



CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Giulliany Lima Bezerra

APLICAÇÃO DE TAAS (TEST AS A SERVICE) COM CYPRESS

Palmas – TO

2022

Giulliany Lima Bezerra
APLICAÇÃO DE TAAS (*TEST AS A SERVICE*) COM *CYPRESS*

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo.

Palmas – TO

2022

Giulliany Lima Bezerra
APLICAÇÃO DE TAAS (*TEST AS A SERVICE*) COM *CYPRESS*

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo.

Aprovado em: ____/____/____

BANCA EXAMINADORA

Prof. Esp. Fábio Castro Araújo.

Orientador

Centro Universitário Luterano de Palmas – CEULP

Prof. M.e Jackson Gomes de Souza.

Centro Universitário Luterano de Palmas – CEULP

Profa. Fernanda Pereira Gomes.

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2022

RESUMO

BEZERRA, Giulliany Lima. **Aplicação de TaaS (*Test as a Service*) com Cypress**. 2022. 33 f. Projeto Tecnológico (Graduação) - Curso de Engenharia de Software, Centro Universitário Luterano de Palmas, Palmas/TO, 2022.

No processo de aplicação de testes de software tem-se, como objetivo, a redução de defeitos encontrados no desenvolvimento. Com a inserção e utilização de ferramentas de software no dia-a-dia das pessoas com mais frequência, surge ainda mais a necessidade de testes como forma de primar pela qualidade do que está sendo desenvolvido. Devido a essa celeridade no desenvolvimento e a necessidade de cuidar da qualidade, a falta da automação dos testes vai se tornando crítica à medida que o software é desenvolvido. Neste contexto, o presente trabalho consistiu no desenvolvimento de uma estrutura para a aplicação de TaaS (*Test as a Service*) utilizando *Cypress*. Como resultado, foi obtida uma configuração capaz de executar os cenários de teste criado para validação das funcionalidades do SIG Extensão em um ambiente na nuvem, assim como um relatório de resultados obtidos após executar estes cenários de teste.

Palavras chave: Testes; Automação, TaaS; SIG Extensão; *Cypress*.

LISTA DE FIGURAS

Figura 1 - Arquitetura do <i>Cypress</i>	10
Figura 2 - Fluxo do Processo CI	11
Figura 3 - Fluxo do Processo CD	12
Figura 4 - Tela Inicial do SIG Extensão	14
Figura 5 - Fluxo de Trabalho	15
Figura 6 - Versão do Node.js Utilizada	19
Figura 7 - Instalação do <i>Cypress</i>	19
Figura 8 - Pastas do <i>Cypress</i>	20
Figura 9 - Login.spec.js	20
Figura 10 - Login.spec.js	21
Figura 11 - Execução Local	22
Figura 12 - Repositório Gitlab	23
Figura 13 - Arquivo de Configuração CI	24
Figura 14 - Envio ao Repositório	25
Figura 15 - Inicialização da <i>Pipeline</i> no GitLab	25
Figura 16 - Execução da <i>Job</i>	26
Figura 17 - Relatório de Execução da <i>Job</i> Sucesso	27
Figura 18 - Relatório da Execução da <i>Job</i> Sucesso	28
Figura 19 - Relatório da Execução da <i>Job</i> Falha	28
Figura 20 - Relatório da Execução da <i>Job</i> Falha	29

LISTA DE TABELAS

Tabela 1 - Cenário de Teste TC001	17
Tabela 2 - Cenário de Teste TC002	18
Tabela 3 - Cenário de Teste TC003	18

LISTA DE ABREVIATURAS E SIGLAS

TaaS - *Test as a Service*

CEULP/ULBRA - Centro Universitário Luterano de Palmas

API - *Application Programming Interface*

IDE - *Integrated Development Environment*

URL - *Uniform Resource Locator*

CI - *Continuous Integration*

CD - *Continuous Delivery*

SIG - Sistema de Informação e Gestão

Web - *World Wide Web*

SUMÁRIO

1 INTRODUÇÃO	7
2 REFERENCIAL TEÓRICO	9
2.1 TESTES E AUTOMAÇÃO	9
2.2 CYPRESS	10
2.3 INTEGRAÇÃO CONTÍNUA E ENTREGA CONTÍNUA	11
2.3.1 Integração Contínua	11
2.3.2 Entrega Contínua	12
2.4 TESTE COMO SERVIÇO	12
3 METODOLOGIA	14
3.1 MATERIAIS	14
3.2 MÉTODOS	15
4 RESULTADOS	17
4.1 CRIAÇÃO DO PLANO DE TESTE	17
4.2 DESENVOLVIMENTO DA AUTOMAÇÃO DE TESTES	19
4.3 CRIAÇÃO DO REPOSITÓRIO	22
4.4 CONFIGURAÇÃO DO CI NA AUTOMAÇÃO	24
4.5 EXECUÇÃO DA AUTOMAÇÃO EM AMBIENTE DE TERCEIROS	25
4.6 CRIAÇÃO DO RELATÓRIO DE RESULTADOS	27
5 CONSIDERAÇÕES FINAIS	30
6 REFERÊNCIAS	31

1 INTRODUÇÃO

O cenário de desenvolvimento de software atual tem motivado a inserção da tecnologia no dia-a-dia das pessoas com mais frequência. Isso acarreta no desenvolvimento de softwares que possam resolver problemas do cotidiano, com funcionalidades complexas e consequentemente evidencia uma maior necessidade de testes como forma de primar pela qualidade do que está sendo desenvolvido.

