IAGHO CRISTIAN CHAGAS DE CARVALHO

IMPLEMENTAÇÃO DE TESTES DE SOFTWARE EM UMA EMPRESA PRIVADA DE TRANSPORTE

LAVRAS - MG

2023

IAGHO CRISTIAN CHAGAS DE CARVALHO

IMPLEMENTAÇÃO DE TESTES DE SOFTWARE EM UMA EMPRESA PRIVADA DE TRANSPORTE

Relatório Técnico de Estágio Supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para a obtenção do título de Bacharel.

Prof. DSc. Paula Christina Figueira Cardoso
Orientadora

LAVRAS - MG

2023

IAGHO CRISTIAN CHAGAS DE CARVALHO

IMPLEMENTAÇÃO DE TESTES DE SOFTWARE EM UMA EMPRESA PRIVADA DE TRANSPORTE

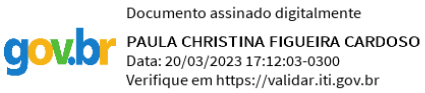
Relatório Técnico de Estágio Supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para a obtenção do título de Bacharel.

Aprovada em 17 de fevereiro de 2023

PAULA CHRISTINA FIGUEIRA CARDOSO - UFLA

PAULO AFONSO PARREIRA JUNIOR - UFLA

GABRIEL EDMILSON PINTO - UFLA



Prof. DSc. Paula Christina Figueira Cardoso

Orientadora

LAVRAS - MG

2023

Dedico especialmente a minha mãe Noemia que sempre me ajudou e me apoiou, não medindo esforços para que eu tenha o melhor mesmo com todas as dificuldades, aos meus avós e toda a família que sempre tiveram orgulho da pessoa que me tornei.

"Por vezes sentimos que aquilo que fazemos não é senão uma gota de água no mar. Mas o mar seria menor se lhe faltasse uma gota".

(Madre Teresa de Calcuta)

RESUMO

Este trabalho apresenta o relatório de estágio em implementação de testes na empresa Expresso Nepomuceno S/A. O estágio teve como objetivo a criação e implementação de testes, para melhorar a qualidade e a confiabilidade dos softwares que eram desenvolvidos pela empresa. Para a realização das atividades foram necessários conhecimentos sobre algumas tecnologias e metodologias importantes, como a linguagem de programação Java, conceitos sobre programação WEB, testes manuais, automatização de testes, JUnit, Selenium e outros temas relacionados. O relatório descreve os métodos de desenvolvimento, atividades, e as ferramentas utilizadas. As atividades que o estagiário desenvolveu promoveu o conhecimento de novas tecnologias, aplicação dos conhecimentos adquiridos no curso de graduação e trabalho em equipe para a geração de produto dentro de uma organização.

Palavras-Chave: *WEB, Selenium*, JUnit Java, Transporte Rodoviário

ABSTRACT

This work presents the internship report on test implementation at Expresso Nepomuceno S/A. The objective of the internship was to create and implement tests to improve the quality and reliability of the software presented by the company. In order to carry out the activities, knowledge about some important technologies and methodologies was necessary, such as the Java programming language, concepts about WEB programming, manual tests, test automation, JUnit, Selenium and other related topics. The report describes the development methods, activities and tools used. The activities that the intern developed promoted knowledge of new technologies, application of knowledge acquired in the evolution course and teamwork for the generation of products within an organization.

Keywords: WEB, Selenium, JUnit Java, Road Transport

Sumário

1 INTRODUÇÃO

Neste capítulo será apresentada uma contextualização sobre o setor rodoviário, sua ligação com a empresa onde o estágio ocorreu e as ferramentas envolvidas nas atividades. Também serão apresentados os objetivos do trabalho desenvolvido.

1.1 Contextualização

No Brasil, o setor rodoviário é o principal responsável pelo transporte de cargas entre as economias mundiais, contando com 1.720.700 km de extensão total da malha rodoviária. Essa malha rodoviária é utilizada para o escoamento de 75% da produção no país (G1, 2018), e que cresceu apenas 0,5% no período compreendido entre 2009 e 2019 (10 anos). No transporte de cargas existiam 219.965 empresas, 435 cooperativas e 724.098 autônomos registrados em 2019. O total de veículos autorizados para realizar transporte de cargas era de 2.270.861 (CNT, 2020).

Esses números indicam a importância do transporte de cargas no nosso país, pois é por este meio que é feito a maior parte do abastecimento da produção no país, levando-o desde o ponto de partida ao ponto de chegada.

1.1.1 A Empresa

A Expresso Nepomuceno é uma empresa de transporte rodoviário e logística que foi fundada no dia 13 de outubro de 1959, em Lavras (Minas Gerais), por Agnaldo de Souza (atual presidente de honra da companhia). Em sua trajetória, ao longo dos 61 anos de mercado, a empresa acumulou experiência e conquistou tanto parcerias como a confiança dos seus colaboradores. Atualmente a empresa possui 22 filiais e 27 pontos de apoio localizadas nos estados de Bahia, Espirito Santo, Goiás, Mato Grosso do Sul, Minas Gerais, Paraná, Pernambuco, Rio De Janeiro, Rio Grande do Sul, Santa Catarina, São Paulo e Tocantins (informações retiradas do site da empresa[1])

A empresa é referência no segmento automotivo, prestando serviços que vão desde o transporte da matéria prima até a entrega do produto final para as principais montadoras e autopeças do Brasil. No segmento florestal, a Expresso Nepomuceno presta serviços de transporte e movimentação desde a colheita até a entrega de celulose para os clientes, sendo reco-

[1] https://www.expressonepomuceno.com.br/

nhecida por atender todas as normas do setor e auditorias FSC[2].

Em 2010, a empresa iniciou suas operações no setor sucroenergético em todo o processo de CCT (corte, carregamento e transporte), se consolidando entre os maiores operadores deste setor, sendo responsável por toda a logística de abastecimento das usinas – adicionalmente, a Expresso Nepomuceno oferece serviços de manutenção de veículos e de estradas com o objetivo de aumentar a produtividade da operação.

Grandes indústrias do segmento químico e petroquímico utilizam, há mais de 20 anos, os serviços de transporte, armazenagem, *crossdoking*[3] da empresa na distribuição para os estados do sudeste.

No segmento de alimentos e bebidas, a Expresso Nepomuceno trabalha com operações de frota dedicadas em projetos personalizados de distribuição urbana e transferência entre fábricas e clientes. A empresa também opera nas principais rotas do agronegócio do país com veículos específicos e equipados com tecnologia embarcada no transporte de grãos e granéis sólidos.

