Faculdades Integradas de Taquara - Faccat
Av. Oscar Martins Rangel, 4.500
Taquara, RS, CEP 95600-000

Curso de Sistemas de Informação

AUTOMAÇÃO DE PROCESSOS MANUAIS NO SUPORTE DE PRODUTO UTILIZANDO A FERRAMENTA SELENIUM

Joel dos Passos Moraes Junior
Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil
joeljunior@sou.faccat.br

Luiz Rodrigo Jardim da Silva
Orientador
luizsilva@faccat.br

RESUMO

Este artigo tem como objetivo apresentar o projeto de desenvolvimento de um conjunto de automações de processos que irão auxiliar colaboradores do setor de suporte de produto em uma empresa de tecnologia do Vale dos Sinos, a executar tarefas repetitivas e custosas de forma automática, mais rápida e com menores chances de erro. Desenvolvida com o auxílio da metodologia Scrum solo, esta automação será utilizada no cotidiano do colaborador de suporte para editar componentes de documentações, assinar processadores a incidentes, reportar indicadores de desempenho e métricas de usuários.

Palavras-chave: Automação, Processos Manuais, Suporte de Produto, Empresa de Tecnologia.

MANUAL PROCESS AUTOMATION IN PRODUCT SUPPORT USING SELENIUM FRAMEWORK

ABSTRACT

This meta-paper intends to present the development project of a group of automation related to manual tasks that will assist workers from the product support sector in a tech company based in Vale dos Sinos, to execute repetitive and costly tasks automatically, in a faster pace and decreasing the occurrence of errors. Developed within the Scrum solo methodology, this automation will be used by the support team during their daily activities to edit components from documentations, assign processors to incidents, report Key Performance Indicators and user metrics.

Keywords: Automation, Manual tasks, Product Support, Tech Company.

1 INTRODUÇÃO

Com a evolução tecnológica e o surgimento de sistemas e processos mais complexos, surge a necessidade de garantir a consistência dos dados processados e realizar mais testes e procedimentos em menos tempo. Atualmente, com o grande uso de sistemas web por parte da população, há um maior número de processos e ações executadas por usuários, muitas das quais levam um certo tempo para serem realizadas e pelo fato de dependerem da intervenção humana, as chances de ocorrência de erro aumentam. A partir disso, surgem os processos de automações que ajudam a evitar os problemas supracitados.

Essas automações melhoram a qualidade do software/sistema utilizado, com a mínima necessidade de intervenção humana, ao mesmo tempo que executam mais tarefas em menos tempo (GOJARE; JOSHI; GAIGAWARE, 2015).

Aplicações web vêm se tornado cada vez mais complexas de serem desenvolvidas e testadas, além de que testes manuais requerem mais trabalho. Levando em conta os pontos citados e considerando também o relevante número de softwares implementados como aplicações web, surge como uma alternativa a utilização de ferramentas que auxiliam nessas automações, facilitando todo o processo de desenvolvimento e testes de forma automática (KUMAR; SAXENA, 2015).

Portanto, deve-se avaliar a ferramenta de automação considerando o ambiente em que ela será utilizada e qual o seu objetivo. Todo esse processo de avaliação deve ser executado com o objetivo de evitar retrabalho e evitar gastos desnecessários, sejam eles de tempo ou dinheiro (OLIVEIRA, 2010). A partir disso e de um levantamento realizado na empresa em que o projeto será aplicado, citado no tópico 2.1.1 do presente artigo, pretende-se com esse trabalho desenvolver um conjunto de sistemas de automações que auxiliam na realização de processos manuais, utilizando a ferramenta Selenium para aplicações web. O contexto em que essas automações serão inseridas, se dá no dia a dia do time de suporte de produto de uma empresa do ramo de desenvolvimento de software, localizada em São Leopoldo (RS).

Atualmente nessa empresa, muitas vezes é necessário editar componentes de centenas de artigos de documentação, também conhecidos como ‘Knowledge Base

Articles (KBA – Artigos Base de Documentação)', além de atribuir dezenas de chamados diariamente para cada pessoa e revisar um grande número de incidentes com o intuito de sinalizar campos que devem ser preenchidos.

Dessa forma, essas automações irão auxiliar os engenheiros de suporte na atualização, criação e edição de artigos de documentação utilizados interna e externamente, além da atribuição de chamados para consultores, revisão e categorização de chamados, e geração de relatórios.

Na Seção 2 será abordada a fundamentação teórica relacionada ao estado da arte do trabalho, analisando-se o cenário atual, a importância de automações em um contexto geral, ferramentas de automação e as tecnologias utilizadas no presente trabalho. Especificamente sobre o uso do Selenium com a linguagem de Programação Java e paradigma de orientação a objeto. Ainda nessa seção trata-se sobre trabalhos relacionados.

Na Seção 3 deste trabalho, aborda-se a metodologia de desenvolvimento utilizada em todo o processo de criação, análise e especificação do projeto. A parte de análise e desenvolvimento é mais detalhadamente tratada na Seção 4.

Já na Seção 5, apresentam-se os testes realizados e os resultados obtidos a partir da implementação das automações de processos manuais no contexto analisado e se elas foram capazes de auxiliar os engenheiros de suporte de produto durante a execução de suas atividades.

2 FUNDAMENTAÇÃO TEÓRICA

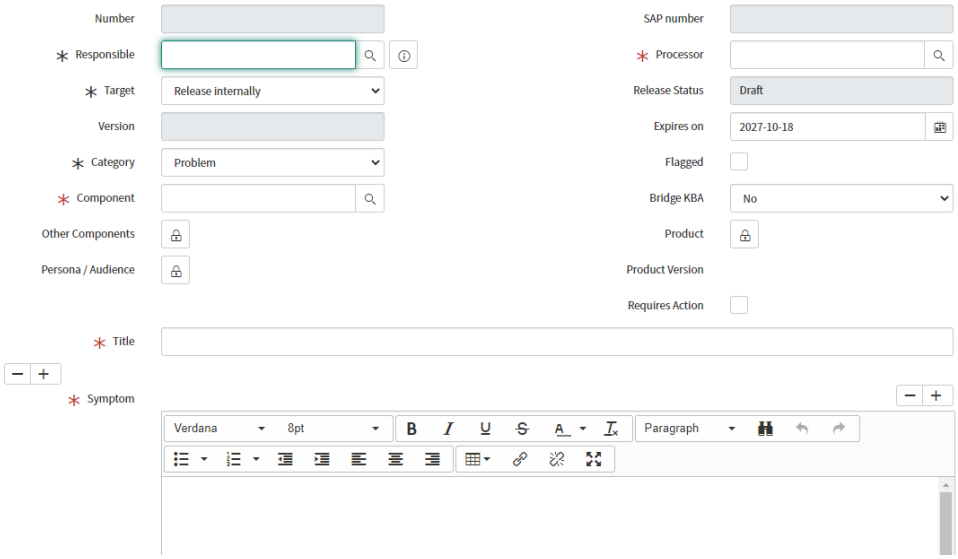
2.1 CENÁRIO ATUAL

Antes de iniciar o desenvolvimento do modelo de projeto, analisou-se internamente algumas das atividades para entender quais delas era mais frequentemente executadas pelos engenheiros de suporte no seu cotidiano e foram levantadas algumas perguntas acerca de uma possível automação de alguns dos processos e da importância desse tipo de atividade.

Para isso, destacaram-se inicialmente algumas tarefas que eram executadas manualmente pelos colaboradores e que levavam um tempo considerável para sua finalização. Dentre elas, pode-se pontuar a edição de alguns campos específicos presentes nas documentações sendo eles o componente (*Component*), nome de

processador (*Processor*), responsável (*Responsible*). Na figura 1 é possível visualizar um exemplo ilustrativo desse tipo de documentação e seus campos apenas para fins de visualização, por esse motivo os campos não estão preenchidos:

FIGURA 1 – Exemplo ilustrativo de documentação



Fonte: Autor (2022).

Outra tarefa constantemente executada é o processo de assinar um colaborador a um incidente específico. Para isso, o usuário deveria acessar o incidente, editar o campo inserindo o nome do futuro processador e então salvar. Esse tipo de ação é executada manualmente para cada incidente diariamente e ao considerar que há um elevado número de casos, a automação passou a ser uma opção a se considerar. Dessa forma, apenas um usuário executaria essa automação e assinaria esses incidentes para uma lista de colaboradores pré-definidos.