Segundo Bartié (2002, p.45), ao passo que o desenvolvimento do software avança e muitas funcionalidades precisam ser testadas, a falta de automação de testes torna-se crítica, tornando-se um dos principais problemas que uma empresa pode encontrar. Tendo em vista que ao realizar uma alteração ou adição de funcionalidade pode ocasionar impactos no sistema de maneira parcial ou como um todo, para realizar uma análise do mesmo de forma totalmente manual, acaba gerando um custo maior de recurso e tempo.

Os testes de validação utilizam a metodologia *end-to-end*, que tem como função comprovar se o que foi construído atende as necessidades para o que lhe foi proposto. De forma a fornecer uma cobertura de teste do início ao fim. Esse processo de validação tem como base o documento de especificação do produto. Podendo ser feito tanto de maneira manual quanto automatizada, de maneira que simule a ação de um usuário real.

Os testes são essenciais durante o desenvolvimento de um *software* (PAULA FILHO, 2019). A realização de testes de maneira totalmente manual se torna uma atividade desgastante à medida que o desenvolvimento de software avança. Além de investir tempo de recursos que poderiam ser utilizados para outras atividades importantes do projeto. “A automação de teste é uma das melhores formas de reduzir o tempo de teste no ciclo de vida do software, diminuindo o custo e aumentando a produtividade do desenvolvimento de software como um todo.” (FANTINATO; CUNHA; DIAS; MIZUNO; CUNHA, 2004, p.219). Uma ferramenta bastante utilizada atualmente é o *Cypress*, um *framework* para a realização de testes automatizados *end-to-end* usando como linguagem de programação o *JavaScript*.

Para auxiliar na automação de testes, a aplicação de TaaS (*Test as a Service*) consiste em realizar atividades como testes em software baseado na web ou *mobile* em uma estrutura em nuvem e as entregar como um serviço para cliente (GAO *et al.* 2013, p.212). Isto é, um modelo de terceirização para executar cenários de testes de validação em um ambiente virtual ou fornecido na nuvem de outras empresas. Sua utilização se torna um facilitador no processo

de testes de software, e segundo Candea *et al.* (2010, p.155) pode “servir como um serviço de certificação disponível publicamente, que permite comparar a confiabilidade e segurança de produtos de software.” Além de auxiliar os desenvolvedores na construção de um produto melhor.

O ambiente terceirizado consiste em uma plataforma fornecida por empresas como *Amazon*, *Microsoft* e *Google*. Nesta plataforma são executadas atividades em que o contratante poderá utilizar os serviços conforme sua necessidade. Sua utilização pode ser feita de qualquer localidade desde que se tenha acesso a uma fonte de internet.

Neste contexto, o presente trabalho visa solucionar o problema de como aplicar TaaS (*Test as a Service*) em um ambiente terceirizado utilizando *Cypress*. Para resolver este problema, foi levantada a hipótese de que é possível aplicar o TaaS em um ambiente terceirizado, utilizando uma estrutura de cenários de testes automatizada em *Cypress*, que possa ser executada na nuvem.

O sistema escolhido para este trabalho foi o SIG extensão, que consiste em apoiar os docentes do CEULP/ULBRA em atividades extracurriculares. O acadêmico realiza o acompanhamento e inscrição em eventos científicos proporcionados pela instituição. Ao final de cada evento o acadêmico obtém um certificado eletrônico.

Deste modo, o trabalho tem como objetivo geral a criação de uma estrutura que permita a execução de testes de validação automatizados das funcionalidades do SIG Extensão. Com a possibilidade de realizar verificação com mais flexibilidade e segurança. Tal expectativa se dá especialmente ao se considerar que uma estrutura como esta é criada objetivando que a realização automática dos testes reduza ao mínimo, ou mesmo impeça, a ocorrência de erros durante a execução dos cenários. Além disso, tem-se como objetivos específicos: Levantar serviços a serem utilizados; Escrever cenários de testes que validem as funcionalidades do SIG Extensão; Criar uma estrutura automatizada com os cenários de testes em *Cypress*; Configurar e executar a estrutura criada em ambientes remotos e avaliar os resultados obtidos.

