Automatizando Testes Funcionais em uma Esteira de Desenvolvimento de Software

Maxwandell S. de Azevedo[1], Daniel Faustino Lacerda de Souza[2]

[1]Departamento de Ciências Exatas (DCX) – Universidade Federal da Paraíba (UFPB)
Rua da Mangueira, s/n, Companhia de Tecidos - Rio Tinto – Paraíba - PB – Brasil

Abstract. *Quality guarantee, no software development is a concern for large projects, much is studied for saving or final product. Software test is the final phase for quality assurance, in this phase errors are detected that may occur in production, errors that may compromise the final quality of the product. In view of this final phase of development, this article presents the automation of tests that work on a software development path, using ferramentas and frameworks for the optimization of the implementation of this process.*

Resumo. *Garantia de qualidade ao longo do desenvolvimento de software é uma preocupação para grandes projetos, muito se estuda para melhorar o produto final. Teste de software é a fase final para garantia de qualidade, nessa fase são detectadas erros que poderiam ocorrer em produção, erros esse, que podem comprometer a qualidade final do produto. Tendo em vista essa fase final do desenvolvimento, este artigo apresenta a automatização de testes funcionais em um esteira de desenvolvimento de software, utilizando ferramentas e frameworks para a otimização desse processo.*

1. Introdução

No processo de desenvolvimento de *software*, garantir a qualidade do produto é um enorme desafio em consequência da alta cobrança para realização de entregas, complexidade do produto e diversos outros fatores que envolvem o processo de desenvolvimento. Uma maneira eficaz de garantir qualidade no processo é através da verificação e validação (V&V), cujo objetivo é mostrar que um sistema está em conformidade com sua especificação e que atende as expectativas do cliente (Sommerville, 2011). Teste de *software* é um importante passo da validação do sistema, Inthurn (2001) ainda complementa, "teste de *software* tem como objetivo aprimorar a produtividade e fornecer evidências de confiabilidade, em complemento a outras atividades de garantia de qualidade ao longo do processo de desenvolvimento do *software*".

Teste de *software* é uma atividade complexa, muitas empresas pode não adotar o método no desenvolvimento do seu produto devido a diversos fatores como limitações de tempo, qualificação dos desenvolvedores envolvidos no ciclo de desenvolvimento do produto, por falta de conhecimento de ferramentas que podem ajudar nessa atividade e outros motivos relacionados às mudanças e implementações de novas funcionalidades

"Trabalho de conclusão de curso, sob orientação do professor Daniel Faustino Lacerda de Souza submetido ao Curso de Licenciatura em Ciência da Computação do Centro de Ciências Aplicadas e Educação (CCAE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de LICENCIADO EM CIÊNCIA DA COMPUTAÇÃO."

feitas a pedido do cliente. Mas em um mercado competitivo, a inclusão de testes na esteira de desenvolvimento se torna imprescindível para garantir a qualidade do produto, uma vez que, a inclusão desse ciclo no processo minimiza erros, torna o produto confiável, garantindo assim uma relação de confiança da empresa com o cliente.

Para a implementação de testes no ciclo de desenvolvimento é necessário planejar e estruturar as etapas, as atividades, os artefatos, os papéis e as responsabilidades do teste, por fim, é necessário seguir uma abordagem de teste. Muito se ouve falar da abordagem de testes automatizados, uma das vantagens de utilizar essa metodologia é permitir que ele seja repetido várias vezes, sendo mais fácil encontrar novos erros através da repetição e da simulação de cenários específicos. Os testes automatizados são programas ou scripts simples que exercitam funcionalidades do sistema testado e fazem verificações automáticas nos efeitos colaterais obtidos (Bernardo e Kon, 2008).

Atualmente, a empresa ESIG/Software é um player do mercado de desenvolvimento de *softwares* para gestão educacional baseada no Rio Grande do Norte. Uma das características atuais da empresa no que diz respeito à disciplina de testes de *software* é a execução de testes de validação de interfaces de forma manual. Apesar da eficácia dos testes de validação realizados no sistema dentro do processo de desenvolvimento, o fato de serem realizados de forma manual pode ocasionar problemas como, repetições de teste impactando em uma tarefa muito cansativa e com tendência a erros. Nesse sentido, esse estudo tem como intuito analisar e auxiliar o processo de desenvolvimento do *software* acadêmico da empresa ESIG/Software através dos testes automatizados.

O objetivo desse artigo é apresentar a implementação dos testes funcionais automatizados na esteira de desenvolvimento da aplicação acadêmica de empresa ESIG/Software, partindo desde a definição da ferramenta a ser utilizada para execução automatizada dos testes, passando pelo processo de especificação e realização de validação do uso da ferramenta por meio da construção e execução de baterias de testes para um dos produtos da empresa. Por fim, deverá ser realizada uma análise do impacto que a inclusão desse procedimento causa ao processo de desenvolvimento.

Este trabalho está dividido nas seguintes seções: a seção 2 trata sobre temas relacionados ao trabalho, trazendo um pouco da fundamentação teórica referente à disciplina de qualidade e testes; a seção 3 traz uma visão geral da empresa por meio de sua caracterização, ferramentas de gestão e limitações; a seção 4 detalha a proposta e execução da implementação e transição dos testes manuais para os automatizados; por fim, a seção 5 apresenta as considerações finais.

2. Fundamentação Teórica

Nesta seção será descrito conceitos relacionados ao trabalho. São apresentados, qualidade de *software*, processos de *software* e testes de *software*.