Mais um exemplo de atividade executada pelos engenheiros de suporte é a coleta de dados chave de performance a partir do acesso de monitores que contém essa informação. Tendo isso em vista, uma automação que gera um relatório desses dados e os envia por e-mail seria um opção totalmente válida e tiraria essa atribuição manual do próprio colaborador de suporte. Além disso, destacaram-se as opções de geração de relatórios relacionados aos incidentes que não possuíam categorização de erro, cuja categorização se dá a partir do preenchimento de alguns campos no incidente que não são necessariamente mandatórios a nível de sistema mas devem ser preenchidos a nível de processo.

Por último, há a necessidade de enviar mensagens de acompanhamento a incidentes onde o cliente não retornou para o time de suporte. Atualmente o colaborador do suporte acessa o incidente de forma manual, escreve a mensagem para o cliente e envia. Portanto, quando há um grande número de incidentes, essa tarefa passa a ser repetitiva e custosa, sendo totalmente passível de automação pois ela pode ser executada de forma simples contendo uma mensagem padrão de acompanhamento (*follow-up)* e uma lista de incidentes.

2.1.1 Importância da automação

Ferramentas de automação são importantes ferramentas de design e execução de *scripts* de teste os quais consomem menos recursos, principalmente de tempo e alocação de força de trabalho manual (JAIN; KALURI, 2015). Além do mais, com o atual aumento do número de informações a serem processadas por humanos no seu dia a dia, assim como um maior volume de demandas, muitas vezes chega-se a um estado onde a pessoa está totalmente sobrecarregada e as chances de cometer um erro aumenta consideravelmente durante a execução de suas tarefas. Considera-se também que a performance pode ser prejudicada na realização de atribuições e a partir disso levantam-se pontos que justificam a importância de automações em um contexto geral, sendo capaz de reduzir a carga de trabalho e demanda de uma pessoa, assim como diminuir o fator de fatiga, entregar um nível de estabilidade e corretismo de informações, possibilitar ao colaborador a alocação de recursos em outras tarefas até mesmo mais importantes e também realizar as atividades manuais de forma mais rápida (BRETON; BOSSÉ, 2003).

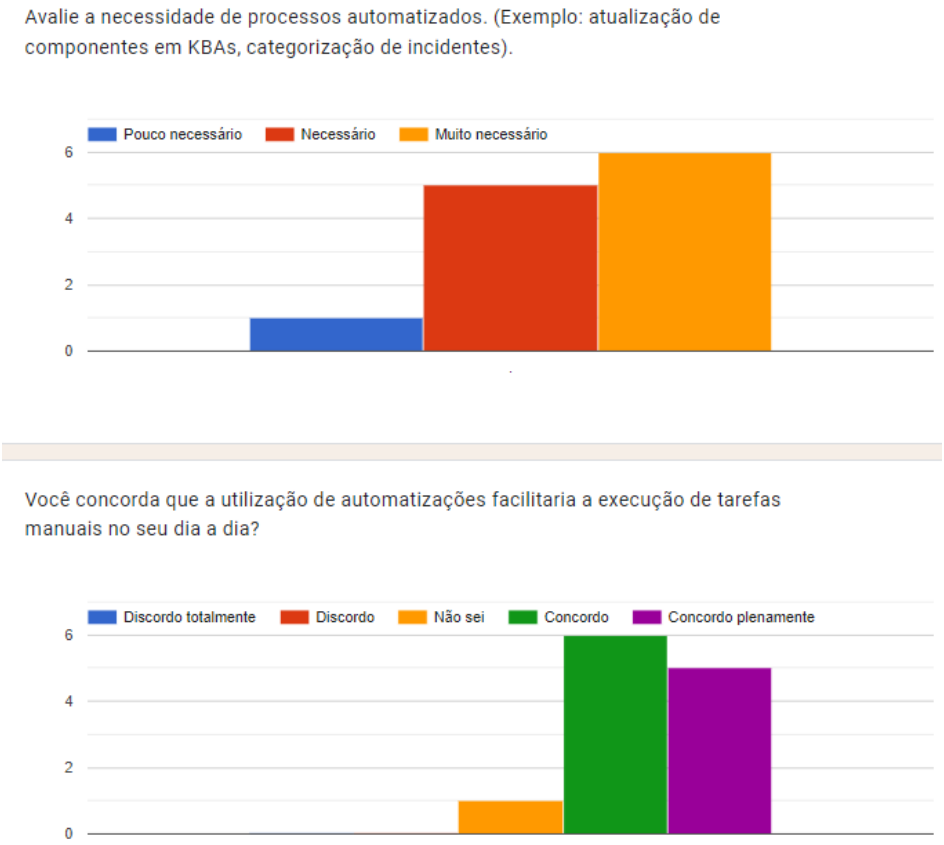
Tendo em vista a análise do cenário atual e a notabilidade e benefícios de automações em um contexto geral, foram realizadas 4 perguntas a respeito da importância que os colaboradores viam em relação ao uso de automações para execução de suas atividades ao analisar o dia a dia do colaborador de suporte. Essas perguntas foram respondidas a partir de um formulário realizado no *Google Forms* no qual foram feitos os seguintes questionamentos:

- Avalie a necessidade de processos automatizados. (Exemplo: atualização de componentes em KBAs, categorização de incidentes).
- Você concorda que a utilização de automatizações facilitaria a execução de tarefas manuais no seu dia a dia?

- Atualmente você conhece algum processo, que é executado de forma automática, e auxilia na execução de alguma tarefa manual no seu trabalho?
- Se sim, descreva esse processo e o nome da ferramenta utilizada.

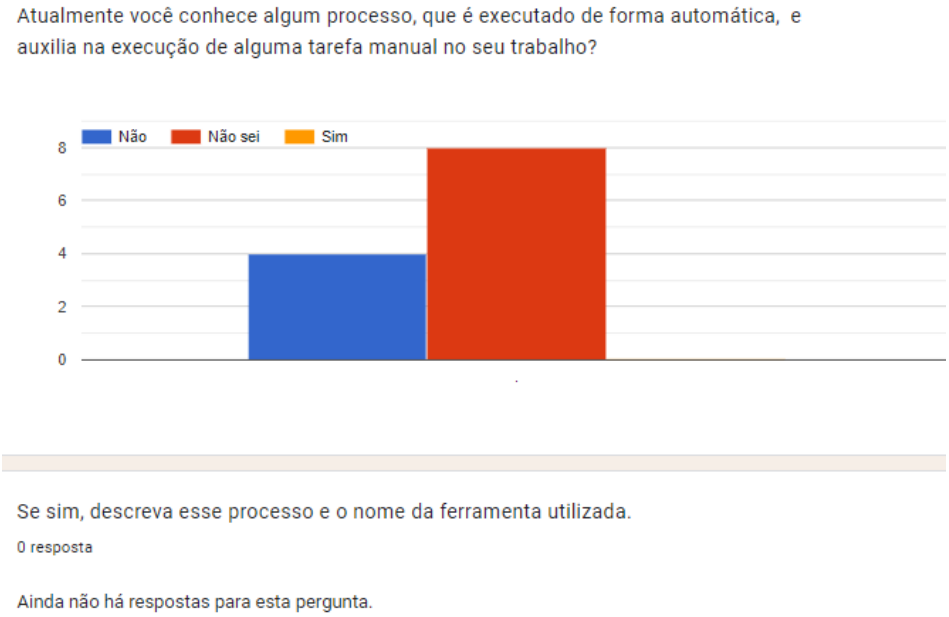
Obtiveram-se 12 respostas dos atuais colaboradores que atuam no suporte de produto e com isso foi possível gerar gráficos de barra os quais ilustram a necessidade vista pelos colaboradores com a utilização dessa automação e a sua concordância em relação a esses tópicos. Na figura 2 e 3 é possível visualizar os resultados obtidos a partir das questões de múltipla escolha.

FIGURA 2 – Resultado de pesquisa de concordância realizada com os funcionários do suporte de produto que atuam no cenário analisado



Fonte: Autor (2022).

FIGURA 3 – Resultado de pesquisa de concordância realizada com os funcionários do suporte de produto que atuam no cenário analisado



Fonte: Autor (2022).

Com os resultados, foi possível entender que as automações são sim necessárias e podem facilitar consideravelmente a execução das tarefas pelos engenheiros de suporte. A partir disso, pôde-se entender a real importância e justificou-se o desenvolvimento do trabalho.

2.2 FERRAMENTAS DE AUTOMAÇÃO

Durante o processo de desenvolvimento do projeto, analisaram-se algumas ferramentas de automação as quais também seriam capazes de atender a demanda e suportar as rotinas de teste. Além do Selenium, framework escolhido para a realização do projeto, consideraram-se opções como Cypress, Katalon e Testsigma. De acordo com as restrições de software da empresa onde será aplicado o projeto, o Selenium framework é o único dentre os citados acima que possui uso autorizado pelo departamento de *Compliance* e segurança e, portanto, a única ferramenta válida para a realização do trabalho no ambiente de trabalho.