2 REFERENCIAL TEÓRICO

2.1 TESTES E AUTOMAÇÃO

Segundo Paula Filho (2019, p.709) testes “é uma atividade na qual um produto, sistema ou componente é executado sob condições especificadas, com observação e registro dos resultados e avaliação de um ou mais aspectos.”. De forma mais específica, teste de software é um processo de validação a fim de verificar se o software está se comportando de acordo com o que lhe foi proposto. Para suprir e testar todas as partes do software, o processo de validação é dividido em baixo nível e alto nível.

A etapa de baixo nível contém a fase de testes de unidade. Ela tem como “objetivo testar componentes individuais de uma aplicação” (BARTIÉ, 2002, p.140). A fase de testes de integração tem como “objetivo validar a compatibilidade entre componentes da mesma arquitetura tecnológica” (BARTIÉ, 2002, p.148). A etapa de alto nível contém a fase de testes de sistemas que valida a solução como um todo e a fase de aceitação que trata do último processo formal de detecção de erros no sistema, antes de sua disponibilização no ambiente de produção (Bartié 2002, p.139-157). Após esse processo o sistema é disponibilizado para o cliente e usuários.

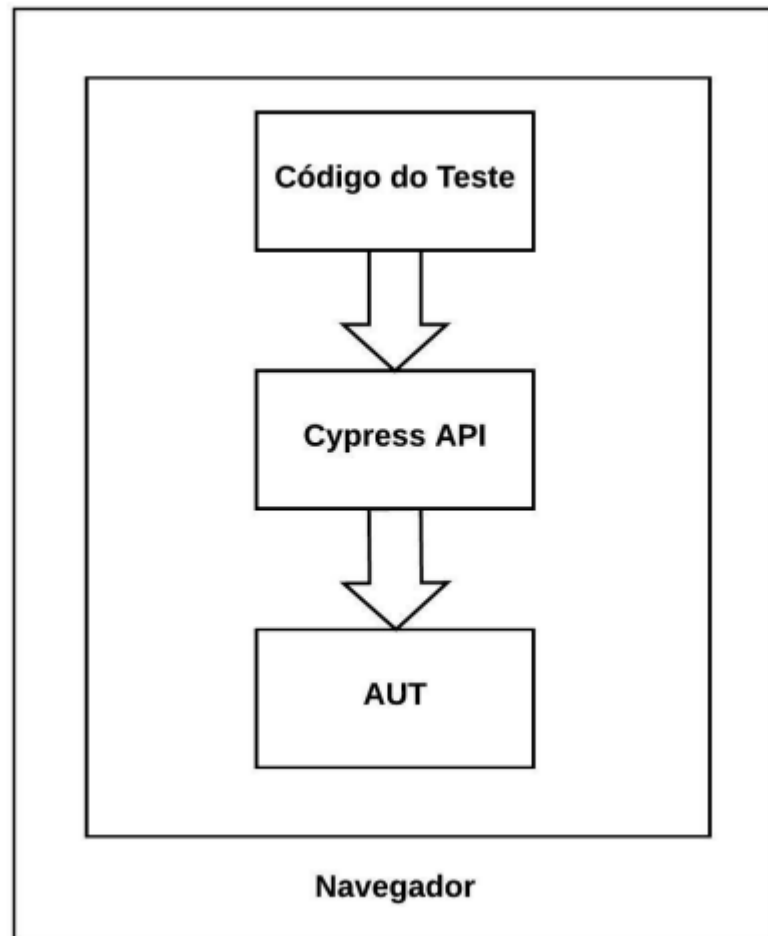
Segundo Siqueira e Farina (2019, p.10), “a automação dos testes de software pode ser definida como o uso de um software para que ele faça o controle da execução destes testes, através da aplicação de estratégias e ferramentas”. Essa automação chega a ser bem semelhante aos testes manuais, isto é, ao executar a automação a máquina irá seguir os comandos descrito no cenário de teste assim como o testador segue a plano de teste, a principal diferença é que a execução será realizada de forma rápida e precisa.

Como aconselhado por Costa (2004, p. 2), “a automação dos processos de testes deve focar naquelas atividades repetitivas, que feitas manualmente são sujeitas a erros, consomem muito tempo e exige dos testadores paciência redobrada para repetir os mesmos passos dos testes e conferir os resultados”. Portanto é necessário que seja realizado previamente um planejamento de como será feito o processo de automação dos testes, para que não haja inconsistências.

2.2 CYPRESS

Cypress consiste em um *framework* para automação de testes *front-end*, fundado por Brian Mann em 2015. O seu desenvolvimento permitiu abordar “os principais pontos problemáticos que os desenvolvedores e engenheiros de controle de qualidade enfrentam ao testar aplicativos modernos” (CYPRESS, 2021). Sua arquitetura é estruturada em três camadas na qual pode ser observado na Figura 1.