2.1. Qualidade de software

A qualidade de *software* é um processo relacionado ao desenvolvimento, o intuito é garantir que o *software* será entregue ao fim do processo atendendo às expectativas da empresa e do cliente, um produto desenvolvido com a utilização de boas práticas e

técnicas de desenvolvimento adequadas. Qualidade de *software* é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos prevenindo e eliminando defeitos, como afirma Bartié (2002).

Entretanto, qualidade de *software* é uma discussão complexa e difícil de ser conceituada, é normal encontrar opiniões divergentes sobre o assunto, muitas delas abordam o fato que o *software* deve atender os requisitos, ou seja, as necessidades do usuário. Segundo Magalhães (2006), para que um *software* seja considerado de qualidade é preciso que esteja em conformidade com os seus requisitos, atenda aos requisitos e expectativas do cliente e seja bem aceito por seus usuários. O debate tem o ponto de vista do desenvolvedor sobre qualidade de *software*, muitos abordam que o *software* é de qualidade quando desenvolvido com boas práticas, técnicas adequadas e atendendo a todos os requisitos estabelecidos pelo cliente. Então, qualidade de *software* é um dos pré-requisitos básicos para qualquer empresa que tem o objetivo de impulsionar seu produto no mercado. Na medida em que o mercado competitivo oferece produtos com demandas de alta complexidade e um tempo de entrega cada vez menor, o conceito de qualidade adotado ao longo deste trabalho é o custo/benefício e satisfação ao cliente, para elevar o produto e alcançar cada vez mais importância em uma área bastante competitiva.

Apesar do conceito de qualidade em um primeiro momento parecer óbvio, quando observada com maior proximidade, a complexidade do assunto se torna um pouco maior. Para gerir a qualidade de um *software* existem técnicas, ferramentas, etapas de controle e garantias que serão adotadas em meio ao processo de desenvolvimento do *software*.

2.2. Teste de Software

Em um cenário de desenvolvimento do produto, pode ocorrer erros em diversas partes do ciclo de vida. Nesse caso, o teste de *software* não é algo dispensável e sim uma atividade crucial no processo de desenvolvimento de um produto que deseja entregar qualidade, os testes são as melhores alternativas para empresas minimizar erros e garantir a qualidade do produto. Quanto mais cedo no processo de desenvolvimento de um *software* um bug for encontrado, menor será o custo de sua correção. *Bugs* encontrados pelos programadores custa R$ 10 para consertar e *bugs* encontrados pelos clientes custam US $1.000 para consertar (Black 2000).

A implementação de testes na esteira de desenvolvimento também possui seus processos, cada etapa deve-se traçar objetivos e possuir resultados,ou seja, ela ocorre de forma sistemática seguindo um cronograma de atividades previstas.

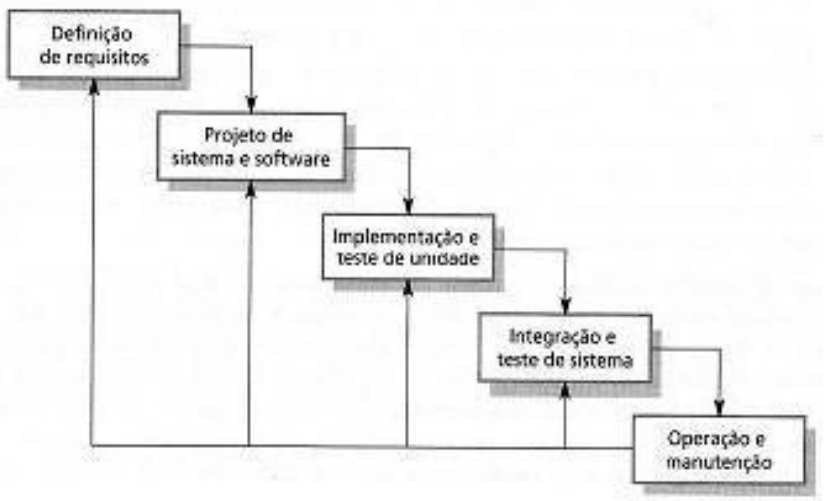

Figura 1. Etapas de testes de *software*

fonte: https://s3.amazonaws.com/qcon-assets-production/images/provas/25803/Imagem%20008.jpg

A fase de testes varia em relação ao modelo de ciclo de vida escolhido, mostrando de forma genérica as fases de testes, geralmente temos a definição dos requisitos uma das primeiras fases definidas, esta etapa contempla a escolha dos tipos de testes, casos de usos, requisitos funcionais e não funcionais. O projeto de teste é a etapa associada a recomendações e análise para a escolhas de *software* de teste e também é realizado a elaboração de um plano de testes. A implementação é a fase que contempla a elaboração dos casos de testes e scripts, onde os dados são avaliados e posteriormente editados novamente. A integração dos testes é a etapa que valida os casos de testes apontados anteriormente, realizando ações para documentar os resultados obtidos e reportar os problemas encontrados. Na operação e manutenção geralmente são realizadas ações para discutir e corrigir os resultados obtidos durante as fases anteriores, logo após, refazer todas as validações novamente. Segundo (OLIVEIRA, 2010), a fase de teste deve iniciar-se desde das especificações dos requisitos do sistema, ou seja, logo que se inicia o projeto de desenvolvimento. É necessário que para o sucesso das etapas de testes mostrada na Figura 1 ela se inicie em conjunto com o desenvolvimento.