2.2.1 Cypress

O framework Cypress possibilita a criação de testes escritos na linguagem JavaScript e funciona em praticamente qualquer front-end ou web site. Diferentemente de outras ferramentas de automação, o Cypress não necessita da instalação de bibliotecas externas para o desenvolvimento de projetos, todas elas já estão inclusas em uma única instalação. Cypress não é uma ferramenta baseada na arquitetura do Selenium e os seus testes são executados juntamente com a aplicação, ao contrário de seu concorrente que executa comandos remotos através de rede (CYPRESS, 2021).

2.2.2 Katalon

Katalon é uma ferramenta desenvolvida com base no Selenium e Appium, é utilizado para testes de aplicativo web, REST e dispositivos móveis. Possui um ambiente de desenvolvimento disponível para todos os sistemas operacionais e plataformas. O Katalon Studio possui compatibilidade com vários sistemas de versionamento como GitHub e GitLab (KATALON, 2021).

2.2.3 Testsigma

Testsigma é uma ferramenta que possibilita a testagem de aplicações web, aplicativos mobile e APIs. Com o seu uso, é possível criar testes regressivos usando processamento de linguagem natural. Possui integração com Browserstack para execução de testes, assim como sendo possível rodar as automações em rede local (TESTSIGMA, 2021).

2.3 TECNOLOGIAS E PARADIGMAS

Nesse tópico serão abordadas as principais tecnologias utilizadas no desenvolvimento deste projeto. A partir do seu uso e integração, é possível programar scripts na linguagem de programação Java que se comunica diretamente com o Selenium e WebDriver for Chrome. Este mesmo WebDriver realiza a comunicação com o navegador e executa as ações especificadas no script programado.

Para o desenvolvimento e codificação dos scripts na linguagem Java, utilizou-se o ambiente de desenvolvimento (*Integrated Development Environment* – IDE) Jetbrains IntelliJ IDEA 2021.x Community.

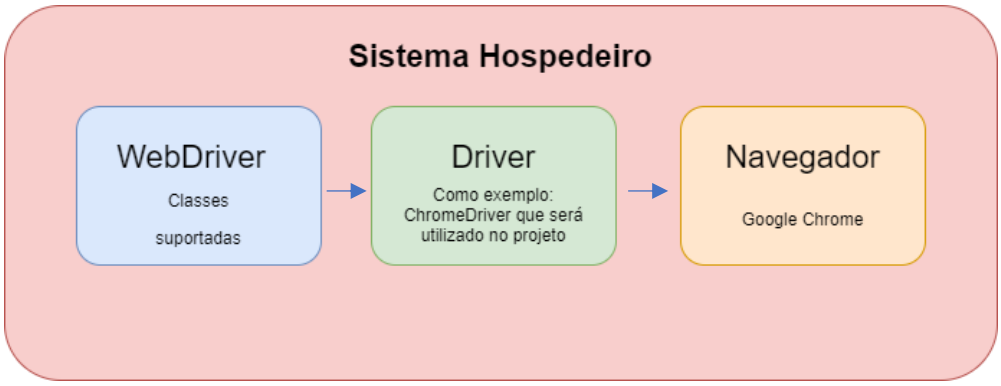
2.3.1 Selenium

Selenium é um framework utilizado principalmente para desenvolver automações de aplicações web tendo como objetivo testar funcionalidades de um sistema e ilustrar o comportamento de um usuário nesse ambiente através de passos que simulam as ações tomadas pelo usuário (SELENIUM, 2021).

Neste trabalho utilizou-se especificamente o Selenium WebDriver for Chrome, chamado de ChromeDriver que é utilizado para controlar ações no navegador Google Chrome através da localização de elementos web e interação com suas funcionalidades (CHROMEDRIVER, 2021).

O WebDriver consiste em uma API orientada a objeto que realiza a comunicação com um navegador a partir de um Driver. Essa comunicação entre WebDriver e navegador é ilustrada com a figura 4 abaixo.

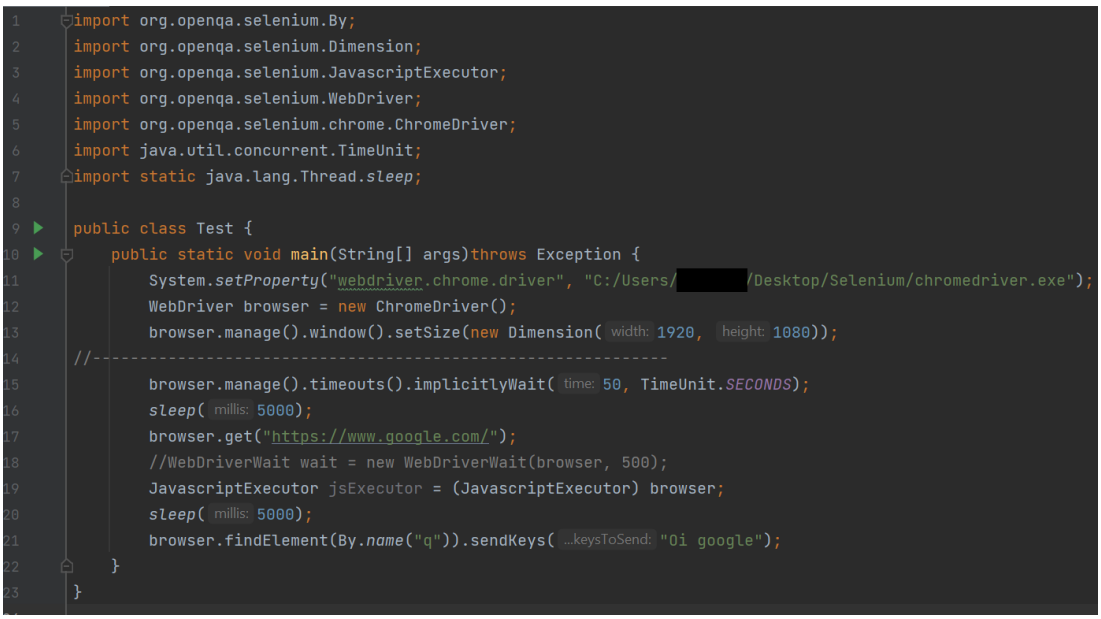
FIGURA 4 – Comunicação entre WebDriver e Navegador



Fonte: Autor (2022).

A Figura 5 demonstra uma simples porção de código que ao ser executada, abre o navegador Google Chrome e digita as palavras ‘Oi Google’ na barra de pesquisa. É uma representação de código que ao ser executado ilustra como é feita a comunicação entre WebDriver e navegador. Neste mesmo script também é possível visualizar os pacotes e interfaces importados que são necessários para realizar essa comunicação.

FIGURA 5 – Código em linguagem Java exemplificando a comunicação entre WebDriver e Navegador



```java
import org.openqa.selenium.By;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.concurrent.TimeUnit;
import static java.lang.Thread.sleep;

public class Test {
    public static void main(String[] args)throws Exception {
        System.setProperty("webdriver.chrome.driver", "C:/Users/[redacted]/Desktop/Selenium/chromedriver.exe");
        WebDriver browser = new ChromeDriver();
        browser.manage().window().setSize(new Dimension( width: 1920,  height: 1080));
//------------------------------------------------------------
        browser.manage().timeouts().implicitlyWait( time: 50, TimeUnit.SECONDS);
        sleep( millis: 5000);
        browser.get("https://www.google.com/");
        //WebDriverWait wait = new WebDriverWait(browser, 500);
        JavascriptExecutor jsExecutor = (JavascriptExecutor) browser;
        sleep( millis: 5000);
        browser.findElement(By.name("q")).sendKeys( ...keysToSend: "Oi google");
    }
}
```

Fonte: Autor (2022).

2.3.2 Paradigma de Orientação a Objeto

O paradigma de Orientação a Objeto possibilita estruturar e organizar o código mais facilmente, possibilitando a reutilização do código para que o desenvolvedor não precise repetir toda uma mesma lógica já presente no código (PRESSMAN, 2005). Este ponto facilita a manutenção do código e a possibilidade de alterações futuras. Dessa forma, o desenvolvimento dos scripts de automação poderá ser auxiliado, os quais mesmo executando rotinas diferentes com diferentes objetivos, podem utilizar partes de código em comum (W3schools, 2021).