FIGURA 1 - ARQUITETURA DO CYPRESS



Fonte: Silva (2019)

Todo o processo é executado dentro do próprio navegador do *Cypress*. A camada de código de teste utiliza a interface do *Cypress API* que, por sua vez, é responsável pela execução dos comandos na aplicação de testes. De acordo com *Cypress* (2021), ele “também opera na camada de rede lendo e alterando o tráfego da web em tempo real”. Sua utilização chega a ser semelhante a do *selenium*, entretanto segundo Mobaraya e Ali (2019) em seu artigo foi observado que o *Cypress* propõe uma melhor desempenho e eficiência em seus testes comparado com o *selenium*, uma vez que para o desenvolvimento dos *scripts* no *Cypress* a nível de linhas de código são relativamente menores.

Cypress utiliza o *JavaScript*, “linguagem de programação leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de *script* para páginas *Web*”(CONTRIBUTORS, 2022). Para realizar a execução dos arquivos de teste o *Cypress* utiliza um *drive* universal, que suporta os tipos de navegadores *Firefox*, *Edge*, *Chrome* e *Chromium*.

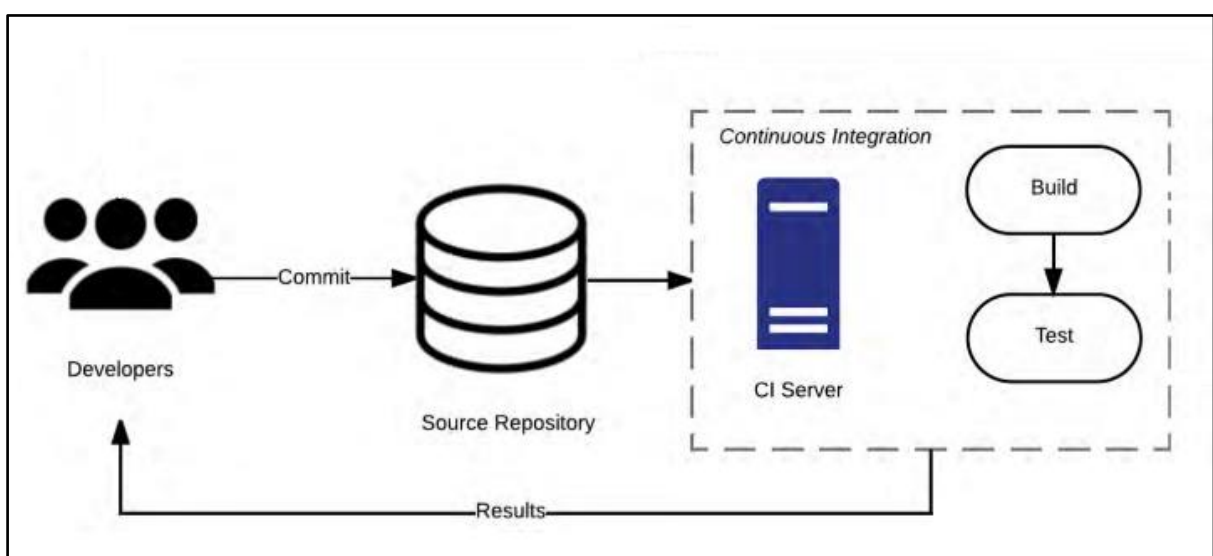
Os cenários de teste escritos devem estar em um arquivo de teste, arquivo este que possui extensão *.spec* para que seja possível a sua execução. Para que um teste seja considerado útil, ele deve validar todas as condições que foram definidas nos cenários e conseqüentemente retornar uma resposta informando se as condições foram atendidas ou não (MWAURA, 2021). No *Cypress* essas verificações são realizadas por meio dos comandos de *exist* e *equal*.

2.3 INTEGRAÇÃO CONTÍNUA E ENTREGA CONTÍNUA

2.3.1 Integração Contínua

A Integração Contínua (*Continuous Integration - CI*) consiste em uma prática de desenvolvimento de software em que os membros da equipe trabalham com frequência na alteração de seus códigos. Certamente com ciclos de lançamentos e melhorias relativamente curtas (SHAHIN; BABAR; ZHU, 2017). Em cada integração é realizado um *build* (versionamento do que está sendo integrado) de forma automatizada, a fim de prevenir e identificar possíveis problemas.