Ainda nesse contexto, além das fases de testes, temos os tipos dos testes que podem ser implementados ao processo, cada teste tem suas especificidades e é necessário uma análise do que o produto precisa para introduzir de forma mais eficaz o tipo de teste ao que se encaixa melhor ao processo. A seguir serão apresentados os tipos de testes mais utilizados e seu funcionamento (PERES, 2009).

Em muitos casos o teste unitário é a melhor alternativa para introduzir ao processo, nele podemos testar cada unidade de forma isolada, cujas ações podem receber algum argumento ou em casos que retornam algum valor. Em outros casos podemos fazer os teste de regressão, se o processo de desenvolvimento passar por grandes modificações e por repetidas inclusões de novas funcionalidades o teste de regressão é o ideal para introduzir, nesse tipo de teste é realizado a verificação de

possíveis impactos que as mudanças podem causar ao código-fonte, comprometendo a lógica já escrita e causando mudanças a requisitos já brevemente entregue. Temos a possibilidade de automatizar os testes através dos teste de automação, o objetivo desse tipo de teste é diminuir o tempo gasto reduzindo e simplificando os scripts, tendo a possibilidade de repetir o conjunto de testes várias vezes reduzindo o esforço humano e a chance de falhas.

3. Visão geral da empresa

Esta seção aborda uma visão geral da empresa por meio de sua caracterização, ferramentas de gestão e limitações.

3.1. Surgimento da empresa

Por volta dos anos 2002 e 2003 surgiu os sistemas SIG/SIGEduc, o sistema SIG iniciou de uma necessidade da UFRN (Universidade Federal do Rio Grande do Norte) em ter um *software* de gestão educacional, já o SIGEduc atendia as necessidades da secretaria de educação do estado do Rio Grande do Norte. O início contava apenas alguns ex-alunos de áreas relacionadas à tecnologia da UFRN.

A universidade Federal do Rio Grande do norte na época não era uma referência na área da tecnologia da informação, mas, com a jornada de muito sucesso entre os anos de 2002 e 2010 com os sistemas implementados e em perfeito funcionamento tanto na universidade quanto na educação do estado do Rio Grande do Norte, passa ser referência nacional de um sistema de gestão educacional.

O sucesso dos sistemas SIG/SIGEduc é notada por diversos estados e universidades como solução para seus problemas de gestão digital, entre os anos 2010 e 2011 alguns desenvolvedores que iniciaram o desenvolvimento dos sistemas se unem para inicializar a empresa ESIG, tendo como objetivo entrar no mercado com o sistema e atender a necessidade das universidades e estados que tinham como objetivo um sistema de gestão educacional. A estruturação da empresa ESIG no mercado se dá gradualmente, mantendo a estabilidade ao longo dos anos a expandindo seus negócios e crescendo com a inclusão da empresa QUARK, formando a ESIG GROUP, empresas formadas pela ESIG/Software e QUARK/Tecnologia, com a ESIG tendo foco no desenvolvimento ao atendimento das necessidades do cliente e focando no seus produtos de gestão educacional os sistemas SIG/SIGEduc, já a QUARK tem um modelo mais voltado ao mercado buscando trazer soluções para áreas da saúde.

A expansão se tornou um sucesso no mercado e referência na área, no ano de 2021 a empresa ESIG GROUP, composta pela ESIG/Software e pela QUARK/Tecnologia foi premiada como a segunda melhor empresa para se trabalhar no Rio Grande do Norte pela GPTW.

3.2. Caracterizando a empresa

O desenvolvimento de um *software* envolve várias etapas, por isso, diversas dessas etapas são decisivas para o sucesso do desenvolvimento. Para isso, a empresa utiliza metodologias para auxiliar o gerenciamento do projeto, são elas: A metodologia ágil

Scrum e a metodologia OKR(*Objectives and Key Results* ou Objetivos e resultando Chaves, em português).

Ao inserir o Scrum no processo, a empresa segue passos necessários para atingir os objetivos. Um dos passos iniciais é documentar tudo que foi acordado entre ela e o cliente, é necessário registrar tudo e mapear as atividades de acordo com sua ordem de prioridade. Assim, a empresa terá uma visão geral das ações executadas, cada passo dado não será perdido ao longo do desenvolvimento, possibilitando executar melhor os passos seguintes. Essa documentação é gerenciada pela ferramenta Jira substituindo a idéia do “Quadro Scrum”, na empresa a utilização do Jira ocorre da seguinte maneira, o quadro é marcado pelas seguintes colunas: backlog, backlog priorizado, em execução, validação N1 e validação N2. no qual são representados por retângulo descrevendo as atividades e sendo movida entre as colunas de acordo com seu status.

Pautada com a metodologia, a empresa trabalha com ciclos dinâmicos denominados "sprints", ou seja, ciclos de trabalhos com durações fixas. A realização da sprint começa com a *Planning*. reunião de alinhamento com o que será realizado naquele ciclo, a sprint é finalizada com as conclusões de todas as atividades, com a reunião *Review* é exposto as atividades realizadas por cada membro da equipe, com discussões de pontos negativos e positivos, dificuldades e pontos de melhoria para o próximo ciclo.

Seguindo, temos os encontros diários de cada equipe, são expostas ações realizadas no dia anterior e ações que aconteceram no dia presente. Com isso, surge um sentimento positivo de cooperação e protagonismo em cada integrante da equipe, de modo que as ações ocorram de forma contínua e sem impedimentos.