A empresa também atende os diversos segmentos da indústria brasileira no abastecimento de matéria prima e entrega do produto final. Além disso, no segmento de locação, oferece contratos de locação de veículo e equipamentos personalizados de acordo com a necessidade do cliente, com serviços de manutenção em oficinas da própria empresa ou no cliente. Percebe-se que a empresa Expresso Nepomuceno oferece uma variedade de serviços, o que a torna relevante para a região, pois gera muitos empregos, além de toda a estrutura de transporte de cargas.

1.2 Objetivos

O objetivo principal desse trabalho é apresentar e detalhar as principais atividades realizadas durante o período de estágio na Expresso Nepomuceno, no período de 22 de outubro de 2020 à 22 de abril de 2021, apresentando os testes realizados e as dificuldades encontradas. Durante o estágio foram criados novos testes automatizados para o sistema assim aumentando a sua confiabilidade.

[2]Sigla em inglês para *Forest Stewardship Council*, ou Conselho de Manejo Florestal, em português.

[3]Sistema de distribuição no qual a mercadoria recebida em um armazém ou centro de distribuição para ser expedida para o consumidor final de forma imediata (redistribuição rápida).

1.3 Estrutura do Trabalho

Este relatório técnico está estruturado da seguinte forma: introdução com contextualização e objetivos, referencial teórico, estado da arte, metodologia, resultados, conclusão e referências bibliográficas.

2 REFERENCIAL TEÓRICO

Neste capítulo são descritos os conceitos básicos no qual se fundamentou o estágio e as atividades envolvidas.

2.1 Ferramentas

Nesta seção são descritas as ferramentas utilizadas durante o estágio.

2.1.1 Fluig

A empresa utiliza um sistema chamado Fluig (criado em 2013 pela empresa TOTVS), que centraliza e registra os dados organizacionais em um único local por meio da comunicação colaborativa, organizando os processos internos, ou seja, digitalizando processos do dia a dia. Com isso aumentando a sua produtividade e acelerando os resultados, somente transformando suas operações que eram manuais, em processos digitais e automáticos.

2.1.2 Selenium

Selenium[4] é um framework de código aberto usado para testar aplicações/sites de uma forma automatizada, disponível nos sistemas operacionais Windows, Mac e Linux, com suporte a várias linguagens de programação, tais como: Java, C# e Python.

As ferramentas que o Selenium contém são bem completas. Por exemplo, o Selenium IDE[5] que permite ao usuário a criação de scripts de testes de forma muito rápida, podendo executa-lo várias vezes.

O Selenium usa o próprio drive de navegação, o Selenium WebDriver[6], que é compatível com os navegadores mais atuais, como o Chrome, Firefox, Internet Explorer e Headless, sendo que cada browser tem seu próprio driver, permitndo a integração do script criado pelo desenvolvedor com o navegador em que se deseja executar a automação.

Existe também o Selenium Grid[7], que é voltado para a Clusterização[8], possibilitando o teste em diferentes navegadores e em diferentes máquinas de forma remota.

[4] https://www.selenium.dev/
[5] https://www.selenium.dev/selenium-ide/
[6] https://www.selenium.dev/documentation/webdriver/
[7] https://www.selenium.dev/documentation/grid/
[8] O termo clusterização se refere a uma arquitetura de sistema que conecta dois ou mais computadores como se fossem um só.

O framework traz bastante flexibilidade o que torna a implementação dos testes em qualquer software possível.

2.1.3 JUnit

JUnit também é uma framework de código aberto que ajuda no desenvolvimento de testes, que facilita a criação e manutenção de código para automação de teste apresentando também os resultados. É possível verificar se cada método da classe funciona conforme o esperado, apresentando possíveis erros ou falhas se existirem.

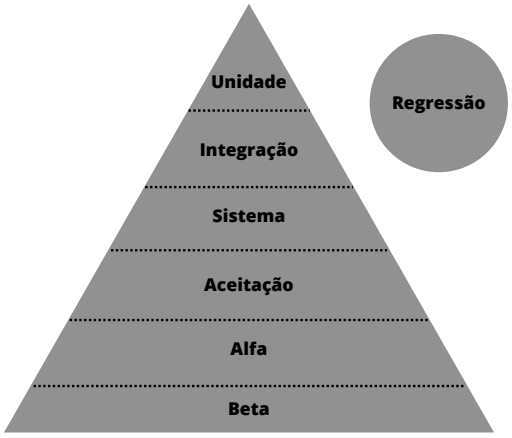
Usando JUnit, os programadores conseguem criar testes padrões, para que possam executar geralmente de forma automatizada, tudo isso com suporte a Java, que é uma das linguagens mais utilizada atualmente.

Esse framework que atualmente se encontra na versão 5.9.0[9], é constantemente atualizada pelo seus colaboradores adicionando novos recursos, correção de *bugs* e mais integrações.

2.2 Testes de Software

Os testes de software são atividades do processo de desenvolvimento de sistemas de software que visam executar o programa de forma sistemática para encontrar defeitos, minimizando a ocorrência de falhas e aumentando a sua confiabilidade (JAMIL et al., 2016). Essas atividades podem ser categorizadas em níveis diferentes, sendo os principais testes: de unidade, de integração, de sistema, de aceitação, alfa, beta e de regressão (Figura 2.1).

Figura 2.1 – Tipos de Teste de software (Autor)



[9]https://github.com/junit-team/junit5/releases. Acesso em 24 de agosto de 2022

Para os testes de unidade almeja-se testar a menor parte testável de um programa – que seriam funções, métodos ou classes. O ideal para esse tipo de teste é que cada unidade seja independente das demais e que também seja o primeiro teste a ser realizado no sistema e/ou software.

Os testes de integração iniciam a fase de validação da comunicação entre os componentes de um sistema, onde os módulos serão testados em grupo. Essa fase deve suceder a fase de testes de unidade.

Nos testes de sistema o programador faz a execução do sistema sob o ponto de vista do usuário final, varrendo as funcionalidades em busca de possíveis falhas em relação ao objetivo original.

Os testes de aceitação em muito se assemelham aos teste de sistema, porém dessa vez quem irá executá-los é, geralmente, o usuário final – possivelmente o cliente –, com a finalidade de verificar se o comportamento está de acordo com as especificações do software.

Na realização dos testes alfa também é testado o sistema por completo e de forma natural, ou seja, não planejada e com um pequeno grupo de usuários reportando falhas não detectadas até o momento. Usualmente, representantes da equipe de programadores acompanham de perto esse tipo de teste no intuito de observar e coletar erros e falhas a serem corrigidas.

Seguindo, na fase de testes beta também é feita uma execução no sistema de forma não planejada – sob o ponto de vista do usuário final –, porém por um grande número de pessoas e sem uma equipe de programadores para acompanhar essas falhas. Dessa forma, são os usuários quem reportam os erros e falhas encontradas.