FIGURA 2 - FLUXO DO PROCESSO CI



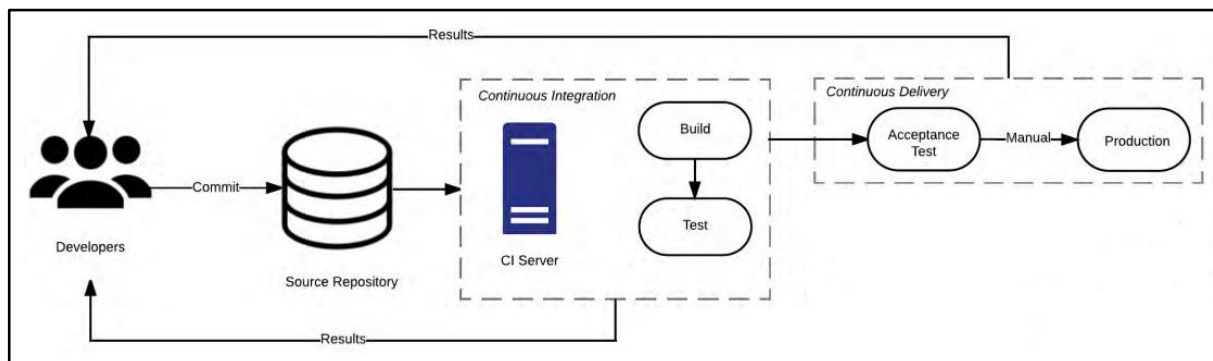
Fonte: Shahin(2017, apud. SUNDMAN, 2013)

Na Figura 2, o processo executado durante a integração contínua é representado. A equipe submete as alterações feitas no código ao servidor central, onde o código será integrado. Após a integração e sincronização do código é executado o *build* e os demais testes.

2.3.2 Entrega Contínua

Segundo Humble e Farley (2014, p.421), a entrega contínua consiste em “um paradigma completamente novo para gerir um negócio que depende de software”. É a forma que você entrega algo, com isso é necessário ter uma participação de todos os envolvidos, seguindo de um suporte da equipe administrativa e do desejo de realizar mudanças no produto. A entrega contínua (*Continuous Delivery* - CD) é uma extensão da integração contínua, seu objetivo é revolucionar a forma de entrega tradicional, possibilitando uma implantação de forma automatizada podendo ser feita por meio de um simples botão (SOUZA NETO, 2018). O fluxo do CD está sendo representado na Figura 3.

FIGURA 3 - FLUXO DO PROCESSO CD



Fonte: Shahin(2017, apud. SUNDMAN, 2013)

Após o processo de execução do build realizado na integração contínua, na CD é realizada a aceitação dos testes, após esse processo de aceitação os novos códigos são enviados para o ambiente de produção.

2.4 TESTE COMO SERVIÇO

O teste como serviço ou TaaS consiste na execução de cenários de testes em uma ambiente na nuvem, oferecendo aos seus usuários uma avaliação dos cenários executados seguida de um relatório dos resultados obtidos (YU *et al.* 2010). A utilização do TaaS pode trazer diversos benefícios para uma empresa. Roodenrijs (2011) apresenta que a utilização do TaaS pode minimizar a necessidade de realização de testes na própria infraestrutura do cliente, além de aliviar a carga de instalação e mantimento da infraestrutura.

Entre as ferramentas que utilizam o serviço de TaaS se encontra o GitLab CI/CD, possuindo eficiência em termos de tempo, menor complexidade administrativa, proporcionando implantação, teste e preparação de contexto, tudo dentro da plataforma (SHARIF; JANTO; LUECKEMEYER, 2020). A busca e utilização de TaaS pelas empresas de tecnologia vem se tornando cada vez maior. Em seu estudo Bertolino, Nautiyal e Malik (2017) apresenta brevemente o interesse das empresas em computação em nuvem, focando em aplicações com potência em testes de software.

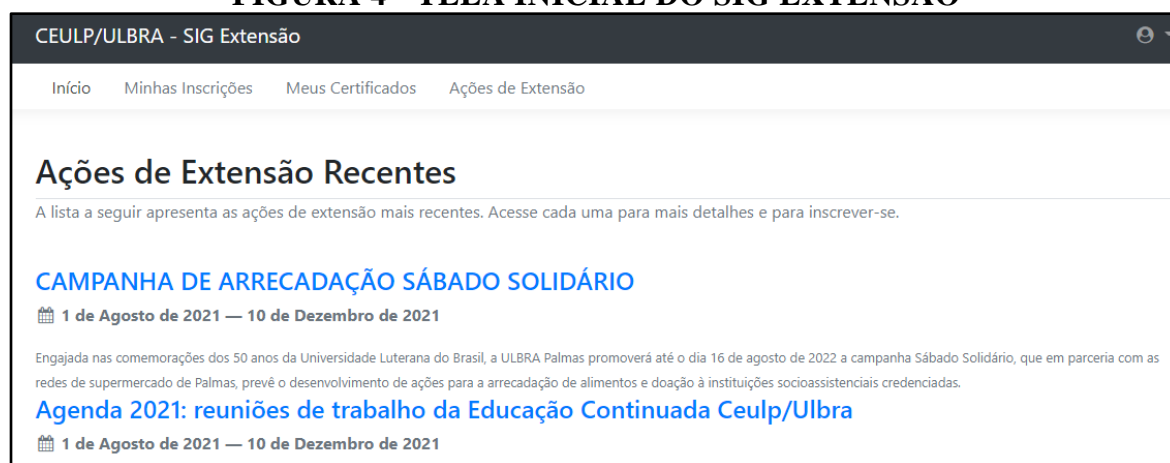
3 METODOLOGIA

3.1 MATERIAIS

Como materiais para o desenvolvimento deste trabalho, foram utilizados o SIG Extensão, *Cypress* e *GitLab*.