A metodologia OKR se comunica perfeitamente com o *Scrum*, porque se trata de uma abordagem para criar metas e engajamento de todos das empresas de forma ambiciosa e consciente. Esse diálogo ocorre por ambas terem características muito próximas, como , alinhamento nas idéias, clareza na comunicação, engajamento e responsabilidades com metas propostas.

OKRs é a junção de objetivos (O) e Resultados Chaves (KRs). a empresa alinha com cada equipe os objetivos que devem ser alcançados ,ou seja, objetivos concretos, significativos e desafiadores. Já os resultados chaves são ações realizadas para se chegar ao objetivo. Essas ações são definidas pelos *Squads* (equipes) e são realizadas e documentadas ao longo do tempo, ao final, Os resultados chave são apresentados trimestralmente numa reunião com todos os *squads* com o intuito de acompanhar e verificar se os objetivos serão alcançados durante o processo.

Na empresa também é observado o processo de teste no sistema, cada membro do squad tem seu papel e responsabilidade relacionada a esse processo. Em alguns squads são feitas as implementações de testes unitários, também são realizados *Code Review* e testes manuais das tarefas da sprint ou tarefas requisitadas de aprimoramento, sempre realizadas pelos desenvolvedores da equipe.

3.3. Problemática

As tarefas são cadastradas no JIRA logo após a planning. Em um cenário comum as atividades ficam em backlog a espera de serem movimentadas pelos desenvolvedores, em sequência após algum desenvolvedor analisar o que deve ser feito e se responsabilizar para o desenvolvimento da tarefa o passo é movê-la para execução. Após chegar a uma solução a nível de código e implementá-la a tarefa é movida para validação N1.

Após esses passos o próprio desenvolvedor faz testes manuais, com propósito de encontrar possíveis erros que podem ter ocorrido ao longo do desenvolvimento. Caso ocorra algum erro o próprio desenvolvedor deve corrigi-lo e refazer o conjunto de testes manuais. A tarefa só é movimentada para validação N2 após o *Code Review* de outro membro da equipe. Esse cenário ocorre para identificar possíveis erros ocorridos durante a fase de desenvolvimento, geralmente também é realizado outro conjunto de testes manuais.

O processo manual de execução de um caso é rápido e eficaz, porém, o efeito a longo prazo pode não ser benéfico, é necessário um esforço significativo para se repetir os conjuntos de testes manuais a cada desenvolvimento das tarefas tornando uma atividade repetitiva e cansativa sujeita a erros. Em algumas ocasiões pode ocorrer erros que impactam em partes que estavam funcionando tornando um regresso e causando um impacto que pode levar a mais atividades de desenvolvimento ocasionando em tempo gasto tornando uma “bola de neve” acumulando e causando problemas.

4. Proposta

Esta seção mostrará a proposta de transição dos testes manuais para os automatizados a partir da escolha da ferramenta, o ambiente para o desenvolvimento dos testes e a validação do processo.

4.1. Automatização dos testes

O processo de execução de testes automatizados é uma escolha alternativa, porém, se comparado aos testes manuais a sua inclusão dentro da esteira de desenvolvimento acrescenta muito ao processo pois ganha em tempo, em execução dos casos de uso podendo se repetir várias vezes facilmente, produtividade porque requer poucos recursos humanos, ou seja, menos membros da equipe para testar a tarefa desenvolvida ocasionando uma maneira mais eficaz para detectar possíveis erros e *bugs*.

Levando em consideração o cenário de desenvolvimento do produto da empresa ESIG/Software, a escolha da ferramenta para tornar os testes manuais em testes automatizados é uma tarefa essencial. Esta etapa deve atender alguns requisitos, dentre eles, ser compatível com a plataforma de desenvolvimento utilizada dentro da empresa que tem como linguagem de programação base o Java. Para isso, foram feitas comparações de algumas ferramentas disponíveis no mercado para automação de testes. Na tabela 1 é apresentado um resumo das características importantes que devem ser atendidas pela ferramenta pretendida a fim de facilitar a integração ao processo de desenvolvimento atual das soluções da empresa.

Tabela 1 - Comparativo das ferramentas

Critérios	*Selenium Webdriver*	*Protractor*	*Cypress*	*Cucumber*
Utilização em IDEs: Eclipse e Intellij	X			
Escrita dos testes na linguagem Java	X			X
Opção para teste em diversos navegadores	X	X	X	X

A ferramenta Selenium WebDriver foi analisada, sendo um *framework* poderoso que possibilita a utilização de múltiplos navegadores, tais como *FireFox*, *Chrome*, *Safari* e o *Explore*. Outros pontos relevantes são o fato de ter uma instalação local simples e compatibilidade com a linguagem de programação Java.

O *Protactor* possui uma sintaxe simples, é uma *framework* para automação de testes em aplicações desenvolvida com AngularJS, construída utilizando como base o *framework* Selenium e também é compatível com diversos navegadores.

A ferramenta *Cypress* é uma ferramenta que não utiliza Selenium como base, ela foi construída especialmente para lidar com as aplicações que utilizam JavaScript moderno, os testes Cypress são escritas apenas em *JavaScript* e também é compatível com diversos navegadores.

O *Cucumber* é uma ferramenta voltada para os testes automatizados que foram criados em um padrão BDD(*Behavior Driven Developmen*t). Com o *Cucumber* você pode criar *scripts* de texto simples a partir da linguagem Gherkin.