Ao contrário dos testes acima, os testes de regressão podem ser realizados em qualquer momento do desenvolvimento, não havendo uma sequência pré-estabelecida. O objetivo então é, após alguma alteração no sistema, reexecutar testes para conferir se tudo continua funcionando corretamente afim de encontrar efeitos colaterais de um desenvolvimento com algum erro ou ação inesperada. Conceitos baseados na revista (Engenharia de Software Magazine, 2007)

2.3 Testes Automatizados de Software

Fazer testes de forma manual pode se tornar uma tarefa inviável de ser realizada, principalmente pela grande quantidade de tempo gasto para que um testador execute cada caminho possível de um software (CARVALHO, DE FREITAS, 2014). Devido a esse problema nasce-

ram os testes automatizados com o intuito de melhorar e facilitar todo esse processo.

Os testes automatizados podem ser definidos em um conjunto de ações que analisam os resultados de um possível caminho que o software tenha, ou seja, são criadas funções que irão testar as classes, métodos ou procedimentos e ao final dos testes é comparado os resultados esperados com os resultados obtidos por essas funções. Toda essa análise é realizada de forma automática, sendo necessária a criação do teste uma única vez e pode ser reutilizado em qualquer outra etapa do processo de desenvolvimento de software, verificando se o sistema continua em com todas as suas funcionalidades operando corretamente após alguma alteração no código.

Existem vários benefícios que os testes automatizados podem trazer, a redução de custos é um deles. Mesmo que no início o investimento para aplicação destes testes no sistema seja bem considerável, após a sua implementação o custo geral dos testes é menor comparado ao teste manual, tudo devido as otimizações que essas automações trazem. Podemos citar o estudo de caso de Chiavegato et al. (2013) que também foi usado como trabalhos relacionados deste documento para mostrar que é possível otimizar tempo e esforço com testes automatizados.

Um outro beneficio que também é valido de ser citado é o aumento da produtividade, pois como os testes automatizados são mais rápidos de serem executados, os programadores conseguem encontrar erros antes de se tornar algo mais grave, ou seja, conseguem realizar as correções preventivas, aumentando a produtividade graças a esse *feedback* ativo que o método produz.

Além disso a eficiência nas operações é notória se levarmos em consideração que está sendo utilizado uma máquina para executar os testes. Muitas vezes os testes manuais podem levar muito tempo para serem realizados, quando comparados em testes automatizados, que trazem agilidade das operações no time de Controle de Qualidade (QA)[10], principalmente em cenários mais complexos.

Robotium, Watir e Appium são frameworks para automatização de testes, porém o mais conhecido e utilizado atualmente é o Selenium, com cerca de 180 mil usuários somente em sua página do GitHub (SELENIUMHQ, 2023), suportando os testes de unidade, integração e regressão.

[10]Em inglês Quality Assurance

2.4 Conceitos de AdvPL

A *Advanced Protheus Language (AdvPL)* é uma linguagem de programação orientada a objetos e a eventos, baseada em dBase que foi um dos primeiros sistemas de gerenciamento de banco de dados para microcomputadores. Protheus assim como o AdvPL é de propriedade da TOTVS e é o lider do mercado brasileiro quando o assunto é sistema ERP[11], onde tem como objetivo o controle de toda a gestão de processos e informações permitindo a integração de todos os setores de uma empresa. As vantagens de usar um sistema ERP é a melhoria da produtividade geral da instituição, disponibilizando as informações que são gerenciadas de forma rápida e segura para a tomada de decisão, além do ganho de qualidade nos processos.

2.5 A importância dos Testes

O Selenium vem se tornando uma ferramenta bastante popular no mercado de testes automatizados, pois seguindo um padrão de capturar e reproduzir, fornece uma reprodução visual do teste sendo executado. Vale a pena mencionar também todo o suporte e documentação oferecidos em seu site oficial, além de ser uma ferramenta de código aberto onde a comunidade constantemente desenvolve, questiona e responde os outros desenvolvedores em diferentes funcionalidades e casos de uso da ferramenta (ALVES, 2019).

Os testes de software são muito importantes para a área de desenvolvimento. Segundo Bernardo (2008), a tarefa de desenvolver softwares é muito complexa, exigindo organização, atenção e muita comunicação. Para melhorar o trabalho dos desenvolvedores, os testes criam a possibilidade de fazer o uso de situações específicas para serem avaliadas, incontáveis vezes, garantindo o menor índice possível de falhas humanas e também facilitando a identificação de comportamentos de software não desejados.

As práticas mais utilizadas na automação de testes são explicada no artigo (CAETANO, 2008) publicado na revista Engenharia de Software Magazine, 5ª edição, na qual faz observações importantes como encarar a automação de testes como um projeto novo exigindo um planejamento detalhado. Também se fez menção que é recomendado a automatização dos testes criticos primeiro, além de considerar dimensionar a infra-estrutura adequadamente pensando que a automatização dos testes é um investimento a longo prazo.

[11] Enterprise resource planning. Em tradução livre: Sistema integrado de gestão empresarial

2.6 Trabalhos Relacionados

Foi analisado um estudo de caso por Chiavegato et al. (2013), onde se usava a automatização em um sistema para soluções hospitalares. O sistema em questão era grande e complexo, o que gerava bastante esforço para toda a equipe. Decidiram então melhorar dois módulos, dos nove que existiam, e para isso foram realizados vários testes automatizados utilizando BDD (*Behaviour Driven Development*) – em tradução livre: desenvolvimento orientado a comportamentos –, que é uma das técnicas ágeis[12] para desenvolvimento de software cujo objetivo é minimizar as falhas de comunicação entre o cliente e a equipe de desenvolvimento, onde o desenvolvedor escreve as especificações explicando e descrevendo o comportamento que a funcionalidade deve possuir.

A ferramenta JBehave é comumente empregada para esse tipo de desenvolvimento, e foi aplicada no artigo em conjunto com o Selenium. Podemos assemelhar os objetivos, os resultados que foram destacados e o uso da ferramenta Selenium no artigo com este relatório, porém no artigo o sistema para qual foi feito os testes era um sistema ligado a área da saúde, e este trabalho é na área do transporte. Uma outra coisa que podemos também diferenciar é que no artigo foi utilizado para realizar os testes o BDD, o que não foi utilizado durante o estágio.

Almeida *et al*., (2019) escreve um artigo mostrando frameworks como o JUNIT, HTTPUNIT e JWEBUNIT para a automatização de testes, onde apresenta o que são essas ferramentas: fazendo anotações, funcionalidades, a aplicação deles com códigos, exemplos de saídas e tutoriais, levando ao leitor um entendimento maior sobre as ferramentas em questão. Este artigo o autor utilizou de frameworks que não foram utilizados durante o estágio, com exceção do JUNIT na qual foi a base para todo o desenvolvimento.