As funcionalidades que serão testadas são do SIG Extensão, desenvolvida pela Fábrica de Software do CEULP/ULBRA. Voltada para o acompanhamento e inscrições em atividades de extensão, é possível obter certificados das atividades realizadas pelos acadêmicos. Na visão do acadêmico a plataforma é dividida em quatro funcionalidades, como pode ser observado na Figura 4.

FIGURA 4 - TELA INICIAL DO SIG EXTENSÃO



Fonte: Próprio Autor.

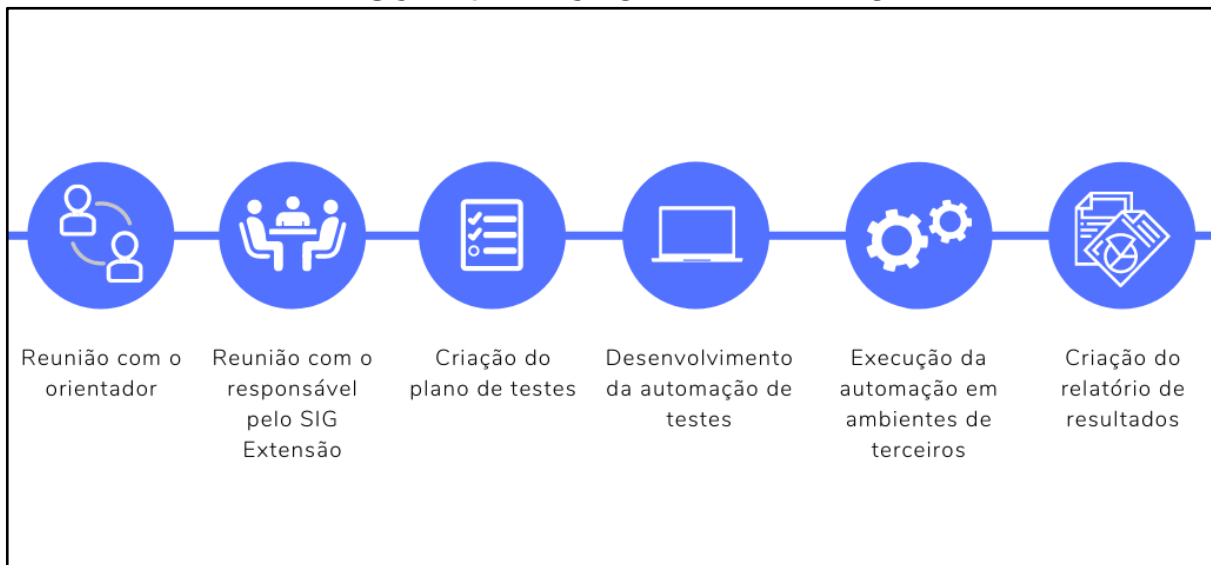
No primeiro item "Início", são apresentadas as ações de extensão mais recentes do CEULP/ULBRA, na qual o acadêmico pode acessar para ter mais detalhes e consequentemente se inscrever. No segundo item "Minhas Inscrições" são apresentadas as ações de extensão na qual o acadêmico se inscreveu, além de possibilitar o acesso aos detalhes da ação inscrita. No terceiro item "Meus Certificados", são apresentados todos os certificados disponíveis para visualização e *download*. Por fim, o quarto item "Ações de Extensão", contém todas as ações de extensão do CEULP/ULBRA na qual o acadêmico poderá efetuar uma busca, acessar mais detalhes e se inscrever.

Na automação dos testes foi utilizado o *Cypress, framework* que permite escrever cenários de testes automatizados para *interface front-end*. Para a execução dos testes em ambiente de terceiros foi utilizado o *GitLab* que consiste em um gerenciador de repositório para a web. Possuindo funcionalidades para realização de integrações contínuas.

3.2 MÉTODOS

Para o desenvolvimento deste trabalho, foi necessário realizar algumas etapas para o alcance dos resultados. Essas etapas são apresentadas na Figura 5, seguidas de uma breve explicação ao longo deste tópico.

FIGURA 5 - FLUXO DE TRABALHO



Fonte: Próprio Autor.

A primeira etapa, reunião com o orientador, teve como propósito discutir onde o trabalho será aplicado, qual o seu objetivo, como será o seu desenvolvimento e quais serão as tecnologias utilizadas. Desta forma, sendo apresentado ao final da reunião a proposta de aplicação de TaaS na plataforma SIG Extensão do CEULP/ULBRA.