A ferramenta Selenium WebDriver foi escolhida dentre as alternativas citadas acima visto que ela atende a todos os requisitos que observamos como necessários para auxiliar a transição dos testes automatizados para os testes manuais. Sendo uma das ferramentas mais viáveis da atualidade pelo fato de ser gratuita e ser bastante usada pela comunidade de *software*, gerando uma massa de fontes para auxiliar no desenvolvimento dos testes. A ferramenta pode utilizar java para a escrita do código, pode se integrar em IDEs como Eclipse e Intellij e outra vantagem é a versatilidade de execução em quase todos os navegadores. Vale ressaltar que um dos pontos mais importantes é que como a ferramenta pode ser usada em ambientes como Eclipse e Intellij, facilita integração dos testes com o ambiente de desenvolvimento do sistema SIGEduc que utiliza o eclipse como seu ambiente e está migrando para a utilização do Intellij, por isso , os testes foram realizados na IDE do Intellij.

4.2. Ambiente de automação

A configuração do ambiente de automação dos testes é uma parte importante na implementação, para isso, foram utilizadas algumas ferramentas. Como dito anteriormente, a principal ferramenta para automação será a Selenium WebDriver, sendo utilizado o intellij como IDE e o JUnit dará apoio para a automação dos testes, inicialmente pensado para os testes de unidade o JUnit é um *framework* poderoso e que auxiliará a transição dos testes manuais para os testes automatizados.

O JUnit ficou encarregado de fazermos a verificação dos nossos casos de testes, realizar a verificação das saídas dos casos de teste comparando valores esperados com valores retornados e estruturar a nossa automatização através das anotações (Annotations).

Já o Selenium é um framework para testes funcionais em aplicações web, ela usa o próprio driver do navegador para a automação, ele carrega a página em um navegador embutido e executa ações para simular o comportamento de um usuário. Para isso acontecer, o testador constrói os scripts utilizando a ferramenta. Em seguida descrevemos um pouco da estratégia de construção dos testes automatizados utilizando a ferramenta bem como seu ambiente de execução dos testes.

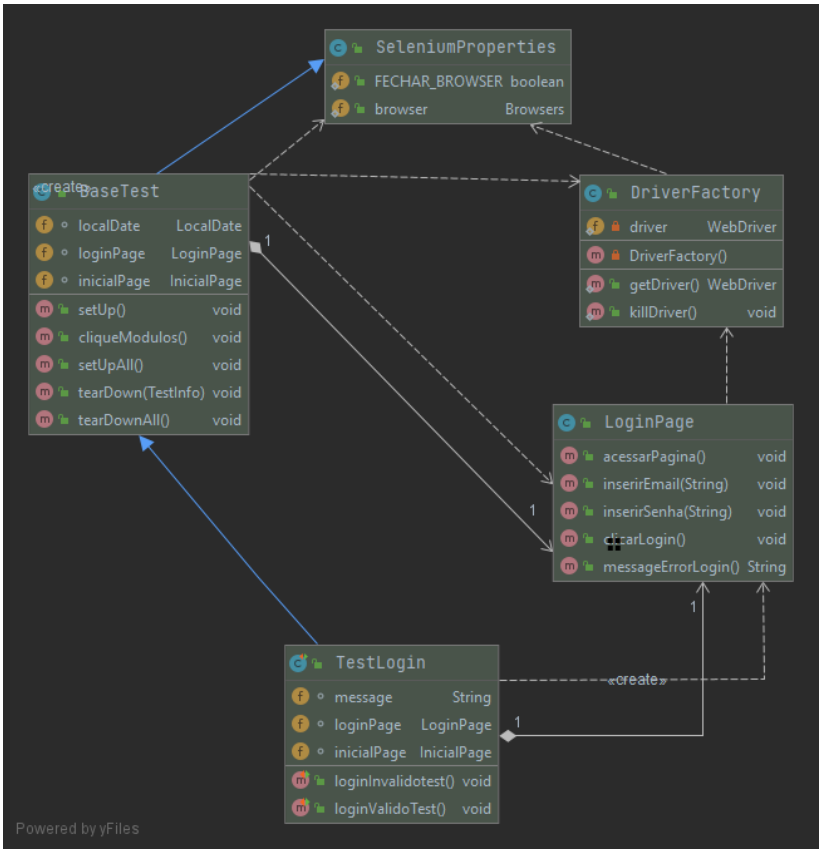

Figura 2. Diagrama de classes do projeto de automação

fonte: Autoria própria

A Figura 2 ilustra um diagrama de classe com os relacionamentos básicos para definição e escrita de novos casos de teste a partir do framework do Selenium. A classe "*SeleniumProperties*" que vemos na Figura 2, é utilizada para definição dos navegadores que são utilizados para elaborar os testes e também faz a verificação para fechar o browser. A "*DriverFactory*", atualmente está "setando" a informação de qual navegador usar através do método "*getDriver*", o "*killDriver*" é utilizado para finalizar o driver. Ainda na Figura 2 observamos a classe "*BaseTest*", ela contém métodos de refatoração do código, ou seja, nela fazemos a reutilização de código que possivelmente iria ser repetida em grande parte do projeto, utilizamos o JUnit para auxiliar o processo de refatoração e também utilizar de forma otimizada recursos do Selenium WebDriver como tirar uma foto a cada execução dos testes e finalizar o navegador ao final de todos os testes. Os métodos utilizam anotações do JUnit, no método "setUpAll" utilizamos a anotação @beforeAll, indicando que antes de cada execução do teste nas classes que utilizam o @Test, as ações que estão no corpo do método deverão ser executadas, nos métodos "tearDown" e "tearDownAll" são utilizadas as anotações @AfterEach e @AfterAll respectivamente, com essa anotação o método tearDown é executado ao final de cada teste e o método "tearDownAll" ao final de todos os testes.