Com esse paradigma pode-se reutilizar um objeto já implementado, modificá-lo e transformá-lo em um objeto novo sem que altere a estrutura e a lógica presente no objeto anterior, fator que poderia afetar outras partes do sistema que utilizam esse mesmo objeto caso não existisse essa possibilidade de reutilização. Esse é um princípio muitas vezes atribuído ao conceito de Polimorfismo. Farinelli (2007) exemplifica da seguinte forma:

> Podemos retornar ao nosso exemplo do telefone, onde sabemos que a companhia telefônica pode efetuar diversas alterações no tratamento telefônico, como é o caso de converter as linhas analógicas para digitais, ou aumentar a quantidade de serviços oferecidos, e isso tudo sem modificar o método que você utiliza para telefonar.

2.3.3 Linguagem de Programação Java

Java é uma linguagem de programação considerada como a base da maioria das aplicações em rede, na qual foi projetada com o objetivo de permitir o desenvolvimento de aplicações portáteis que podem ser executadas pela maior parte dos aparelhos que possuem capacidade computacional (JAVA, 2021). Dentre as linguagens de programação suportadas pelo framework Selenium, destaca-se a utilização de Java.

2.3.4 Ambiente de Desenvolvimento

Um Ambiente de Desenvolvimento, também conhecido como *Integrated Development Environment* (IDE) é uma ferramenta utilizada primordialmente para criação e edição de código, depuração e execução de código. Cada IDE possui suas funcionalidades específicas, mas como ponto em comum podemos destacar o fato de que elas possibilitam a criação e execução de código em apenas um ambiente integrado (ALURA, 2022). Existem vários ambientes de desenvolvimento (IDE) e para o presente trabalho utilizou-se JetBrains IntelliJ IDEA 2021.x Community, versão gratuita. Justifica-se a escolha pois essa IDE, sob certas condições, possui seu uso permitido no ambiente interno da empresa onde o projeto é estabelecido (INTELLIJ, 2021).

2.4 Trabalhos Relacionados

Existem alguns trabalhos semelhantes que fizeram uso do Selenium para a execução de tarefas repetitivas e manuais, como por exemplo Crozatti (2021)**,** que utilizou-se do Selenium para executar casos de testes automatizados em uma plataforma e-commerce. Processo pelo autor descrito como extremamente útil para evitar erros e anomalias possivelmente causadas por ações humanas.

Outro trabalho relacionado, na visão de Romanini e Sotto (2019), aborda um processo de interações com um sistema web *Internet Herokuapp* em que realizaram-se rotinas e métodos para efetuar login e testar os tempos de resposta para definir falhas.

Referencia-se também o trabalho de Moraes (2013), onde realizaram-se testes de regressão e automação em processos de testagem de software na empresa

*Web*Aula. Dentre os benefícios trazidos por essas automações, destacou-se que os colaboradores passaram a ter mais tempo para se dedicar às suas atribuições sem precisar ficar horas executando testes repetitivos e também conseguiram encontrar erros no sistema analisado antes mesmo de chegar ao cliente (MORAES, 2013).

De qualquer forma, não há nenhum processo de automação atualmente no suporte de produto para as tarefas citadas na introdução, a partir disso destaca-se a importância e a relevância de tal projeto. Como não há trabalho relacionado considerando o mesmo cotidiano e ambiente em si, o projeto como um todo é um diferencial pois ele permite executar mais tarefas em menos tempo e com menores chances de erro. A automação desenvolvida contribui diretamente com toda a alocação de recursos no time de suporte e destaca-se pelo fato de ser inovador no contexto do suporte de produto na empresa analisada. Por mais que existam trabalhos de automação com Selenium, como os supracitados, o presente trabalho não envolve testes automatizados para analisar ferramentas, encontrar erros de lógica e avaliar o desempenho do sistema, mas sim para executar tarefas nesse sistema que atualmente requerem intervenção manual e repetitiva do usuário. Portanto, ao considerar o ambiente de suporte de produto de uma empresa de desenvolvimento de software localizada em São Leopoldo (RS), o presente trabalho destaca-se por ser o pioneiro neste cenário.

3 METODOLOGIA

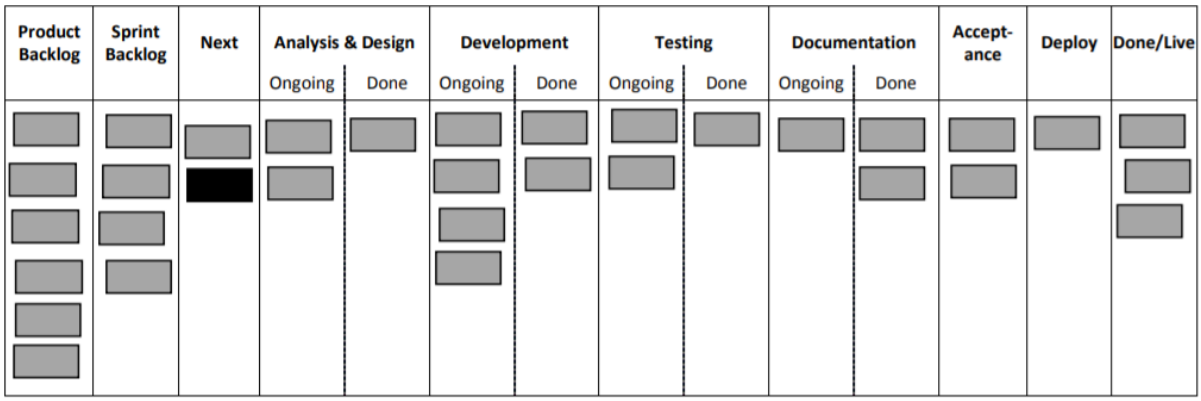
A metodologia utilizada para o desenvolvimento deste trabalho é de caráter qualitativo pois ela não se baseia intrinsecamente em números, mas sim nas experiências dos engenheiros de suporte de produto considerados como público-alvo nesse trabalho. A metodologia se dá a partir do entendimento do indivíduo em relação a atividade por ele praticada, compreendendo as suas motivações e necessidades ao executar suas tarefas no dia a dia.

Inicialmente realizou-se um levantamento das funcionalidades atendidas pela automação em si. A partir disso, com a utilização da metodologia Scrum Solo, foi possível organizar as soluções a serem desenvolvidas e distribuir conforme a ordem de desenvolvimento e prioridade. Para o desenvolvimento do sistema, foi escolhido o modelo de desenvolvimento de software Scrum solo com o uso do quadro Kanban para gerenciar o projeto. O Scrum é uma metodologia que divide o ciclo de

desenvolvimento em Sprints, sendo que em cada uma dessas Sprints é realizado uma Planning (MAHNIC, 2014).

O quadro Kanban é considerado uma representação visual do fluxo de trabalho, dividido em colunas, ajuda a controlar as tarefas a fazer, o que já foi feito, o que está em progresso e também o que está em teste. Dessa forma, o desenvolvedor consegue focar nas funcionalidades pendentes e na entrega contínua à medida que novas soluções ficam prontas (MAHNIC, 2014). Na Figura 6 é possível visualizar um quadro Kanban completo composto por diversas colunas de controle.

FIGURA 6 – Representação de Quadro Kanban



Fonte: MAHNIC (2014)

4 ANÁLISE E DESENVOLVIMENTO DO SISTEMA

Na etapa de análise, identificaram-se os requisitos funcionais e não funcionais do sistema em geral.

4.1. Requisitos

TABELA 1 – Requisitos Funcionais

RF1:	Assinar processadores de forma automática
RF2:	Reportar incidentes duplicados
RF3:	Editar, adicionar e remover componentes de documentações
RF4:	Editar, adicionar e remover processadores de documentações
RF5:	Reportar incidentes sem categorização de erro e KBA
RF6:	Reportar informações de CSATs, KM, KPIs (Indicadores de Performance)
RF7:	Envio automático de mensagens de follow-up a clientes

RF8:	Visualizar as opções de automação através de uma interface gráfica

Fonte: Elaborado pelo autor (2022)

TABELA 2 – Requisitos Não Funcionais

RNF1:	Disponibilidade: O sistema estará disponível para qualquer usuário que possua o seu código-fonte e poderá ser utilizado apenas caso o usuário esteja online, ou seja, conectado à internet.
RNF2:	Segurança: Para que seja feito o acesso pelo usuário, além de estar conectado à internet, é necessária a conexão via VPN e utilizar-se de um notebook corporativo também disponibilizado pela empresa, fatores os quais servem como autenticação em nível de servidor e nível de aplicação, mais especificamente através de Autenticação única chamada de Single Sign-On (SSO). Dessa forma, para que se possa usufruir das automações, o usuário deve ter os respectivos acessos a sistemas internos e permissões de edição/remoção/adição de documentações, rotinas e incidentes.
RNF3:	Usabilidade e implementação: O sistema roda localmente através de um executável, localizado localmente na máquina do usuário e quando executa comunica-se com ambiente web, especificamente acessando apenas o navegador Google Chrome através da ferramenta Selenium. Para isso, as rotinas são desenvolvidas na linguagem Java.
RNF4:	Organizacionais: Para o desenvolvimento do sistema são utilizadas apenas ferramentas que estão em conformidade com os padrões empresariais estabelecidos e que estão autorizados pelo time de compliance da organização.