[12]Técnicas ágeis são um método de acelerar entregas de um determinado projeto. Dividindo as entregas em ciclos menores para a revisão de planejamentos e a maior agilidade em corrigir eventuais erros

3 ATIVIDADES REALIZADAS

Nem todos os procedimentos, processos e testes realizados no estágio poderão ser detalhados porque correspondem a dados reais dos clientes da empresa e, portanto, sigilosos, além do produto intelectual também. Todavia, será apresentada uma visão adaptada de forma a descrever bem as atividades realizadas durante o período de realização do estágio sem comprometer o sigilo das informações mais sensíveis. O estágio teve duração de 6 meses, com carga horária de 6 horas por dia.

3.1 Período de treinamento

No início do estágio, a área de TI da empresa estava se desenvolvendo e ainda não havia uma equipe para testes automatizados, o que colocava a implementação dos testes em uma fase inicial. Em decorrência disso, a empresa ofereceu cursos na área de testes na plataforma Alura [13], com uma duração de aproximadamente um mês, além de exercícios práticos para fixação de conteúdo.

3.2 Teste Automatizado: Despesas e Receitas

Logo após a finalização dos cursos foi iniciado o desenvolvimento do primeiro teste automatizado somente verificando a confiabilidade dos dados que eram passados em um processo dentro da plataforma Fluig, onde eram feitas as solicitações de despesas e receitas que a empresa tinha. O testes então, tinha como objetivo navegar até a plataforma, entrar no processo de solicitação, criar uma solicitação, passar por uma auditoria (um usuário com mais permissões), que iria verificar se os dados estavam corretos e só assim aceitar a solicitação. Os diagramas que ilustram o fluxo deste processo de Solicitação de Despesa e Receita pode ser visto na figura 4.1; o fluxo do processo de auditoria pode ser visto na figura 4.2, ambas no final do documento.

3.2.1 Funções iniciais

Inicialmente foi necessário a criação da classe contendo todo o código que era designado para o teste em questão. Algumas bibliotecas do JUnit, Selenium e Java foram importadas para utilizações de suas funções. Todo esse teste foi executado em um ambiente de testes. Exemplos de importações:

[13] https://www.alura.com.br/

Listing 1 – Importando Bibliotecas

```
1  import org.junit.Assert;
2  import org.junit.Test;
3  import org.openqa.selenium.By;
4  import org.openqa.selenium.WebDriver;
5  import org.openqa.selenium.chrome.ChromeDriver;
6  import org.openqa.selenium.support.ui.ExpectedConditions;
7  import org.openqa.selenium.support.ui.Select;
8  import org.openqa.selenium.support.ui.WebDriverWait;
9  import java.text.SimpleDateFormat;
10 import java.util.Date;
11 import java.util.List;
```

Após a importação, foram desenvolvidas as funções relacionadas ao teste. A primeira dela é a que iniciaria o *driver*, deixando-o aberto para uso. Seu objetivo foi instanciar e retornar um objeto da classe *WebDriver* da biblioteca do Selenium utilizando o construtor para o Chrome – navegador utilizado em todos os testes. Porém foi preciso colocar um *wait* para que o sistema pudesse esperar a janela do Chrome ser aberta antes de continuar com o teste.

Listing 2 – Iniciar driver

```
1      public WebDriver iniciarDriver(){
2        WebDriver driver = new ChromeDriver();
3        driver.manage().timeouts().implicitlyWait(10, TimeUnit.
             SECONDS);
4        return driver;
5      }
```

Depois disso foi criada uma função para realizar autenticação, a *efetuarLogin*, que também seria utilizada outras vezes. Seu objetivo era navegar até o site do Fluig passando como parâmetro o driver criado na função anterior.

Listing 3 – Efetuar Login (acessando site)

```
1      public void efetuarLogin(WebDriver driver){
2        //Link fantasioso
3        String linkFluig = "http://fluig.com";
4        driver.navigate().to(linkFluig);
```

```
5       ...
6   }
```

Após navegar até a página de login no site do Fluig é necessário realizar a autenticação com um usuário e senha padrão (não como um super usuário, como um Supervisor por exemplo), então estes parâmetros também foram passados no escopo da função. O Selenium utiliza de alguns meios para encontrar elementos na página atual, alguns desses meios é por localização no HTML da página. Assim, bastou procurar pelos elementos que continuam o nome "username" e "password" no HTML usando o método "By.name()" e enviar as chaves para os campos em questão usando o método sendKeys().

Listing 4 – Efetuar Login (Encontrando elementos)

```
1   public void efetuarLogin(WebDriver driver,String usuario,
        String senha){
2       ...
3       driver.findElement(By.name("username")).sendKeys(usuario);
4       driver.findElement(By.name("password")).sendKeys(senha);
5       ...
6   }
```

Continuando, seria necessário clicar no botão "Login" do site, mas dessa vez o botão não continha um nome, então foi preciso procurar pelo texto dentro do elemento usando o método "by.xpath()".

Listing 5 – Efetuar Login (Botão Login)

```
1   public void efetuarLogin(WebDriver driver){
2       ...
3       driver.findElement(By.xpath("//button[@class = 'btn btn-
            gray login-button']")).click();
4   }
```

Uma caixa de informações sempre abria quando o login era realizado com sucesso e precisava ser fechada de alguma forma. Para isso, foi criada outra função chamada "fecharInformativo" na qual era necessário esperar o carregamento das informações e, após isso, utilizar o método "wait.until" passando como parâmetro um outro método que é "presenceOfElementLocated" que também usaria o "xpath" para localizar o elemento "close" responsável por fechar o

informativo mediante um clique. Assim, após encontrar este elemento de fechamento usou-se o método “click()” para fechar a caixa de informações.

Listing 6 – Fechar Informativo

```
1    public void fecharInformativo(WebDriver driver,
        WebDriverWait wait){
2        wait.until(ExpectedConditions.presenceOfElementLocated(
            By.xpath("//button[@class = 'close']")));
3        driver.findElement(By.xpath("//button[@class = 'close']"
            )).click();
4    }
```

Para este primeiro teste foram utilizadas credenciais sempre válidas, i. e. o teste sempre passava pela autenticação e acessava a página inicial na plataforma. Quando a função de login descrita acima era concluída e a caixa de informações fechada, passava a ser necessário identificar o elemento na página que redirecionava, através de um clique, para o processo de solicitação de despesa e receita. Para tal foi criada a função “solicitacaoDespesaEReceita” que identificava e emitia um clique no elemento especificado.