A classe "loginPage" é responsável por mapear o passo a passo dos testes, ou seja, onde a aplicação irá clicar, inserir informações na tela e verificar se alguma informação existe. Alguns comandos são utilizados, o Click() é responsável por clicar no caminho mapeado, já o sendKeys é responsável por inserir o valor passado na classe de teste. Por fim, temos a classe "Test", nela inicializamos os objetos do tipo "page", passamos os parâmetros que devem ser utilizados no momento de mapeá-los na classe "page" e executamos seus métodos com a anotação @Test do JUnit.

Á título de exemplo, a Listagem 1 e 2 ilustra parte de um teste funcional realizado para verificação de login. A partir deste exemplo podemos entender melhor o funcionamento das classes do tipo "page". Os casos de teste referenciados na Listagem 1 e Figura 3 são exemplos de como a ferramenta Selenium é utilizada para construção dos testes e interação com a página web.

```
1 package sigeduc.page;
2
3 import org.openqa.selenium.By;
4 import static core.DriverFactory.getDriver;
5
6 public class LoginPage {
7
8   public void acessarPagina(){
9        getDriver().get("https://treinamento-sigeduc-ba.esig.com.br/");
10   }
11  public void inserirEmail(String email){
getDriver().findElement(By.xpath("//input[@name='user.login']")).sendKeys(email);
12   }
13  public void inserirSenha(String senha){
getDriver().findElement(By.xpath("//input[@name='user.senha']")).sendKeys(senha);
14    }
15  public void clicarLogin(){
```

```
16        getDriver().findElement(By.xpath("//input[@type='submit']")).click();
17     }
18     public String messageErrorLogin(){
19         String message =
getDriver().findElement(By.xpath("//*[@id=\"painel-erros\"]/ul/li"));
20         return message;
21    }
22 }
```

Listagem 1. Código da classe LoginPage

fonte: Autoria própria

```
1 package sigeduc;
2
3 import core.BaseTest; import org.junit.jupiter.api.Assertions;
4 import org.junit.jupiter.api.Test;
5 import org.junit.jupiter.api.TestInstance;
6 import sigeduc.page.InicialPage;
7 import sigeduc.page.LoginPage;
8
9
10 @TestInstance(TestInstance.Lifecycle.PER_CLASS)
11 public class TestLogin extends BaseTest{
12     String messageInvalido = "Usuário e/ou senha inválidos";
13     String messageValido = "Escolha seu Vínculo para operar o sistema";
14     LoginPage loginPage = new LoginPage();
15    InicialPage inicialPage = new InicialPage();
16
17     @Test
18     public void loginInvalidotest() throws InterruptedException {
19         loginPage.acessarPagina();
20         loginPage.inserirEmail("admin");
21         loginPage.inserirSenha("senhaerrada");
22         loginPage.clicarLogin();
23       Assertions.assertEquals(messageInvalido,loginPage.messageErrorLogin());
24
25    }
26    @Test
27     public void loginValidoTest() throws InterruptedException {
28         loginPage.acessarPagina();
29         loginPage.inserirEmail("admin");
30         loginPage.inserirSenha("admin");
31         loginPage.clicarLogin();
32         Assertions.assertEquals(messageValido,loginPage.messageLogin());
33
34     }
35 }
```

Listagem 2. Código da classe TestLogin

fonte: Autoria Própria

Como foi dito anteriormente, a classe "page" é responsável por mapear o passo a passo dos testes, como podemos observar na Listagem 1. Para conseguir instruir o Selenium a realizar as ações que queremos é preciso conhecer o xPath do objeto DOM que se deseja acessar. O xPath funciona como uma rota para cada item da estrutura da página. Após a escrita do script na nossa classe LoginPage é necessário executar as informações na classe de teste. A classe de test ilustrada na Listagem 2 é onde nos comunicamos com nossa classe Page e através do JUnit podemos utilizar seus recursos para automatizar nossos casos de testes.

Ao executar os testes é aberto um navegador, ele acessa a página e faz o procedimento similar ao usuário com as informações passadas no parâmetro dos métodos da classe de "Test". Como podemos observar na Figura 3, a execução do teste com o Selenium e JUnit mostra importantes informações, podemos visualizar a página web acessada e a informação de login fornecida incorretamente, observamos também o tempo de execução do teste realizado, o nome do teste executado que nesse caso foi o loginInvalidotest e a validação do teste com a mensagem de "*Tests passed*".