Fonte: Elaborado pelo autor (2022)

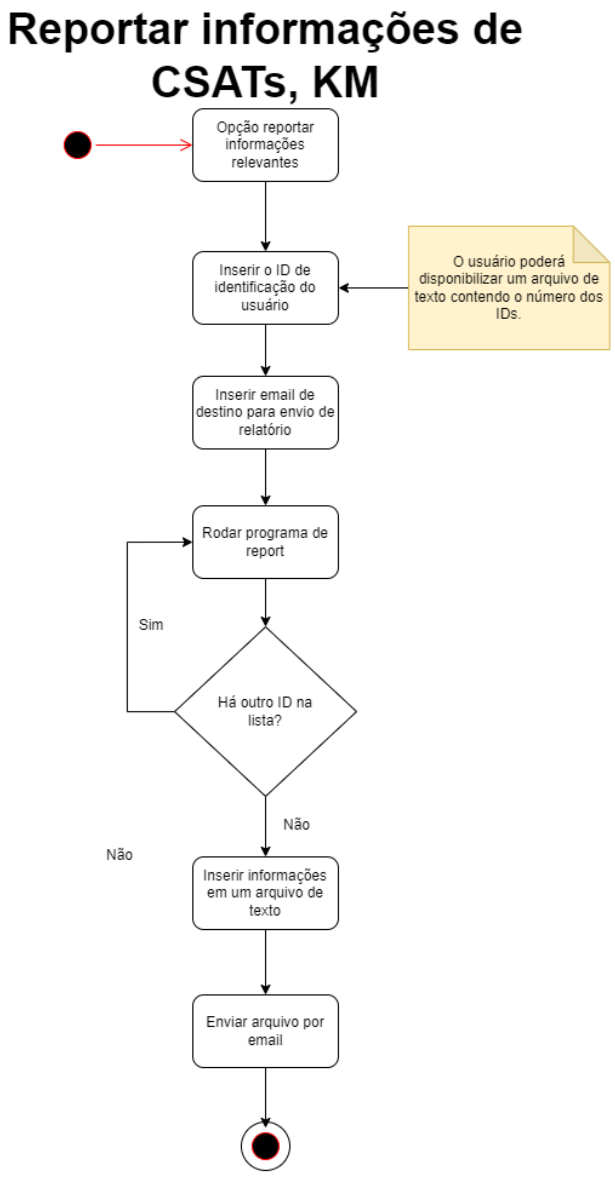
4.2. Diagrama de atividades

À medida que os requisitos funcionais foram definidos, iniciou-se a modelagem do software a partir de um diagrama de atividades.

Diagrama de Atividades consiste em um fluxograma que une as pessoas de diferentes áreas e possibilita um entendimento geral em relação a como um processo funciona. É um gráfico de fluxo que ilustra as etapas de um processo sequencial, ou

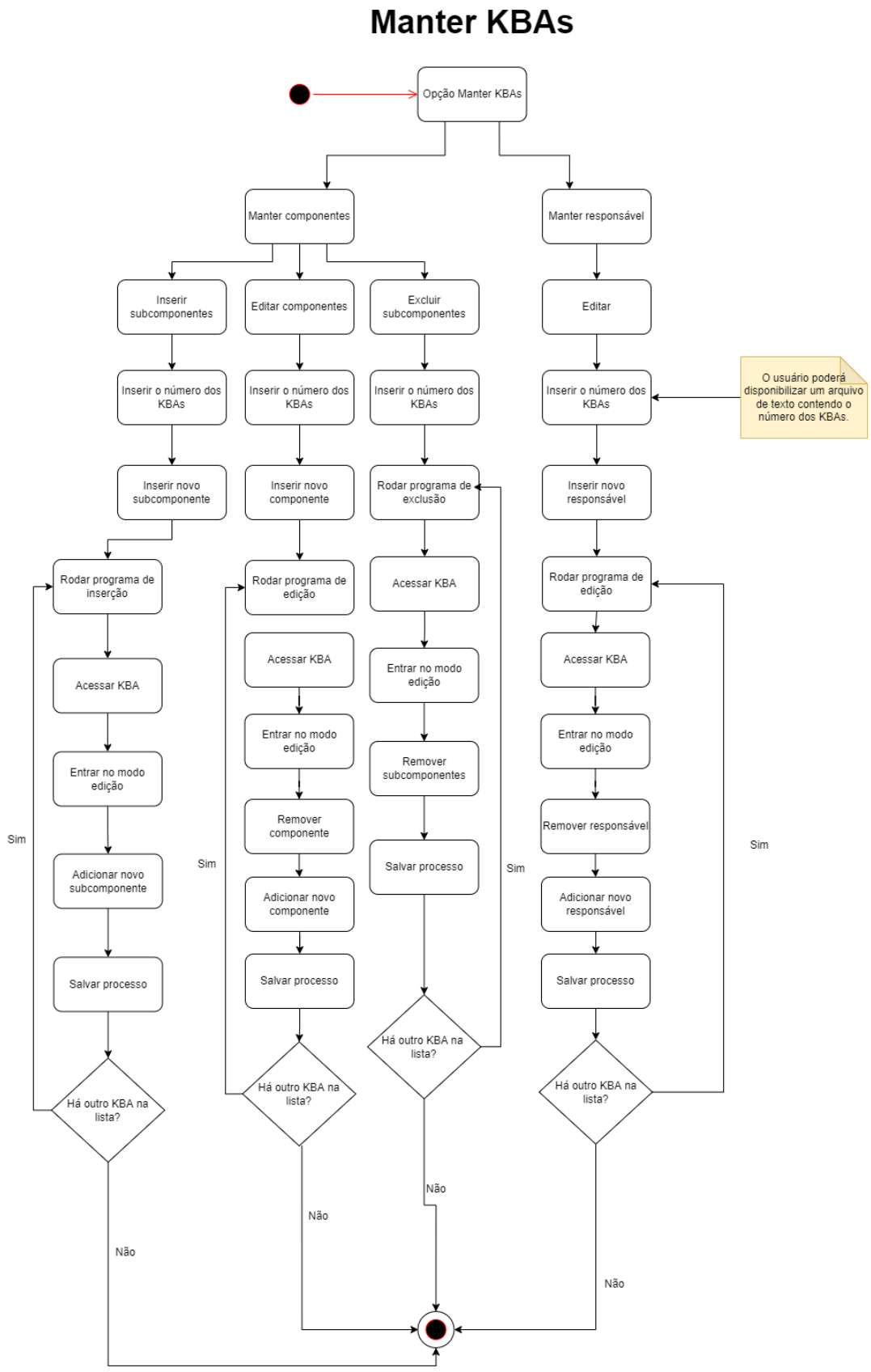
seja, ele demonstra o fluxo de atividades que ocorrem ao executar determinada função de um sistema (FERREIRA; MARTINS, 2010).