A segunda etapa foi com o responsável pela Fábrica de Software do CEULP/ULBRA onde foi desenvolvida a plataforma que é objeto de execução deste trabalho. Nesta reunião foi discutido quais as funcionalidades que a plataforma possui e qual o seu devido propósito. Seguido de uma priorização das funcionalidades que farão parte do escopo deste trabalho.

A terceira etapa, criação do plano de testes, foi realizada a criação do planejamento dos testes, contendo os cenários de teste que irão descrever o passo a passo a ser executado e os retornos esperados ao seguir cada passo.

A quarta etapa, desenvolvimento da automação de testes, foi realizada a automação dos cenários de testes criados na etapa anterior. Para a automação foi utilizado o *framework Cypress* com o uso da linguagem de programação *Javascript*.

A quinta etapa, execução da automação em ambientes de terceiros, após automatizar todos os cenários de testes com *Cypress* e adicionar no repositório do *GitLab*, foi realizada a configuração para execução dos testes em ambiente externo por meio da integração contínua. Aqui foram desenvolvidos os arquivos de configuração necessários para que o ambiente terceirizado execute os testes planejados na etapa anterior.

Ao final, na sexta etapa, foram sumarizados todos os dados obtidos pela a execução da etapa anterior contendo informações dos cenários e execução dos testes.

4 RESULTADOS

Os resultados obtidos com o desenvolvimento deste trabalho são descritos nessa seção. A estrutura de execução segue o seguinte esquema: Criação do plano de teste; Criação do repositório; Desenvolvimento da automação de testes; Configuração do CI na automação; Execução da automação em ambiente de terceiros; e Criação do relatório de resultados.

4.1 CRIAÇÃO DO PLANO DE TESTE

Para criação do plano de teste, primeiramente foi necessário entender o fluxo de ação e resposta esperada. Após ter esse entendimento foram criados três cenários de testes para validação. Dentre os cenários escolhidos com o orientador, estão: Efetuar Login com sucesso, Falha ao efetuar Login e Editar Perfil com sucesso.

Desta forma, tendo em vista que o foco do trabalho consiste em executar os cenários de teste em um ambiente de terceiros, esses cenários foram escolhidos considerando que se trata de cenários bastante comuns e de utilização frequente no sistema, para a nomenclatura dos cenários, foi utilizada a abreviação TC(*Test Case*), seguida de uma identificação numérica e descrição do objetivo do teste. A Tabela 1, representa o primeiro cenário de teste desenvolvido para o trabalho utilizando.

TABELA 1 - CENÁRIO DE TESTE TC001

TC001 - Efetuar login com sucesso	
Acessar a página do SIG Extensão	Apresentar tela inicial
Clicar no botão "Login"	Apresentar tela de Login
Preencher campos com dados válidos e clicar no botão "Entrar"	Apresentar mensagem de êxito no Login e seguir para a próxima tela

Para este cenário foi necessário realizar três passos. Primeiro, acessar a página inicial da plataforma; Como resposta do sistema espera-se que seja apresentada a tela inicial da plataforma. Segundo, realizar a ação de clicar no botão "Login" que se encontra na parte superior à direita; Como resposta do sistema espera-se que seja apresentada a tela de login da plataforma. Terceiro, preencher os campos de usuário e senha com dados válidos e clicar no botão de entrar; Como resposta do sistema espera-se que seja apresentada a tela de usuário autenticado.

Do mesmo modo foi criado um cenário para validar uma falha ao efetuar login. A Tabela 2, representa o segundo cenário de teste desenvolvido para o trabalho.

TABELA 2 - CENÁRIO DE TESTE TC002

TC002 - Falha ao Efetuar login	
Acessar a página do SIG Extensão	Apresentar tela inicial
Clicar no botão "Login"	Apresentar tela de Login
Preencher campos com dados inválidos e clicar no botão "Entrar"	Apresentar mensagem de validação "Por favor, entre com um usuário e senha corretos."

Para este cenário será necessário realizar três passos. Primeiro, acessar a página inicial da plataforma; Como resposta do sistema espera-se que seja apresentada a tela inicial da plataforma. Segundo, realizar a ação de clicar no botão "Login" que se encontra na parte superior a direita; Como resposta do sistema espera-se que seja apresentada a tela de login da plataforma. Terceiro, preencher os campos de usuário e senha com dados inválidos e clicar no botão de entrar; Como resposta do sistema espera-se que seja apresentada a tela de usuário autenticado.

Por último, foi criado um cenário para validar se a edição do perfil do usuário está sendo realizada com sucesso. A Tabela 3, representa o terceiro e último cenário de teste desenvolvido para o trabalho.