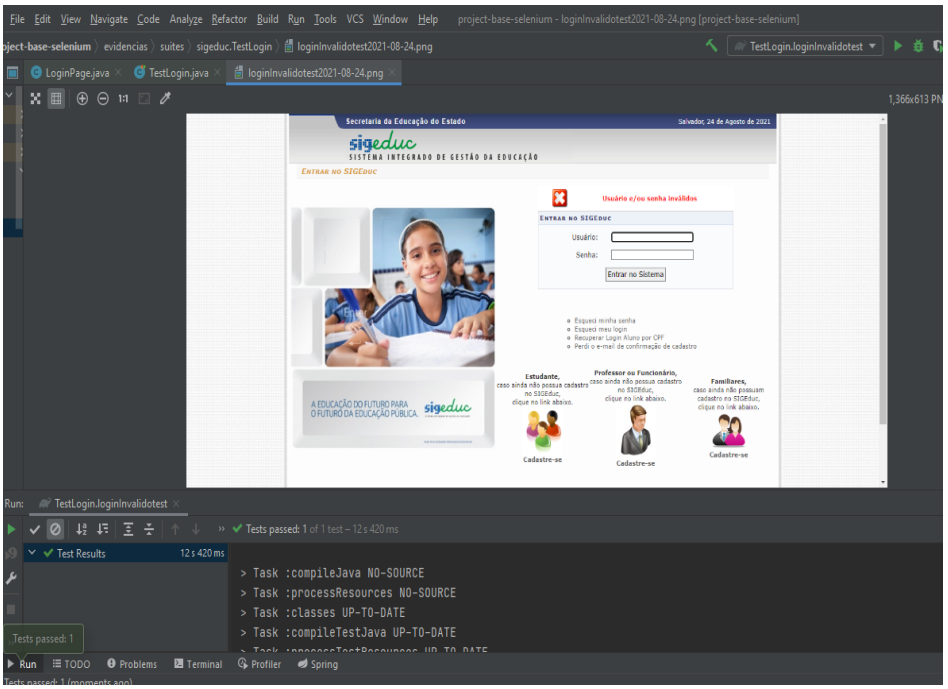

Figura 3. Execução de teste no Selenium

fonte: Autoria Própria

4.3 Validação do processo Automatizado de Testes

O Sigeduc é uma plataforma digital direcionada à gestão da educação básica. O sistema tem diversas funcionalidades, cobre tanto a área administrativa quanto a área educacional. Entre suas funcionalidades temos, o portal da gestão escolar, a administração técnica, o portal de matrícula entre outras funcionalidades.

Na Figura 4 podemos visualizar alguns módulos disponíveis, através deles o SIGEduc é a solução perfeita para sistematizar de forma digital os processos e fluxos de uma administração escolar, tanto pelos próprios administradores da escola como para uma secretaria de estado, dando autonomia para ambas gerenciar processos e esquematizar melhorias futuras, através de relatórios de administração, matrículas online, acompanhamento contínuo do estudante e outras funcionalidades.

Para realização da implementação dos testes automatizados, foram escolhidos inicialmente os módulos, central de matrícula, gestão curricular, diário de classe, gestão do estudante e unidades escolares, nesses módulos serão testados funcionalidades de

cadastrar, listar, alterar, remover , emissão de relatórios, gerenciamento de calendários, gerenciamento de estudantes e outros casos de usos.

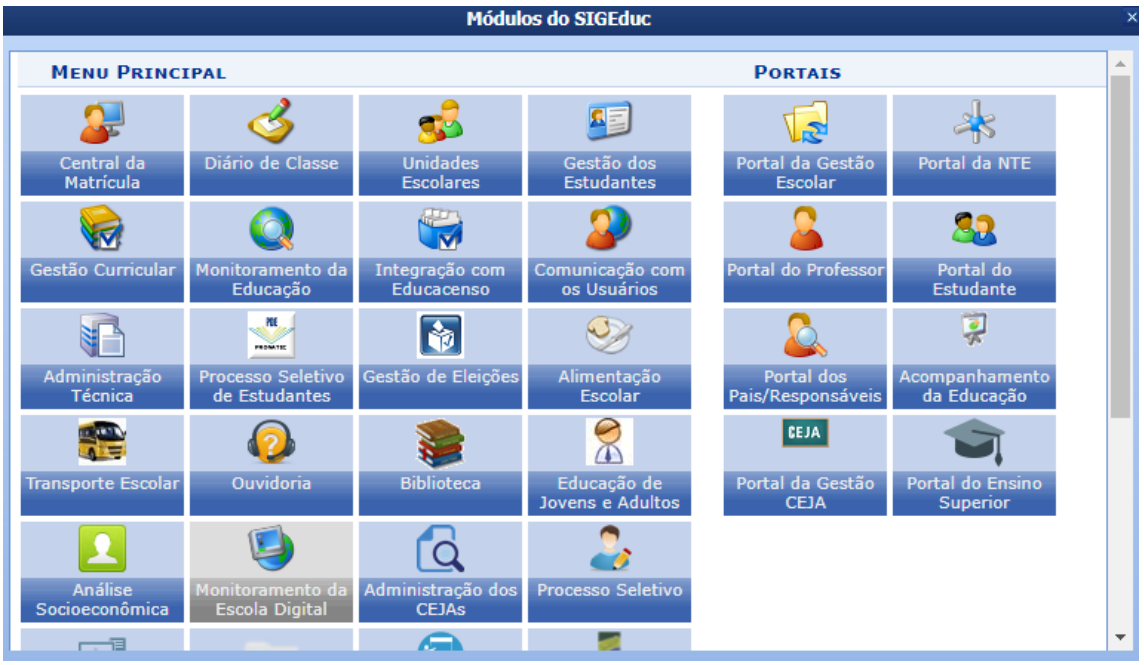

Figura 4. Módulos do SIGEDUC

fonte: https://sigeduc.rn.gov.br

Tabela 2 - Levantamento de funcionalidades por módulos e testes realizados

	Total de Funcionalidades por módulo	*Total de Funcionalidades testadas por módulo*	*Porcentagem de testes realizado por módulo*
Administração Técnica	53	10	18,8%
Gestão Curricular	28	10	35,71%
Central da Matrícula	41	10	24,39%
Diario de Classe	96	10	10,42%

Gestão do estudante	21	10	47,62%
TOTAL:	239	50	20,92%

O critério de escolha dos módulos e funcionalidades a serem testadas se deu ao analisar o seu constante uso, novas implementações e constantes mudanças, funcionalidades que frequentemente apresentam falhas em produção, justificando a escolha para automatização dos testes funcionais e facilitando testes de regressão.