Listing 7 – Solicitação Despesa e Receita

```
1    public void solicitacaoDespesaEReceita(WebDriver driver){
2        driver.findElement(By.xpath("//h3/span[text()='
            Solicitacao de Despesa e Receita']")).click();
3        driver.close();
4    }
```

Neste ponto, o teste já teria o acesso do formulário necessário, seria necessário agora preenchê-lo. O primeiro campo é o “tipo de solicitação” na qual dispunham-se as opções “PAGAR” e “RECEBER”.

Listing 8 – Campo Tipo Solicitação”

```
1    Select tipo = new Select(driver.findElement(By.name("
        TIPOSOLICITACAO")));
2    tipo.selectByValue("PAGAR");
```

O próximo campo era a “empresa”, foi escolhido a opção “EXPRESSO”

Listing 9 – campo ”Empresa”

```
1    Select empresa = new Select(driver.findElement(By.name("
        EMPRESA")));
2    empresa.selectByValue("EXPRESSO");
```

Logo após a inserção da empresa, o formulário consultava o banco de dados para ver quais são as filiais dessa empresa escolhida, então para preencher esse campo foi necessário definir um tempo de espera (*sleep*) para que o sistema concluísse a consulta. Este tempo foi definido ao executar o teste várias vezes e obtemos o maior valor entre eles. Depois das filiais serem carregadas, o teste então selecionava a primeira (01) no formulário.

Listing 10 – Campo ”Filial”

```
1    Thread.sleep(1500);
2
3    WebElement filial = driver.findElement(By.name("FILIAL"));
4    filial.sendKeys("01");
```

O próximo campo era um campo de “data de emissão”. Como o campo era preenchido automaticamente, verificava-se apenas se o campo não estava vazio.

Listing 11 – Campo ”Data de Emissão”

```
1    WebElement dataEmissao = driver.findElement(By.name("
        DATAEMISSAO"));
2    String datacampo = dataEmissao.getText();
3    if (!datacampo){
4        throw new Exception();
5    }
```

o campo “Número do documento” era o próximo, seguido do “tipo de título”:

Listing 12 – Campo ”Numero do Documento”

```
1    WebElement numeroDocumento = driver.findElement(By.id("
        NUMDOC"));
2    numeroDocumento.sendKeys("1234");
3    Select tipoTitulo = new Select(driver.findElement(By.id("
        TPTIT")));
4    tipoTitulo.selectByValue("NOTA CREDITO CLIENTE");
```

Do mesmo modo que os anteriores, era necessário preencher o “nome do cliente”, e após selecionarmos qual era o cliente o próprio formulário já preenchia os campos de “loja” e “nome do fornecedor”:

Listing 13 – Campo ”Nome do Cliente”

```
1    WebElement nomeCliente = driver.findElement(By.id("NOMECLIE"
        ));
2    nomeCliente.sendKeys("Cliente 1");
3    nomeCliente.sendKeys(Keys.ENTER);
```

Os próximos campos eram os campos de “Valor”, e a “natureza da receita/despesa”:

Listing 14 – Campos ”Valor”e ”NATUREZA”

```
1    WebElement valor = driver.findElement(By.id("E1VALOR"));
2    valor.sendKeys("321,00", Keys.ENTER);
3    WebElement natureza = driver.findElement(By.id("NATUREZA"));
4    natureza.sendKeys("ALUGUEL MAQUINA");
```

Os últimos campos a serem preenchidos eram rateio, “centro de custo” e “histórico”, e foram preenchidos da mesma maneira:

Listing 15 – Campos ”Rateio”, ”Centro Custo e ”Histórico”

```
1    Select rateio = new Select(driver.findElement(By.id("RATEIO"
        )));
2    rateio.selectByValue("NAO");
3    WebElement centroCusto = driver.findElement(By.id("
        CENTROCUSTO"));
4    centroCusto.sendKeys("00001000-CONTABILIDADE");
5    WebElement historico = driver.findElement(By.id("HIST"));
6    historico.sendKeys("TESTE");
```

Após todos os campos terem sido preenchidos corretamente, o próximo passo era enviar esse formulário clicando no botão de enviar. Também era necessário guardar, por enquanto em uma variável, o número da solicitação que acabou de ser gerada.

Listing 16 – Enviar formulário

```
1     driver.findElement(By.id("ENVIAR")).click();
2     String numeroSolicitacao = driver.findElement(By.xpath("//
         div/span/a")).getText();
```

Neste ponto é gerado um número da solicitação que é salva em uma variável para ser utilizada posteriormente.

3.2.2 Auditoria do formulário

Depois da solicitação ter sido gerada e obtido o seu número, era preciso realizar *logout* do sistema para autenticar um outro usuário de permissão superior ao que criou a solicitação, assim simulando uma auditoria.

Para proceder com o *logout* também era necessário apagar os *cookies*[14], já que o sistema continha um sistema de guardar as informações do usuário mesmo quando a página era fechada.

Listing 17 – Deslogar

```
1     driver.findElement(By.xpath("//i[@title='Home']")).click();
2     driver.findElement(By.xpath("//*[@id=\"user\"]/button")).
         click();
3     driver.findElement(By.xpath("//*[@id=\"user\"]/button/[@id="
         Sair\"]")).click();
4     driver.manage().deleteAllCookies();
```

Agora que a solicitação já foi criada e não há nenhum usuário logado, volta-se ao estado de aplicação onde se repete todas as funções necessárias para que um usuário acesse a página inicial da plataforma Fluig (autenticação e fechamento da caixa de informações).

Novamente na página inicial, agora com um usuário supervisor, o teste automatizado deve acessar a Central de Tarefas, área na plataforma Fluig reservada para o gerenciamento de atividades vinculadas a processos envolvendo o usuário autenticado. A central é acessada por meio da busca pelo botão com nome de classe e a solicitação é encontrada por meio do seu número (reservado anteriormente).

Listing 18 – Encontrando a solicitação

[14] Cookies são pequenos arquivos criados por sites visitados e salvos no computador do usuário pelo navegador. Esses arquivos contêm informações usadas para identificar visitantes, personalizar páginas de acordo com perfis ou facilitar a transferência de dados entre páginas de um mesmo site.

```
1    public void centraldeTarefasESolicitacao(WebDriver driver,
        String numeroSolicitacao){
2        driver.findElement(By.xpath("//button[@class = 'Central
            de Tarefas']")).click();
3        driver.findElement(By.xpath("//td[@title ="+
            numeroSolicitacao + "]")).click();
4    }
```

Com a recém-criada solicitação aberta no perfil de supervisor, inicia-se um procedimento de checagem dos dados para verificar se estão todos corretos. Este procedimento é executado pela função *verificaDados()* que vai analisar cada um dos dados inseridos no formulário em cada um dos campos, são eles: “TIPOSOLICITACAO”, “EMPRESA”, “FILIAL”, “DATAEMISSAO”, “NUMDOC”, “TPTIT”, “NOMECLIE”, “E1VALOR”, “NATUREZA”, “RATEIO”, “CENTROCUSTO”, “HIST”. Caso tudo esteja de acordo com os valores utilizados para o preenchimento do formulário, a aprovação pode ser feita.

