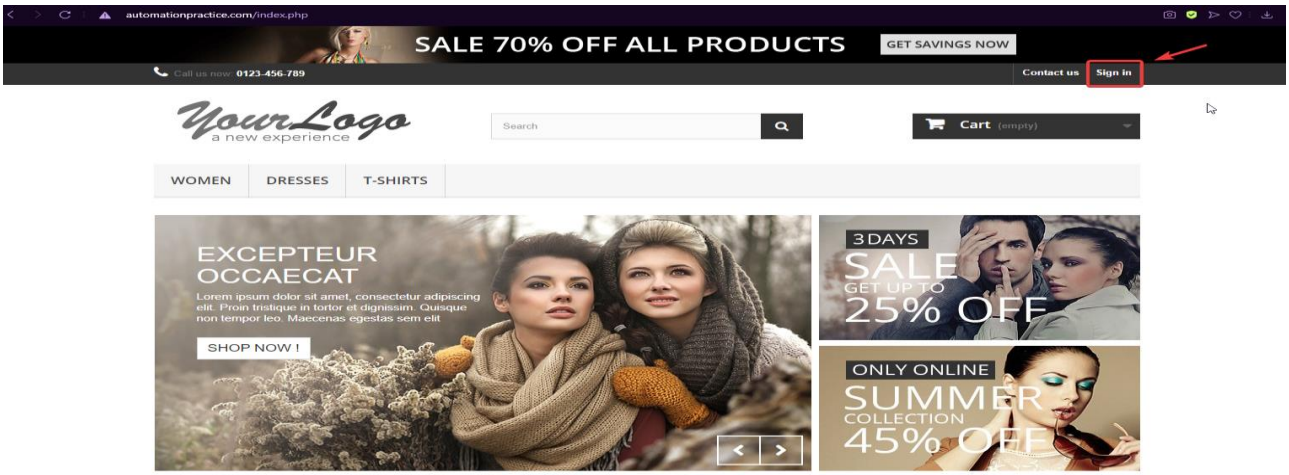


Figura 8- 'Sign In':Fonte- Fonte: o autor

Após clicar em *'Sign* In' o site irá entrar na tela de cadastro ou login, o programa irá colocar o e-mail informado na classe '*cadastro.feature'* e após isso irá clicar em '*Create na account'* Conforme indicação na Figura 9:

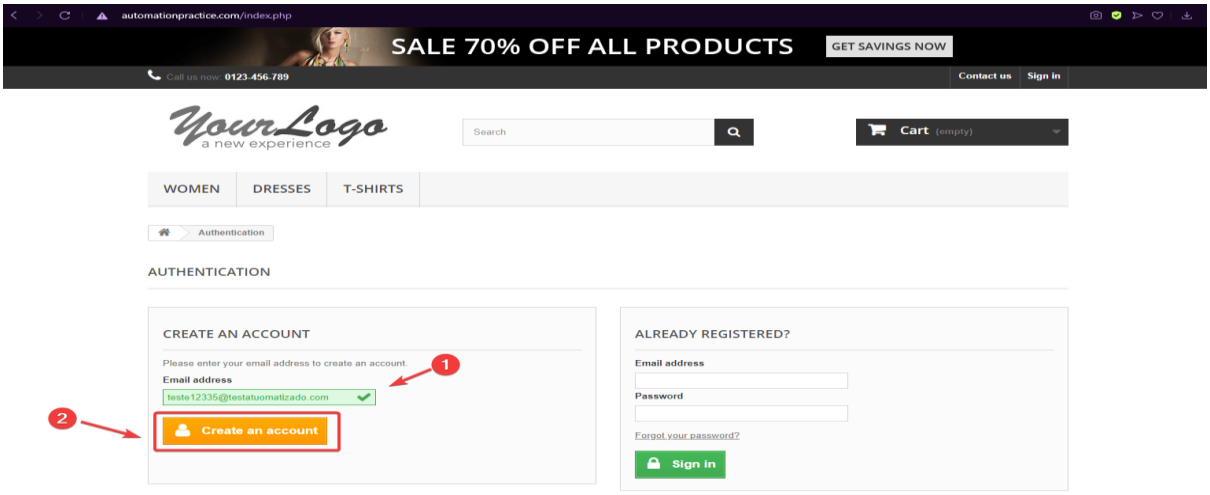


Figura 9- Create an account: Fonte: Autor

Caso o usuário não tenha se cadastrado e ocorrera o preenchimento automaticamente com os dados cadastrados no 'CadastrarUsuarioPage.java': Conforme ilustrado na Figura 10:

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

'Utils.java'. é a classe que possui os elementos que servem para acessar o navegador, abrir o site, e por sua vez direcionar os respectivos comandos para cada campo, verificando o cadastro e no final do programa fechar o navegador.