Para isso, a estratégia passou pela construção de um documento de especificações de caso de teste[1], este documento contém os testes empregados na execução dos testes, a definição de todos os testes realizados, com suas pré e pós condições, passos específicos, critérios de sucesso, estratégia de testes, procedimentos e requisitos.

Como podemos ver na tabela 2, foram testadas cinquenta funcionalidades de um total de 239, divididos em cinco módulos. Com a automatização dos testes, é possível notar um ganho significativo no tempo de execução. Inicialmente, executar os cinquenta testes funcionais automatizados teve duração de oito minutos e quarenta segundos, enquanto realizar o mesmo processo de forma manual teve duração de quinze minutos e trinta e dois segundos, ou seja, a execução da forma manual leva quase o dobro do tempo da forma automatizada. A automatização é mais custosa no início devido a sua implementação inicial, mas, é possível um amplo reaproveitamento de código na implementação de novos cenários de testes, facilitando a manutenção de novos scripts, da maneira que o código está estruturado é possível realizar um novo cenário de teste em dez minutos. Portanto, a automatização é benéfica ao longo das execuções, obtendo um ganho de tempo considerável nas atividades de testes funcionais. Os testes de regressão que antes também levavam grande tempo e feito de forma manual, com a inclusão dos testes funcionais automatizados podem passar a ser feitos de forma automatizada.

5. Considerações Finais e Trabalhos Futuros

Este trabalho teve como objetivo melhorar o processo de desenvolvimento de *software* na empresa ESIG/Software, ao apresentar um processo de realização de testes automatizados.

Portanto, o trabalho contextualiza a importância do processo de teste no desenvolvimento do produto, apresenta a problemática de se trabalhar com testes manuais e o benefício de automatizar testes funcionais na esteira de desenvolvimento. Ainda em vistas a busca do objetivo, foi apresentado um estudo acerca da ferramenta escolhida para construção dos testes automatizados, a partir da análise da ferramenta e do contexto da empresa selecionar uma IDE e linguagem de programação que se

[1]https://docs.google.com/document/d/18vCDPoT-Vc36_vKzZDt-I5kISzg4NXd3/edit?usp=sharing&ouid=100106778617334366177&rtpof=true&sd=true

adequasse mais a tarefa, planejar e implementar um conjunto de casos de teste de referência.

Baseado no que foi apresentado, é possível compreender que testes de forma geral são importantes para o desenvolvimento de software. Notamos que a cultura de teste automatizado quando aplicada ao desenvolvimento traz grandes benefícios, podendo ser financeiro ou não. No entanto, inserir a automação de testes em meio ao processo de desenvolvimento pode ser algo complexo e custoso, por isso se faz necessário o conhecimento de boas práticas, ferramentas, regra de negócio do sistema em desenvolvimento e indícios de problemas.

Assim, pode-se concluir que o trabalho foi de grande importância visto que, foi apresentada uma forma que fermenta essa cultura de teste automatizado e traz a possibilidade de realização de testes funcionais e testes de regressão automatizado dentro do processo de desenvolvimento do software da empresa. É importante comentar que a partir de novos estudos tendo como base esse trabalho, é possível avaliar o impacto das implementações desses testes funcionais na produtividade de desenvolvimento, execução de casos de testes, comparação da velocidade dos testes realizados de forma manual para o automatizado, verificação de requisitos funcionais e não funcionais e na segurança de implementações de novas funcionalidades.

6. Referências Bibliográficas

SOMMERVILLE, Ian. Software engineering 9th Edition. ISBN-10, v. 137035152, p. 18, 2011.

INTHURN, Cândida. Qualidade & teste de software: engenharia de software; qualidade de software; qualidade de produtos de software; teste de software; formalização do processo de teste; aplicação prática dos testes. Visual Books, 2001.

BERNARDO, Paulo Cheque; KON, Fabio. A importância dos testes automatizados. Engenharia de Software Magazine, v. 1, n. 3, p. 54-57, 2008.

BARTIÉ, Alexandre. Garantia da qualidade de software. Gulf Professional Publishing, 2002.

MAGALHÃES, A.L.C. A Garantia da qualidade e o SQA: sujeito que ajuda e sujeito que atrapalha. ProQualiti – Qualidade na produção de software, v. 2, n.2, p. 9-14, 2006.

OLIVEIRA, Ângela. Testes de Software para reduzir custos em um projeto. BAGUETE. Disponível em: <http://www.baguete.com.br/artigos/856/angelaoliveira/23/06/2010/testes-de-software-para-reduzir-custos-em-um-projeto>. Acesso em: 10/11/2021

Peres, Paulo. Teste de Software: Os primeiros passos em busca da qualidade de software. Disponível em: <http://pt.slideshare.net/pauloperes2009/testes-desoftware-uma-viso-geral> Acesso em:10/11/2021