FIGURA 7 – Diagrama de atividades relacionado a reportar informações de indicadores chave de performance



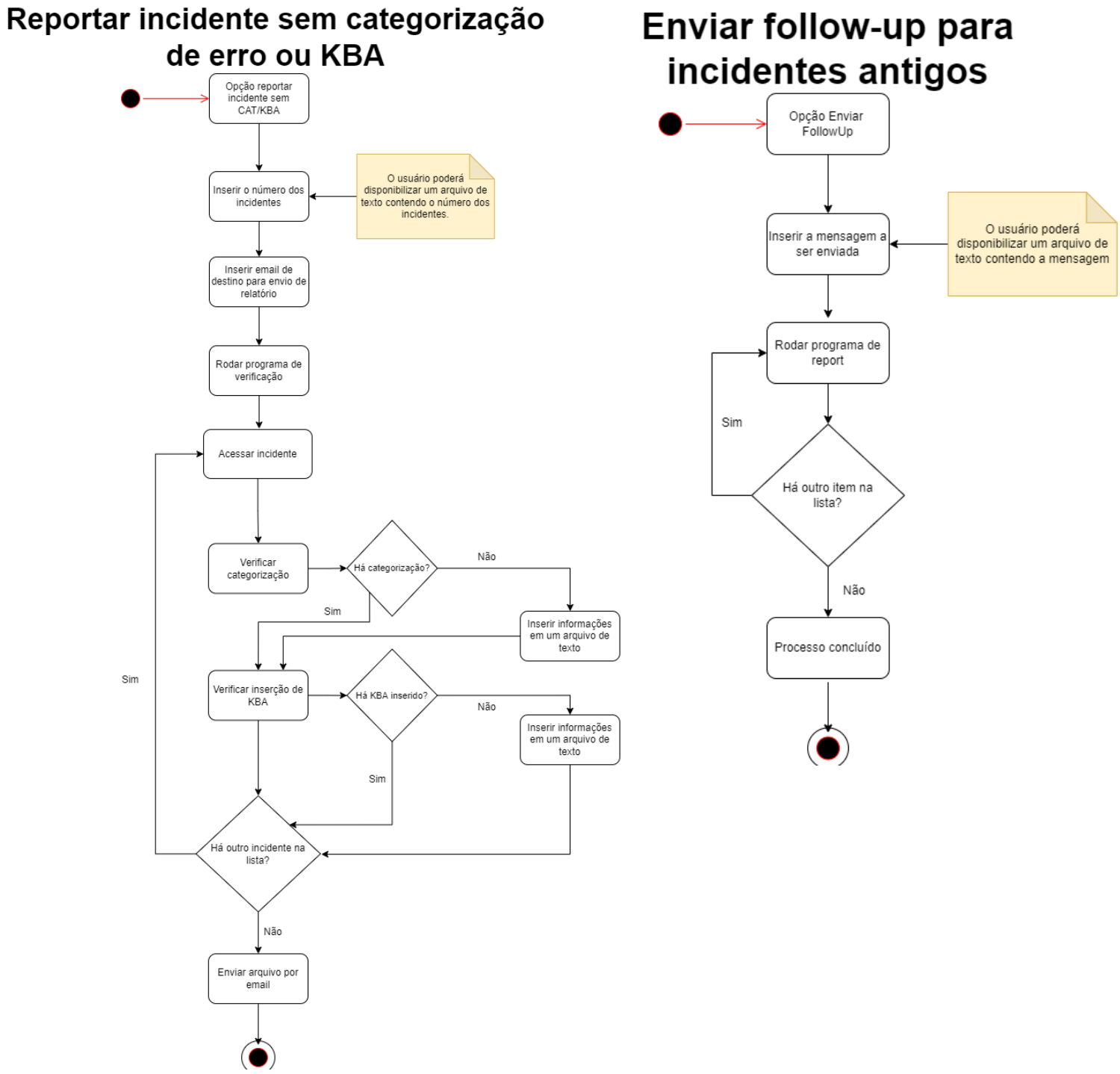
Fonte: Autor (2020).

FIGURA 8 – Diagrama de atividades relacionado a manter informações de documentações, também chamados de *Knowledge Base Articles (KBAs).*



Fonte: Autor (2022).

FIGURA 9 – Diagrama de atividades relacionado a reportar incidentes sem categorização de erro e também enviar mensagens de seguimento *(follow-up)* para incidentes antigos que ficaram sem retorno do usuário.

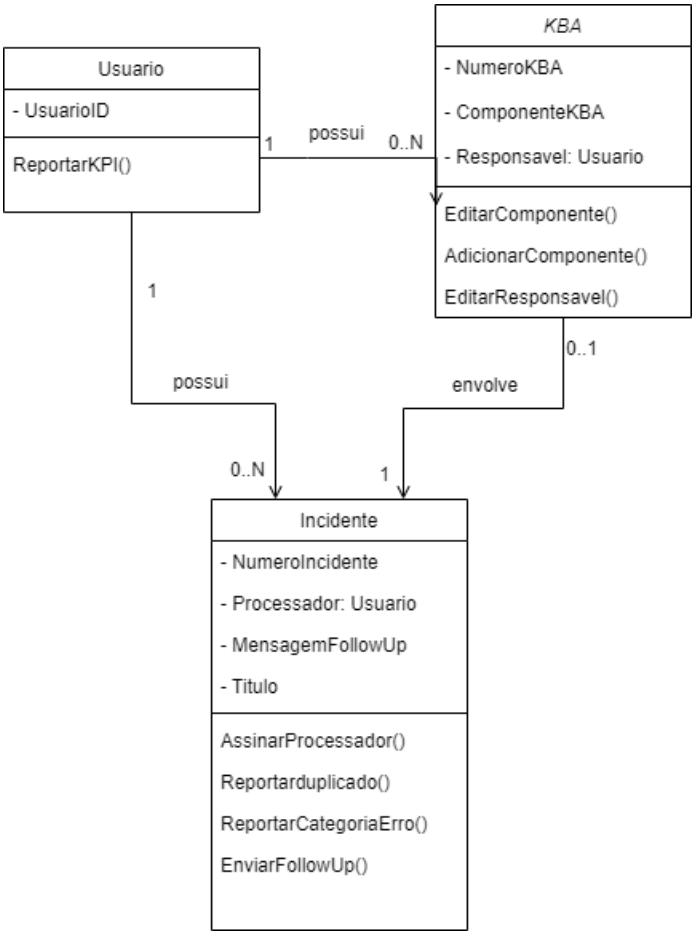


Fonte: Autor (2022).

4.3 Diagrama de Classe

O Diagrama de Classe consiste em um modelo gráfico que ilustra a organização entre as classes utilizadas no projeto de desenvolvimento e suas ligações a partir de métodos e atributos definidos. São diagramas que servem de modelo para os objetos e modelam os componentes de um sistema (SANTOS, 2019).

FIGURA 10 – Diagrama de classes relacionado à estrutura da aplicação.



Fonte: Autor (2022).

4.4 Desenvolvimento do projeto

4.4.1 Desenvolvimento de scripts

Para o desenvolvimento dos scripts utilizou-se a linguagem Java, criaram-se as classes Incidente, KBA e Usuario e os métodos assinarProcessador(),

enviarFollowUp(), reportarKPI(), editarComponente(), adicionarComponente(), editarResponsavel(), reportarCategoriaErro() e reportarDuplicado().

Além disso, para fins visuais, desenvolveu-se um menu de interação através do Swing, um framework utilizado para criação de interfaces gráficas de usuário com a linguagem Java. Esse menu disponibiliza inicialmente as três principais classes, sendo elas Usuario, KBA e Incidente, e em cada uma delas é possível encontrar as funcionalidades de automação que são basicamente os métodos citados anteriormente.

O método AssinarProcessador() acessa uma lista de incidentes sem responsável e, a partir de outra lista contendo o nome dos processadores, vai assinando um processador para cada incidente. Antes desse método ser chamado, o usuário adiciona os nomes dos processadores através de um menu disponibilizado pela interface gráfica. Dessa forma, a automação percorre a lista de incidentes, acessa esse incidente e edita o processador utilizando o primeiro nome da lista de processadores definida na interface, assim que essa operação é feita o incidente é removido automaticamente da lista. Isso pode ocorrer por um número de vezes determinado via código ou até a lista de incidente estiver vazia.

Em relação aos métodos EditarComponente() e AdicionarComponente(), o usuário inicialmente define uma lista em um arquivo de texto contendo o código identificador de cada documentação (KBA). A partir disso, através da interface gráfica, o usuário adiciona o novo componente a ser editado ou adicionado no KBA e o processo ocorre de forma automática. O método acessa cada item da lista através do número identificador, edita e salva essa documentação. Isso ocorre de forma automática a partir do momento que o usuário define a lista de KBAs e o novo componente.

O método EnviarFollowUp() acessa uma lista de incidentes pré-definida e disponibiliza ao usuário uma interface para a escrita de uma mensagem ao cliente. Dessa forma, a automação acessa cada incidente, um a um e envia a respectiva mensagem ao cliente até que a lista desses incidentes esteja vazia.

Os métodos ReportarKPI(), ReportarCategoriaErro() e ReportarDuplicado() quando executados enviam por email ou disponibilizam ao usuário de forma local um relatório de informações relevantes a indicadores chave de desempenho e métricas internas sobre incidentes sem categoria de erro ou incidentes duplicados.

5 RESULTADOS

Tendo em vista que os colaboradores do suporte de produto precisam editar componentes de centenas de artigos de documentação, atribuir dezenas de chamados diariamente para cada colaborador, revisar um grande número de incidentes para analisar o preenchimento e enviar repetitivas mensagens de *follow-up* a clientes, pode-se entender que automações dessas tarefas seriam de grande valia.