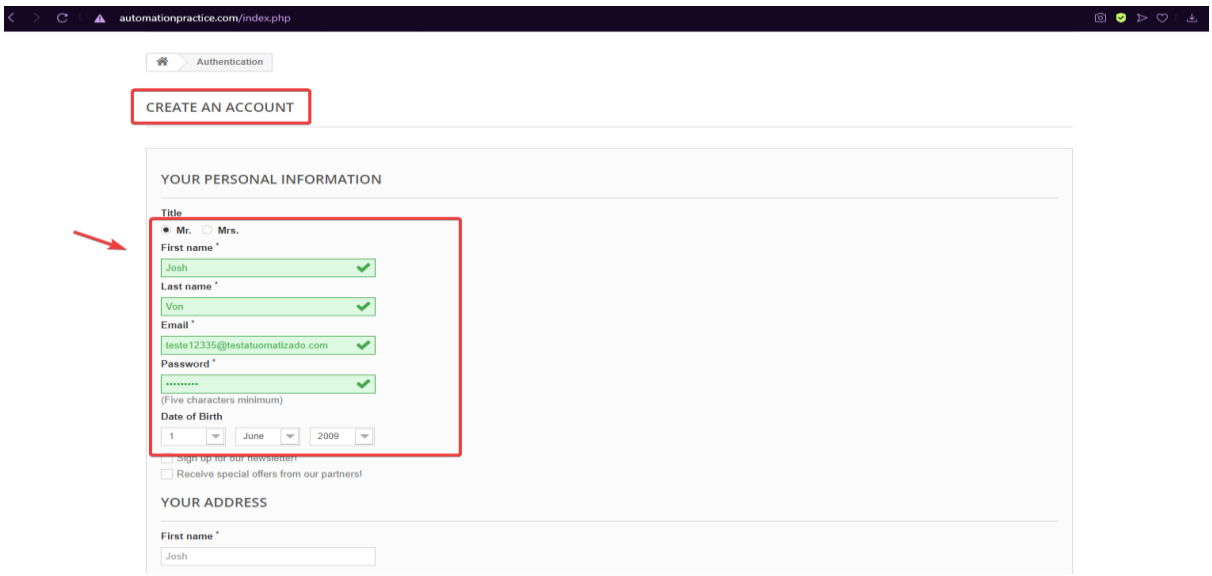


Figura 10- 'CadastrarUsuarioPage.java'-Fonte: Autor

Caso o e-mail já tenha sido criado ele irá retornar a mensagem e não irá continuar o teste, até que cadastre um e-mail valido, de acordo com a Figura 11:

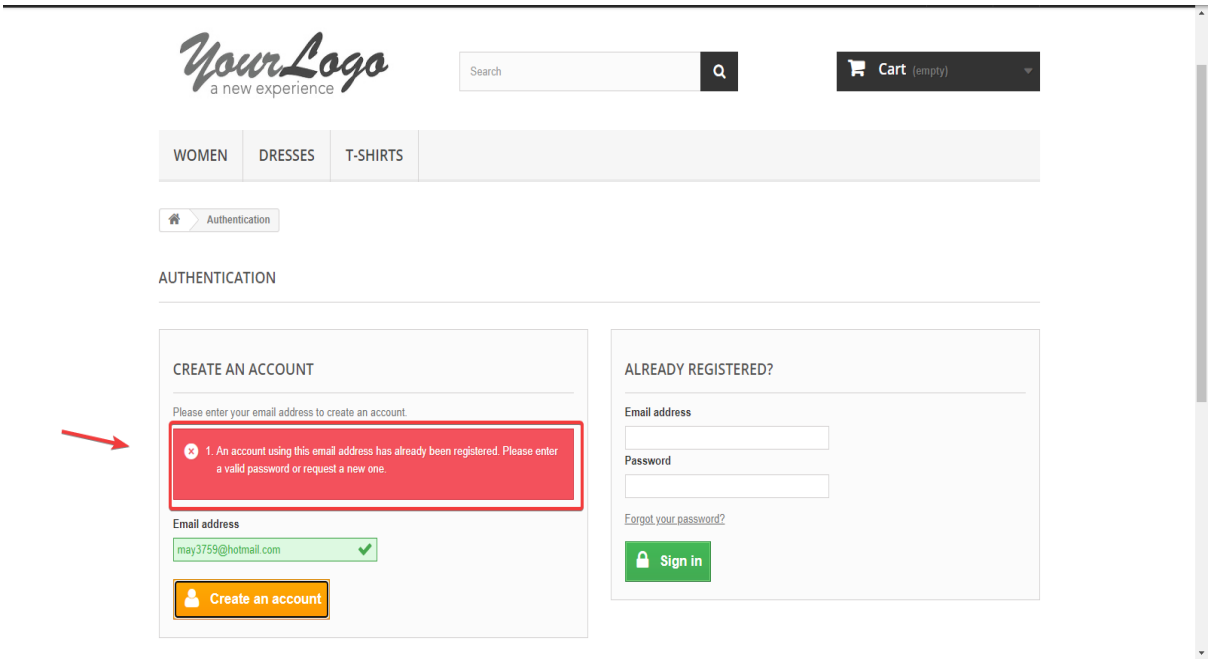


Figura 11- Cadastro de e-mail valido -Fonte: Autor

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Ao rodar o programa com sucesso, o site será fechado e o e-mail cadastrado corretamente, sendo possível verificar no console da plataforma em que o código foi feito o tempo que levou para o programa realizar o teste, e também se foi compilado corretamente, pelo fato de utilizar também o plug-in Cucumber, A ferramenta Cucumber, tem afinidade dos *frameworks* complementares, após finalizar o programa ele retorna um link onde é possível verificar um relatório que mostra o tempo que levou o processo e se encontrou erros ou não, link esse que está destacado na Figura 12 e 13 :