Listing 19 – Verifica dados

```
1    public void verificaDados(WebDriver driver){
2        WebElement tipo = driver.findElement(By.name("
            TIPOSOLICITACAO")).getText();
3        if (tipo != "PAGAR")  throw new Exception();
4
5        WebElement empresa = driver.findElement(By.name("EMPRESA
            ")).getText();
6        if (empresa != "EXPRESSO") throw new Exception();
7
8        /* ...
9        Checagem de todos os campos
10       ... */
11
12       WebElement historico = driver.findElement(By.id("HIST"))
            .getText();
13       if (historico != "TESTE") throw new Exception();
14
```

```
15          driver.findElement(By.xpath("//a[text() = 'Aprovar']")).
                click();
16      }
```

Após aprovada a solicitação é gerado um número de aprovação dessa auditoria, e iremos testá-lo para saber se esse numero de auditoria existe, no caso, se ele é diferente de nulo:

Listing 20 – Checando Auditoria

```
1       String numeroAuditoria = driver.findElement(By.xpath("//div/
            span/a")).getText();
2       if (!numeroAuditoria){
3           throw new Exception();
4       }
```

Se após essa verificação o programa não ter lançado nenhuma exceção, então todo o nosso teste automatizado está completo, ou seja, tudo está funcionando como esperado.

Todo este teste automatizado foi desenvolvido para testar o formulário quando estamos simulando uma despesa, e o mesmo pode ser utilizado para simularmos uma receita, alteramos os valores que enviamos para o formulário, por exemplo o campo “tipo de solicitação”, na qual continha as opções “PAGAR” e “RECEBER”.

3.2.3 Efeitos do teste implantado

A implantação de um teste automatizado em uma parte do sistema tão crucial – e de uso em larga escala – tornou-se uma ferramenta de grande auxílio para garantir a confiabilidade dos dados. Além disso, o teste implantado também trouxe para a empresa uma possibilidade de rápida percepção de erros no fluxo do código-fonte do processo caso algo definido pelos desenvolvedores viesse a não ser satisfeito. Porém esses testes não estavam agendados para ser executado regularmente, visto que o código fonte mudava poucas vezes (devido ao pequeno número de desenvolvedores).

Logo após a finalização destes dois testes iniciais (de despesas e de receitas), a empresa enfrentou um gargalo na qual os desenvolvedores estavam sobrecarregados pelas solicitações, devido ao grande número de pedidos para poucos profissionais disponíveis. Como forma de atenuar este empecilho, o estagiário foi transferido para atuar na área de desenvolvimento em AdvPL (*Advanced Protheus Language*) utilizando os conhecimentos técnicos de programação

e automação desenvolvidos na área de testes automatizados.

3.3 Atividades desenvolvidas com AdvPL

Nesta sub seção será apresentada um breve conceito sobre uma atividade segundaria do estagiário dentro da empresa, envolvendo AdvPl.

Para iniciar o desenvolvimento na área de AdvPL o estagiário precisou passar novamente por um período de treinamento, no qual aprendeu sobre a linguagem e seus conceitos. E após o treinamento, o estagiário era designado a resolver requisições reais de dentro da empresa, como mudar um fluxo dentro do sistema Protheus (um sistema muito similar com o Fluig, abrangindo os principais processos operacionais das empresas).

3.3.1 Desenvolvimento

Entre as requisições em que o estagiário trabalhava, algumas delas eram relacionadas a mudança dos campos em tabelas do banco de dados e mudança de fluxos nos processos existentes.

Após um grande número dessas requisições, os desenvolvedores perceberam que existiam demasiadas solicitações para alterar e adicionar os mesmos campos em tabelas diferentes, portanto era preciso automatizar esse processo, criando um sistema para os solicitantes colocarem as informações que queriam que fossem alteradas e adicionadas, enviadas para um auditor que tinha apenas que checar se havia algum problema com essa tal mudança e aplicá-la.

Foi criado então um sistema que consistia em uma página na qual o solicitante inseria o código da tabela e era-lhe mostrado algumas informações sobre a tabela em questão, após isso era preciso inserir o campo da tabela na qual queriam alterar, caso existisse este campo as informações sobre ele também eram mostradas, caso fosse um campo a ser adicionado não retornava nenhuma informação.

Após o campo da tabela ser carregado o solicitante poderia escolher alterar: o nome do campo, a descrição, o valor padrão, a sigla, o tipo do campo (booleano, vetor de char, inteiro), se o campo deveria ou não aparecer para o usuário (ou se era somente um campo de controle). Dependendo da escolha do tipo do campo algumas informações apareciam para serem preenchidas, como o tamanho do vetor de char, por exemplo.

Quando fosse finalizado o preenchimento de todos os campos no qual queiram ser alterados e adicionados, bastava salvar as informações, e era mostrado uma lista de todos os campos

que o usuário fez a solicitação.

Na visão do auditor por sua vez, só tinha que analisar o campo em questão e avaliar se teria algum problema com essa tal mudança, caso estivesse tudo de acordo com a alteração, ele só precisaria aprovar a solicitação, o sistema se encarregava de criar um código que ia na tabela e alterar os valores automaticamente, sem a necessidade do estagiário criar um código especifico somente com essa mudança.

4 RESULTADOS

A implementação dos testes automatizados foi de grande importância para a evolução da qualidade do projeto. Durante a fase de testes foram levantadas várias sugestões para o melhoramento dos elementos, para maior facilidade de encontrar esses objetos na página. Também houve sugestões sobre a ordenação dos campos que poderiam ser melhoradas, e também foi encontrado alguns pequenos erros durante o preenchimento dos formulários. As sugestões baseadas nos testes não foram aplicadas imediatamente, porém entrou como uma tarefa a ser realizada posteriormente pelos desenvolvedores.

Também houve uma diminuição significativa no número de solicitações que eram criadas pelos requisitantes após a implementação do sistema que automatizava as alterações nos campos das tabelas; pode-se citar também a redução da falha humana ao criar um código com algum *bug* ou erro para essa alteração do campo, aumentando a produtividade, dado que os desenvolvedores podiam se encarregar de encarregar de outras solicitações.