Como exemplo de fluxo, tem-se o cenário onde se faz necessário editar os componentes de centenas de documentações (KBA). Esse processo atualmente é feito por um colaborador do suporte que acessa cada KBA, edita o campo de componente e salva o processo. Trata-se então de uma tarefa totalmente repetitiva e que leva um certo tempo para finalizar. Para isso, a aplicação permite que essas alterações sejam feitas de forma automatizada, em que se faz necessário apenas algumas ações do suporte. Considera-se então o método EditarComponente() nesse processo.

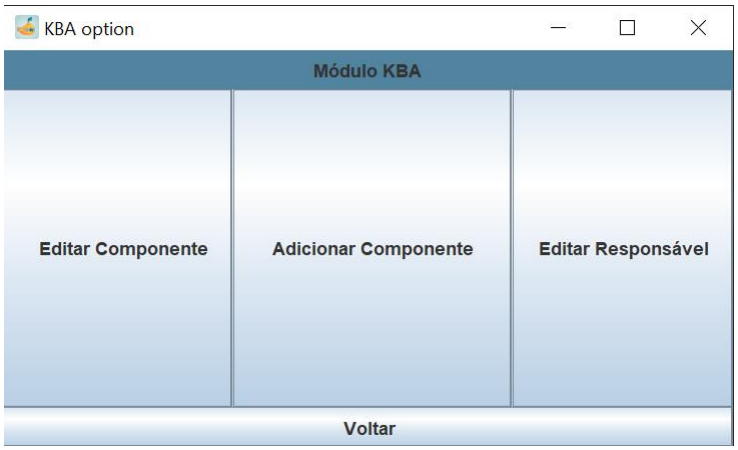
Primeiramente, define-se uma lista contendo o número identificador de cada um desses KBAs, essas informações são inseridas pelo próprio usuário em um arquivo de texto acessado pela aplicação. Ao executá-la, a seguinte interface de menu é disponibilizada ao usuário:

FIGURA 11 – Interface que disponibiliza os módulos de automação da aplicação.

Fonte: Autor (2022).

Assim que o usuário seleciona a opção ‘Módulo KBA’, ele é direcionado ao próximo menu que contém as automações disponíveis que estão relacionadas às documentações:

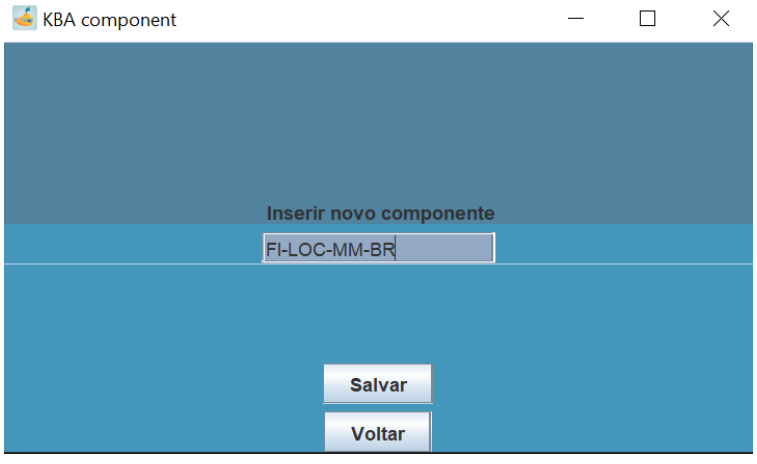
FIGURA 12 – Interface que disponibiliza as automações relacionadas à documentações.



Fonte: Autor (2022).

Seguindo o fluxo de edição de componentes, a opção ‘Editar Componente’ é selecionada pelo usuário e então a página é direcionada ao campo que o usuário deve preencher contendo o novo componente:

FIGURA 13 – Campo para inserção de novo componente.

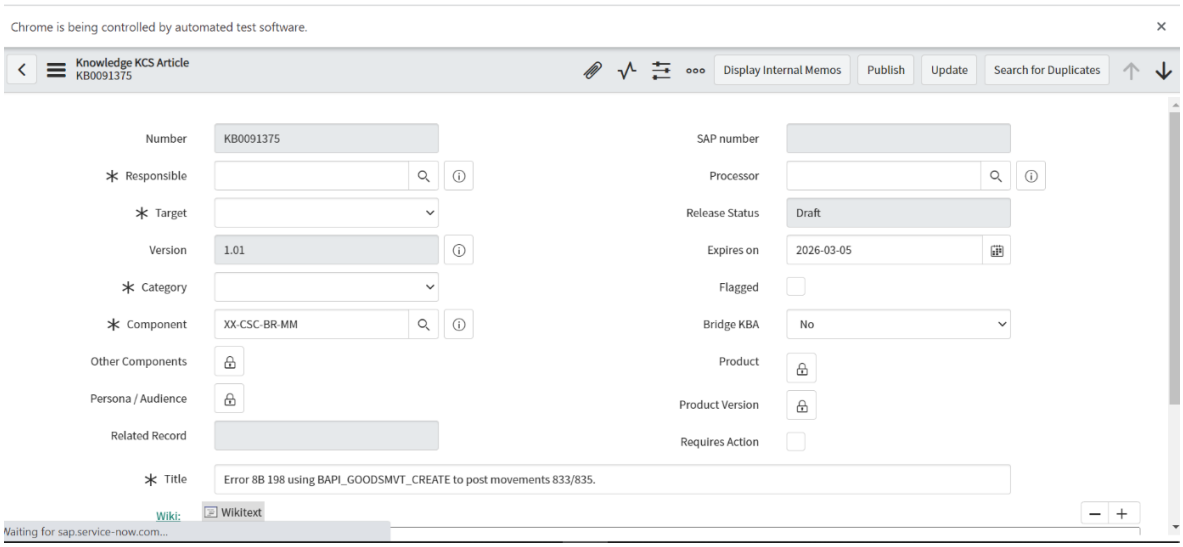


Fonte: Autor (2022).

O novo componente é inserido no campo de entrada como ‘FI-LOC-MM-BR’. Ao clicar em ‘Salvar’, o script de automação é executado e ele abre no próprio navegador o primeiro KBA identificado no arquivo de texto anteriormente citado. Esse KBA é identificado pelo campo ‘Number’ presente na figura 14. Mesma figura que

demonstra a documentação interna com o componente antigo, o qual será alterado automaticamente pelo novo componente inserido pelo usuário.

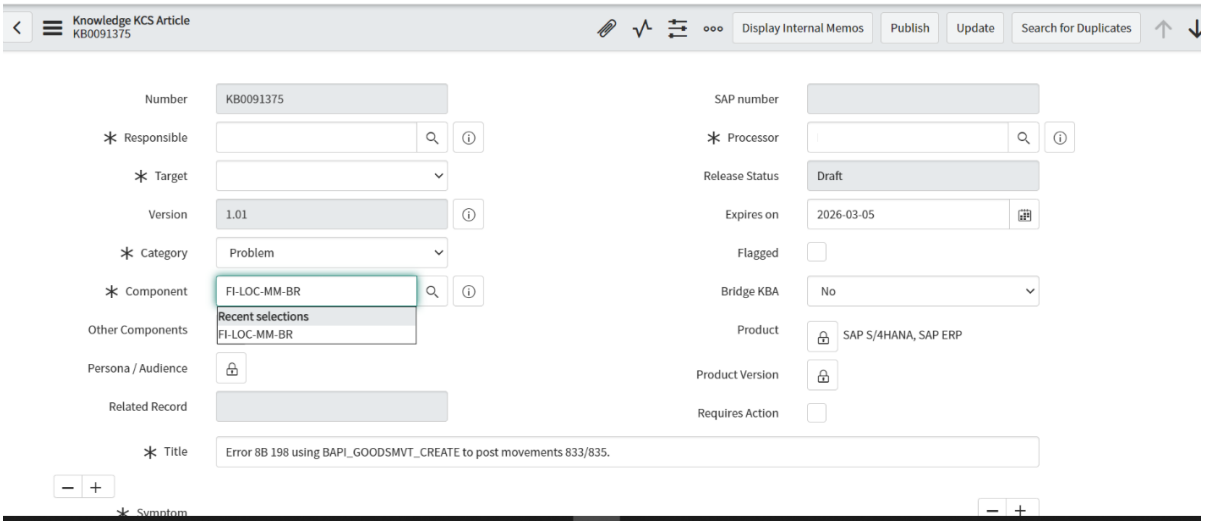
FIGURA 14 – Documentação localizada pelo número identificador e acessada no navegador.