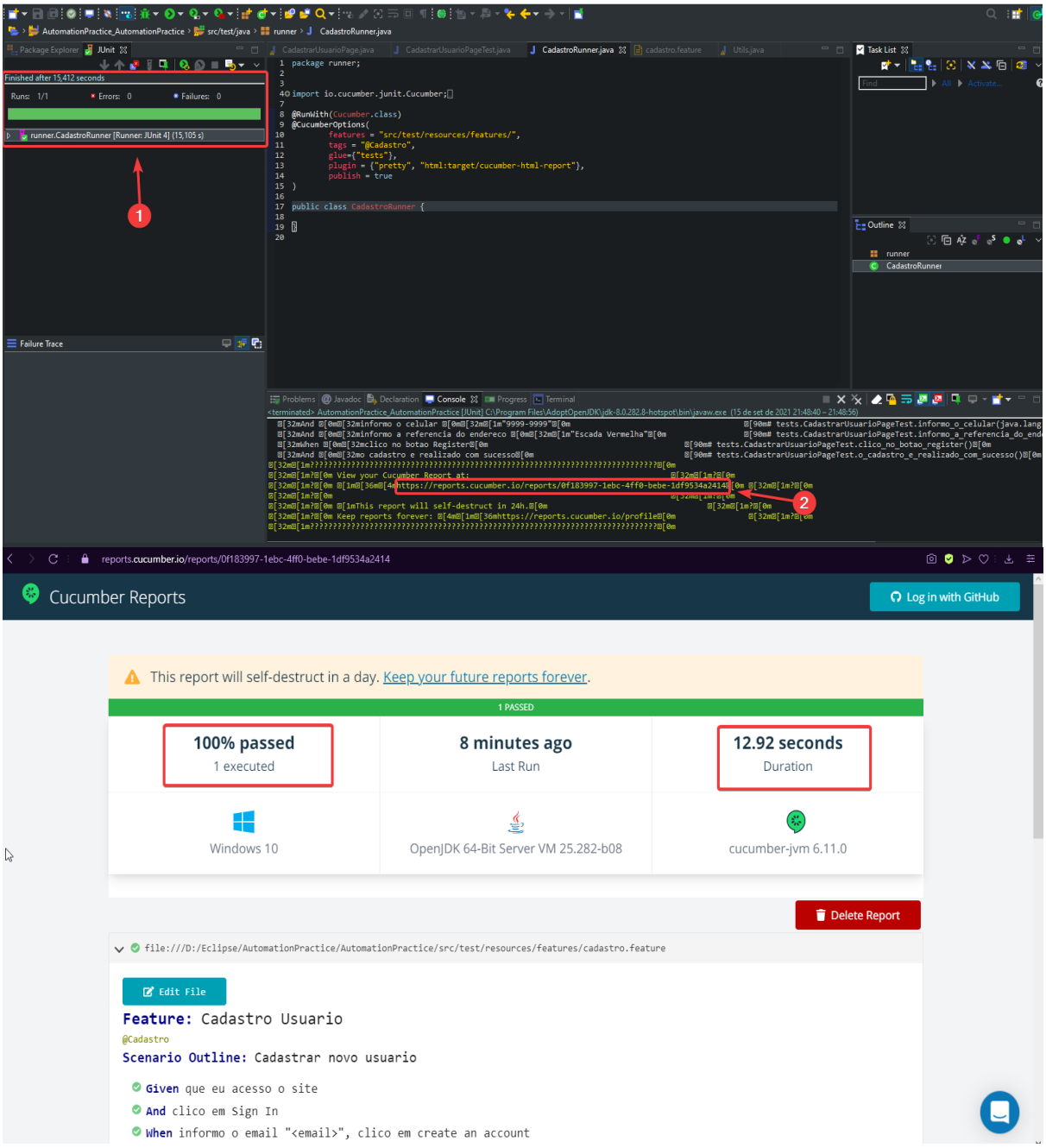


Figura 12 e 13- Relatório tempo de execução -Fonte: Autor

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

O Cucumber conta também com recursos com uso de *plug-ins* de *status report* para reproduzir relatórios dos cenários elaborados, exibindo as falhas e aprovações. O mesmo ocorre quando o usuário tenta cadastrar um e-mail já existente, como destacado na Figura 14 e 15:

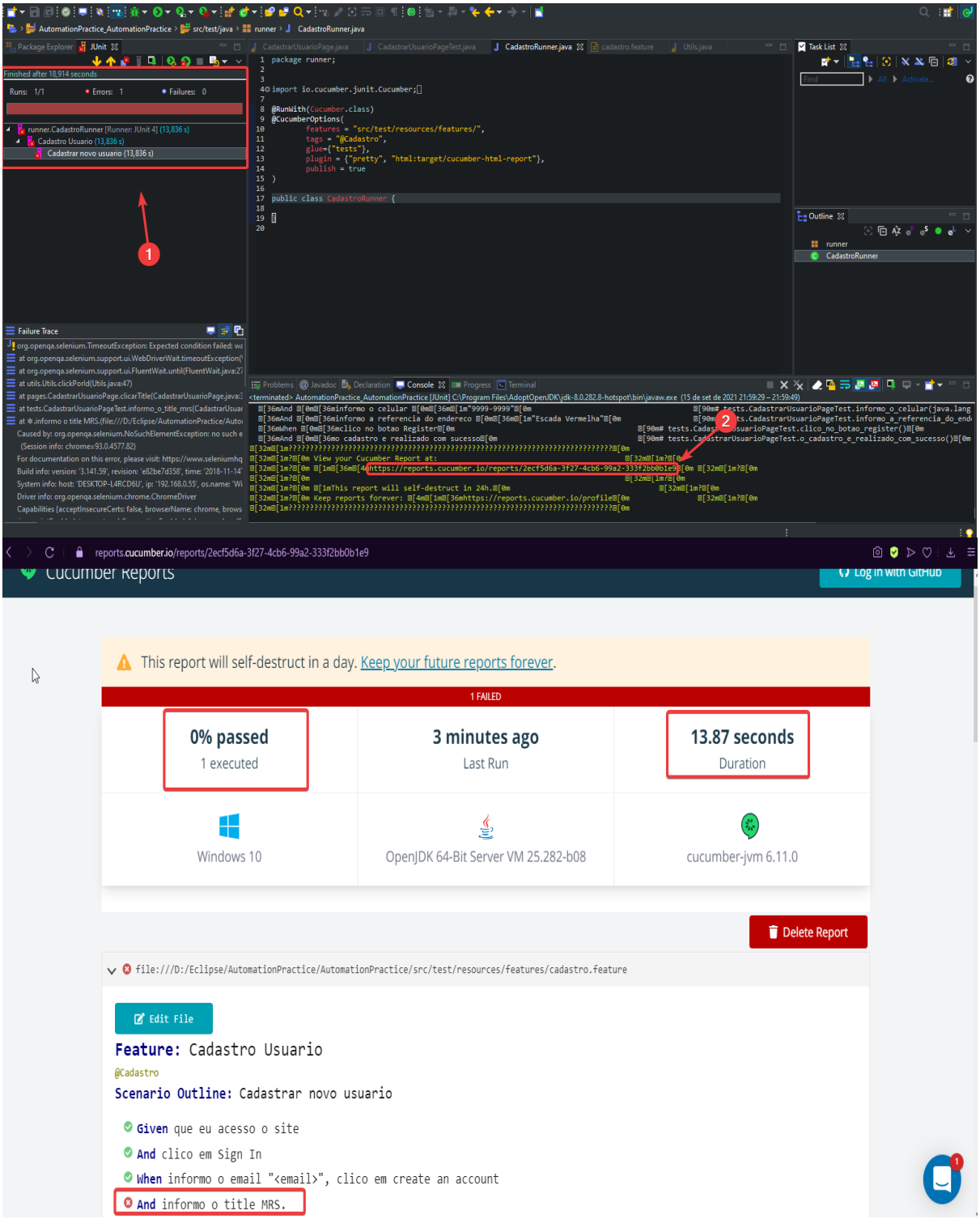


Figura 14 - 15: Relatórios dos cenários: Fonte: Autor

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Os testes automatizados foram executados de forma única, ao analisar o estudo, foi possível observar o tempo reduzido de execução do teste, sendo possível chegar ao um melhor entendimento sobre os conceitos e métodos de testes.

CONSIDERAÇÕES FINAIS

Este artigo teve como objetivo desenvolver um programa de automação de teste com a ferramenta Cucumber. Com cunho bibliográfico das fontes de pesquisas, foi possível alcançar o objetivo propostos.

Em seguida foi realizada descrição detalhada do projeto de automatização, com as ferramentas e tecnologias utilizadas como o Cucumber Utils.java' e JUnit de modo a compreender a funcionalidade dos testes. Os estágios do Cucumber foram executados em uma linguagem comum, clara e objetiva.

É possível concluir dizendo que, o teste de automação com o Cucumber oferece uma camada de comunicação real sobre um *framework* de teste e que durante o desenvolvimento verificou-se que o processo de automação pode reduzir e muito o tempo que um analista de testes empreende em uma tarefa quando comparado com o processo manual.

REFERÊNCIAS BIBLIOGRÁFICAS

BERNARDO, P. C. **Padrões de testes automatizados**. 2011. 221f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2011.

CAETANO, C. **Automação de Testes**: Engenharia de Software Magazine - Melhores Práticas na Automação de Testes. 5ª ed. Rio de Janeiro: Devmedia, 2008. p. 42. Disponível em: https://www.devmedia.com.br/automacao-de-testes/10249. Acesso em: 10 abr. 2021.

FERNANDES, M.; FONSECA, S. T. **Automação de testes de software**: estudo de caso da empresa Softplan. 2020. Monografia (Bacharel) - Universidade do Sul de Santa Catarina, Florianópolis, 2020.

FOWLER, M. **Test Pyramid**. [*S. l.: s. n.*], 2013. Disponível em: https://martinfowler.com/TestPyramid.html. Acesso em: 10 abr. 2021.

ISTQB, International Software Testing Qualifications Board. **Foundation Level Syllabus – BSQTB**. [*S. l.*]: ISTQB, 2018.

MENDONÇA, J. M.; SILVA, R. M. S. **Técnicas de usabilidade e testes automatizados em processos de desenvolvimento de software empírico**. 2014. 113f. Monografia (Bacharelado em Engenharia de Software) - Universidade de Brasília, Brasília, 2014.

MESZAROS, G.; WESLEY, A. **XUnit Test Patterns**: Refactoring Test Code. [*S. l.: s. n.*], 2007.

MOLINARI, Leonardo. **Testes Funcionais de Software**. Florianópolis: Visual Books, 2008. 214 p.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

NEGRELLO, A. **Métodos Ágeis e Qualidade**: como conciliar? [*S. l.*]: IBM, 2013. Disponível em: https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/m_c3_a9. Acesso em: 11 maio 2018.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de Software**. 3. ed. Rio de Janeiro: Alta Books, 2013. 290 p.

SOMMERVILLE, I. **Software Engineering**. 9th ed. São Paulo: Pearson Addison Wesley, 2011.

VELOSO, L. M. **Testes automatizados asseguram a qualidade dos softwares**. [*S. l.*]: Administradores.com, 2014. Disponível em: http://www.administradores.com.br/artigos/tecnologia/testes-automatizados-asseguram-a-qualidade-dos-softwares/80350/. Acesso em: 20 abr. 2021.

WOOTTON B. **Preparing for Continuous Delivery**. [*S. l.: s. n.*], 2013. Disponível em: https://dzone.com/refcardz/preparing-continuous-delivery?chapter=1. Acesso em: 12 maio 2021.