Durante o estágio, diversas técnicas e ferramentas foram estudadas e aplicadas na construção de testes automatizados, culminando no teste automatizado de solicitações de receitas e despesas (Figura 4.1), complementado pelo teste automatizado de auditoria das solicitações (Figura 4.2). Levando em consideração o processo manual de testagem, seria preciso navegar página a página, preencher campo por campo, isso é muito dispendioso e retarda processos de desenvolvimento. Os testes apresentados neste trabalho agilizaram o processo de desenvolvimento dos processos e facilitaram a correção de *bugs*.

4.1 Desafios Encontrados

O estágio ofereceu muitas oportunidades que vieram acompanhadas de grandes desafios, o primeiro deles foi estabelecer um ritmo de trabalho diferente da rotina acadêmica, mesmo na forma de estudar ferramentas, tudo é estritamente voltado para a prática.

Após o período dos treinamentos, o maior desafio foi quando o estagiário começou a receber as solicitações dos outros funcionários dentro da empresa, com demandas e necessidades reais. O que era pedido e desenvolvido pelo estagiário seria usado por grande parte da empresa, implicando em uma pressão muito maior na tarefa. Uma oportunidade parecida com esta tinha acontecido durante a disciplina Programação WEB (GAC116, 2018), onde os alunos receberam uma solicitação do professor onde experimentaria a relação de empresa-cliente, simulando

todas as etapas para a construção de um sistema real.

4.2 Trabalhos Futuros

Algumas melhorias podem ser mencionadas no código de teste que foi desenvolvido, como por exemplo: as variáveis que foram testadas poderiam ser separadas em uma única parte do código, assim facilitando a manutenção e a alteração dos campos do formulário.

Outra otimização para se levar em consideração é uma melhoria a ser feita no código do site em si, melhorando os identificadores dos botões, dos links, das páginas e dos campos do formulário, assim facilitaria ao desenvolvedor encontrar o elemento requerido dentro do código. O que atualmente está desorganizado, com campos faltando identificadores, sendo necessário procurá-los pelo caminho absoluto.

Além das otimizações, a continuação dos testes em outras partes do sistema (como por exemplo a exclusão das solicitações de despesa e receita, a gestão dos pontos por geolocalização, inclusão e demissão de funcionários, gestão de contratos, etc...) seria de suma importância para continuar garantido a qualidade e consistência dos dados no sistema por completo.

Figura 4.1 – Diagrama de Fluxo de Teste Automatizado: Solicitação de Receitas e Despesas.

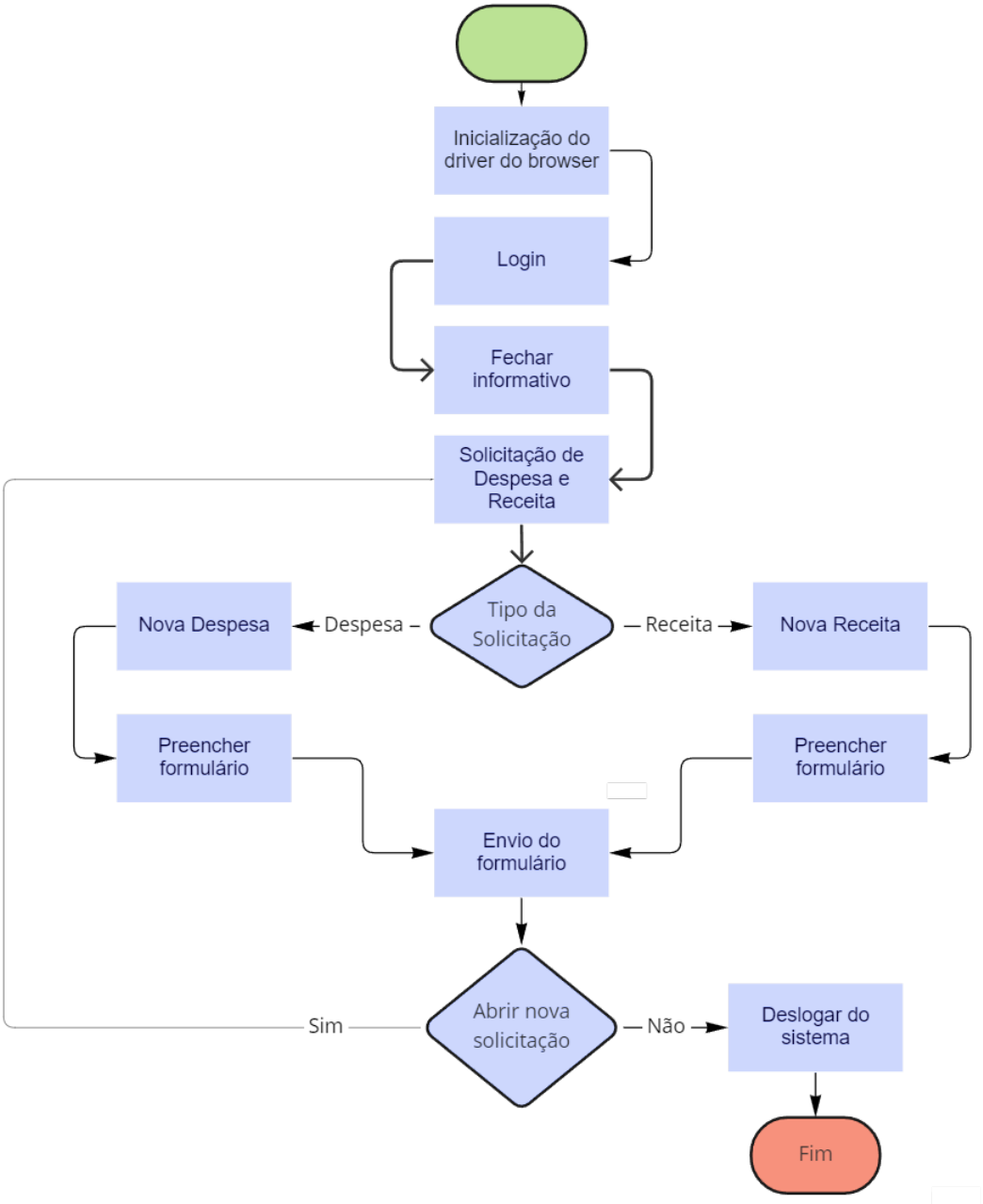
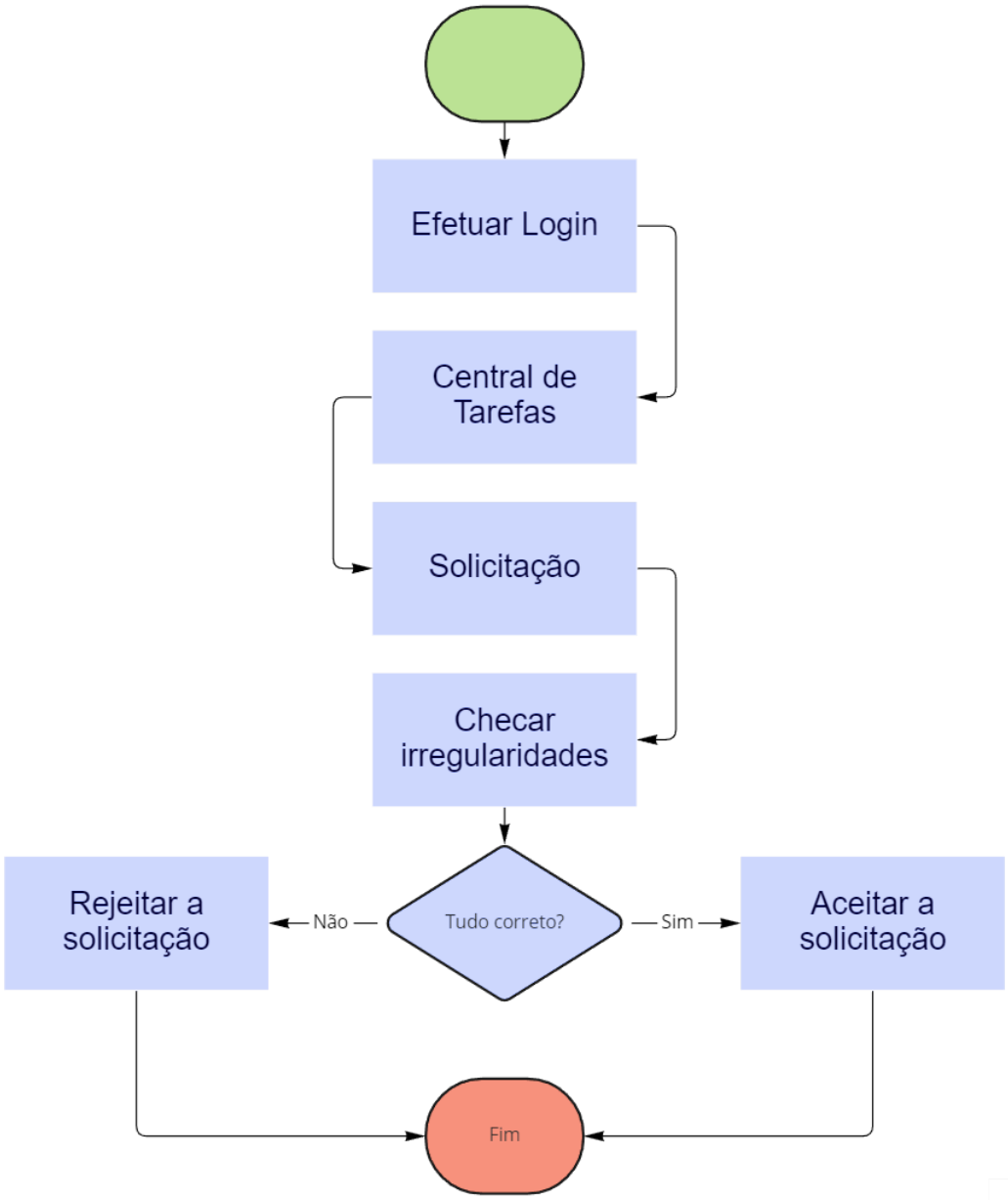