Fonte: Autor (2022).

A aplicação então reconhece o campo ‘Component’ e entende que seu valor será alterado pelo valor ‘FI-LOC-MM-BR’:

FIGURA 15 – Campo de componente sendo editado na documentação.



Fonte: Autor (2022).

Assim que o campo é editado, as alterações são salvas a partir do botão ‘Update’ e então a aplicação segue esse mesmo processo para a próxima documentação da lista pré-definida pelo usuário. Os passos executados na

plataforma onde o KBA se localiza são os mesmo tomados por um usuário manualmente, porém com a aplicação da automação não há a necessidade do usuário repetir esse processo a cada documentação a ser editada.

A partir disso, com o desenvolvimento e a testagem dos scripts de automação, foi possível editar informações de centenas documentações sendo apenas necessário que o usuário disponibilizasse uma lista dessas documentações e a nova informação a ser editada nesses KBAs. Foi também possível assinar uma lista de incidentes a seus processadores de forma automática, sem que o usuário precisasse acessar um incidente de cada vez. A automação conseguiu executar essa tarefa recebendo como parâmetro do usuário a lista de incidentes e outra lista contendo o nome dos respectivos processadores.

6 CONSIDERAÇÕES FINAIS

As automações desenvolvidas possibilitaram aos engenheiros de suporte a atualização, criação e edição de artigos de documentação utilizados de forma interna na empresa apenas pelo colaboradores da empresa de tecnologia e externamente por seus clientes, sem a necessidade de executar tarefas repetitivas, a geração de relatórios contendo métricas relevantes de desempenho e também permitiram que o usuário se comunicasse com dezenas de clientes de uma forma muito mais fácil e menos custosa. A comunicação de *follow-up*, mensagem enviada a clientes pelo fato de que os incidentes não foram respondidos por eles, pôde ser feita através de uma mensagem padrão disponibilizada pelo próprio colaborador do suporte e pelo acesso a uma lista de incidentes já definida via código.

Portanto, com o desenvolvimento dos scripts citados anteriormente, o colaborador do suporte de produto pôde executar mais tarefas em menos tempo, sem a necessidade de agir manualmente em cada incidente ou KBA e diminuindo as chances de adicionar uma informação incorreta. Foi possível concluir que o processo antigo tornou-se totalmente obsoleto à medida que o uso dessas automações ocorreu, inovando e melhorando todo o processo executado pelo suporte e possibilitando um bem-estar maior aos colaboradores do suporte.

Além dos pontos citados anteriormente, essas automações quando executadas são mais eficientes que a interação do usuário com o sistema em si e requerem uma intervenção humana mínima. Consequentemente, menos recursos laborais são necessários, e um trabalho que antes era executado por 4 colaboradores, pode agora ser executado por apenas um desses engenheiros de suporte. Fatores os quais colaboram para uma melhor alocação de recursos, possibilitando maior produtividade e impactando positivamente todo o time de suporte.

Mesmo que o custo de implementação inicial seja de certa forma alto pois o tempo para desenvolver todo o processo levou quase 8 meses, os benefícios decorrentes desse investimento superam os gastos dessa implementação e promovem um ambiente de trabalho menos estressante e mais eficiente. Leva-se em conta que o time de suporte tratado possui conhecimentos técnicos de programação e depuração e suas atribuições não se resumem a atender usuários finais de um sistema como por exemplo um help desk, mas sim tratar incidentes com consultores de empresas que estão implementando o sistema ERP suportado por esse time, depurando códigos e lógicas desse sistema e endereçando possíveis correções ao time de desenvolvimento desse mesmo setor. Conclui-se então que todos os colaboradores de suporte possuem conhecimento técnico o suficiente para executar essas automações e, se necessário, corrigir e adaptar os scripts via programação.

Como projeto futuro, pretende-se estender essas automações a outros times da empresa considerada, disponibilizando não só ao suporte de produto como também a outros setores como desenvolvimento e qualidade a opção de automatizar processos manuais.

REFERÊNCIAS

ALURA. **Integrated Development Interface.** Disponível em: <https://www.alura.com.br/artigos/o-que-e-uma-ide>. Acesso em: 20 out. 2022.

BRETON, Richard; BOSSÉ, Éloi. **The cognitive costs and benefits of automation**. Defense Research and Development, 2003. Disponível em: <https://apps.dtic.mil/sti/citations/ADA422303>. Acesso em: 20 out. 2022.

CROZATTI, Fernando. **Automação dos testes de validação em uma plataforma de comércio eletrônico.**, 2021. Disponível em: <l1nq.com/0Un9y>. Acesso em: 20 out. 2022.

CYPRESS. **Testing has been broken for too long.** Disponível em: < https://www.cypress.io/how-it-works>. Acesso em: 20 out. 2022.

CHROMEDRIVER. **ChromeDriver – WebDriver for Chrome.** Disponível em: < https://chromedriver.chromium.org/home>. Acesso em: 20 out. 2022.

FARINELLI, Fernanda. **Conceitos Basicos de programação orientada a objetos.** Instituto Federal Sudeste de Minas Gerais, 2007. Disponível em: < encurtador.com.br/lCFK7>. Acesso em: 20 out. 2022.

FERREIRA, Jeferson; MARTINS, Eliane. **Fluxo de Exceções Intra Procedimentais a partir do Diagrama de Atividades da UML 2.0**. Instituto de Computação Universidade Estadual de Campinas, Campinas, São Paulo, 2010. Disponível em: <l1nq.com/EGV6V>. Acesso em: 20 out. 2022.

GOJARE, Satish; JOSHI, Rahul; GAIGAWARE, Dhanashree. **Analysis and design of selenium webdriver automation testing framework.** Procedia Computer Science, v. 50, p. 341-346, 2015. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050915005396>. Acesso em: 20 out. 2022.

INTELLIJ. **Por que IntelliJ IDEA**. Disponível em: < https://www.jetbrains.com/pt-br/idea/>. Acesso em: 20 out. 2022.

JAIN, C. R.; KALURI, Rajesh**. Design of automation scripts execution application for selenium webdriver and test NG framework.** ARPN J Eng Appl Sci, v. 10, p. 2440-2445**,** 2015. Disponível em: <l1nq.com/Oxj8r>. Acesso em: 20 out. 2022.

JAVA. **O que é a Tecnologia Java e porque preciso dela?**. Disponível em: < https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html>. Acesso em: 20 out. 2022.

KATALON. **An all-in-one test automation solution.** Disponível em: <https://www.katalon.com>. Acesso em: 20 out. 2022.

KUMAR, Ajeet; SAXENA, Sajal. **Data driven testing framework using selenium WebDriver**. International Journal of Computer Applications, v. 118, n. 18, 2015. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.9076&rep=rep1&type=pdf>. Acesso em: 20 out. 2022.

MAHNIC, Viljan**. Improving software development through combination of scrum and kanban**, 2014. Disponível em: <l1nq.com/DpX5u>. Acesso em: 20 out. 2022.

MORAES, Marina Adriana Faria de. **Automação de testes de regressão e Smoke test em aplicações WEB com o SELENIUM**, 2013. Disponível em: <http://192.100.247.84/handle/prefix/994>. Acesso em: 20 out. 2022.

OLIVEIRA, André. **Novas Funcionalidades e Manutenção no Sistema Online de Distribuição de Disciplinas**. 2017. Disponível em: <l1nq.com/Qpivm>. Acesso em: 20 out. 2022.

PRESSMAN, Roger S. **Software engineering: a practitioner's approach. Palgrave macmillan**, p 100-101, 2005.

ROMANINI, Isabela; SOTTO, Eder. **SELENIUM WEB DRIVER na evolução dos testes manuais**, 2019. Disponível em: <https://revista.fatectq.edu.br/interfacetecnologica/article/view/627/416>. Acesso em: 20 out. 2022.

SANTOS, Déborados et al. **Técnicas de Inspeção para Diagramas de Classes UML: Uma Revisão Sistemática**, 2019. Disponível em: <https://sol.sbc.org.br/index.php/eres/article/view/8494/8395>. Acesso em: 20 out. 2022.

SELENIUM. **Package org.openqa.selenium**. Disponível em: < https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/package-summary.html>. Acesso em: 20 out. 2022.

TESTSIGMA. **Unified, cloud-based test automation platform, powered By AI.** Disponível em: < https://testsigma.com/>. Acesso em: 20 out. 2022.

W3schools. **Java OOP**. Disponível em: <https://www.w3schools.com/java/java_oop.asp>. Acesso em: 20 out. 2022.