Figura 4.2 – Diagrama de Fluxo de Teste Automatizado: Auditoria.



5 CONCLUSÕES

O estágio realizado na Expresso Nepomuceno, como Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Ciência da Computação proporcionou a aplicação dos conhecimentos adquiridos no período acadêmico. Todas as etapas do estágio proporcionaram o desenvolvimento da capacidade de pensar logicamente para solucionar problemas com código.

Toda essa oportunidade e pressão de trabalhar com casos reais dentro da empresa contribuiu muito para a evolução da habilidade profissional do estagiário, onde agora ele consegue analisar a situação da visão e a necessidade do usuário, assim fazendo-o pensar em possíveis erros de interpretação que antes eram voltados somente para a visão de um desenvolvedor/programador, além de estar mais preparado para atuar no mercado de trabalho.

REFERÊNCIAS

CNT. **Anuário CNT do Transporte:** Transporte de cargas. Anuário CNT do Transporte: Estatísticas Consolidadas. Disponível em: https://anuariodotransporte.cnt.org.br/2020/Inicial. Acesso em 21 jan. 2021

M. A. Jamil et al. Software Testing Techniques: A Literature Review. **6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)**, Jakarta, Indonesia, 2016, pp. 177-182, doi: 10.1109/ICT4M.2016.045.

CARVALHO, Thyago Peres; DE FREITAS, Joslaine C. Jeske. Análise de ferramentas de teste automatizado de software. **Proc. Universidade Federal de Goiás**, 2014.

ALVES, Arthur Moreira. **Comparativo de ferramentas de automatização de testes: Selenium IDE e Selenium WebDriver**. 2019.

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

CHIAVEGATTO, Rafael B. et al. Desenvolvimento Orientado a Comportamento com Testes Automatizados utilizando JBehave e Selenium. In: **Anais do Encontro Regional de Computação e Sistemas de Informação Manaus**. 2013.

CAETANO, Cristiano. Melhores Práticas na Automação de Testes. **Revista Engenharia de Software Magazine, 5ª edição. p42-47**, 2008.

ALMEIDA, Igor Matheus Branco de et al. TESTES DE SOFTWARE JUNIT, HTTPUNIT E JWEBUNIT: FERRAMENTAS DE AUTOMATIZAÇÃO DE TESTES. **PEAD No 17 (2019): Nº XVII PESQUISA & EDUCAÇÃO A DISTÂNCIA** n. 17, 2019.

Por que o Brasil depende tanto do transporte rodoviário? G1. mai. 2018. Disponível em: <g1.globo.com/economia/noticia/por-que-o-brasil-depende-tanto-do-transporte-rodoviario.ghtml>. Acesso em 21 mar. 2022.

Conheça 5 benefícios dos testes automatizados de software Testing Company. mai. 2021. Disponivel em: https://testingcompany.com.br/blog/conheca-5-beneficios-dos-testes-automatizados-de-software. Acesso em 22/03/2022

Selenium 2023. Disponivel em: <https://github.com/seleniumhq/selenium>. Acesso em: 06 fev. 2023.

NETO, Arilo; CLAUDIO, Dias. Introdução a teste de software. **Engenharia de Software Magazine**, v. 1, p. 22, 2007.