TABELA 3 - CENÁRIO DE TESTE TC003

TC003 - Editar perfil com sucesso	
Estando autenticada no SIG Extensão, acesse o menu perfil	Apresentar tela de Editar perfil
Alterar os campos com dados válidos e clicar no botão "Salvar"	Apresentar mensagem de sucesso "As informações do seu perfil foram atualizadas." "

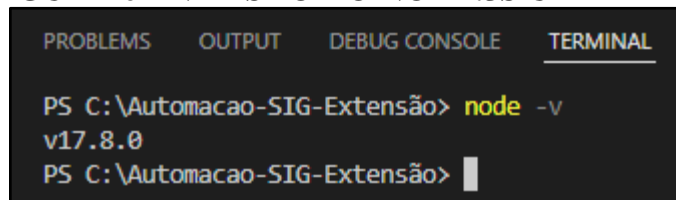
O terceiro cenário, editar perfil com sucesso, está sendo apresentado na Tabela 3. Para este cenário o pré-requisito é estar autenticado na plataforma do SIG Extensão. Após o pré-requisito ser atendido será necessário acessar o menu perfil que se encontra na parte superior a direita da plataforma; Como resposta do sistema espera-se que seja apresentada a tela de edição dos dados de perfil do usuário autenticado. O próximo e último passo é alterar os campos com dados válidos e clicar em salvar; Como resposta do sistema espera-se que seja apresentada uma mensagem de sucesso.

Após ter realizado o planejamento dos cenários de teste, foi necessário criar e estruturar o repositório onde será armazenado os arquivos da automação. Os passos utilizados para sua criação e estruturação estão descritos na próxima sessão.

4.2 DESENVOLVIMENTO DA AUTOMAÇÃO DE TESTES

Para o desenvolvimento da automação dos cenários de teste foi realizada a preparação do ambiente de desenvolvimento. Assim, para realizar a instalação dos pacotes do *Cypress* foi necessário instalar o Node.js com versão igual a 12 ou superior.

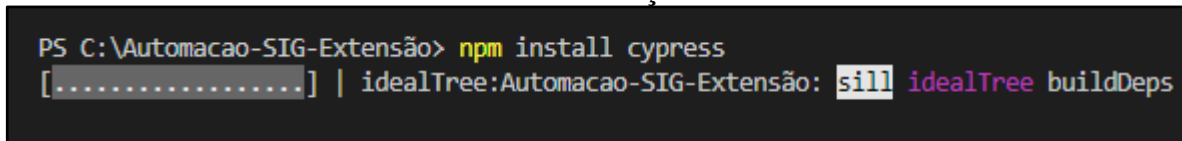
FIGURA 6 - VERSÃO DO NODE.JS UTILIZADA



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Automacao-SIG-Extensão> node -v
v17.8.0
PS C:\Automacao-SIG-Extensão> |
```

Para realizar a verificação da versão do Node.js como apresentado na Figura 6, foi executado o comando \$ **node -v** no terminal. Com a verificação da versão do Node.js, foi realizada a instalação dos pacotes e dependências do *Cypress* como é apresentado na Figura 7, utilizando o comando \$ **npm install cypress**, esse comando realiza a instalação a versão mais atualizada, para este trabalho foi utilizado a versão 9.5.3 do *Cypress*.

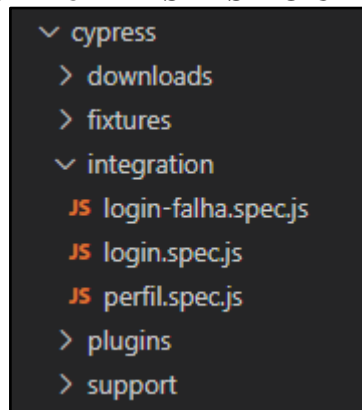
FIGURA 7 - INSTALAÇÃO DO CYPRESS



```
PS C:\Automacao-SIG-Extensão> npm install cypress
[.....] | idealTree:Automacao-SIG-Extensão: sill idealTree buildDeps
```

Após a instalação do *Cypress*, foi utilizado o comando \$ **npx cypress open**, comando esse responsável por inicializar os serviços e geração de pastas do *Cypress*.

Na pasta de *integration*, foram criados dois arquivos de teste para a implementação dos cenários de teste automatizados. No arquivo **login.spec.js** foram descritos os cenários de teste TC001 - Efetuar login com sucesso e TC002 - Falha ao efetuar login. Já no arquivo **perfil.spec.js** foram descritos os cenários de teste TC003 - Editar Perfil com sucesso.

FIGURA 8 - PASTAS DO CYPRESS

Na Figura 9 é apresentado o arquivo de teste **login.spec.js**. Na linha 1 foi realizada a referência do *Cypress*, em seguida na linha 2 a importação do *commands*, arquivo que possui comandos de auxílio. Nas linhas 4 e 5 são declaradas as variáveis que serão utilizadas para o teste. Na linha 7 é informado o contexto, para esse trabalho foi considerada a funcionalidade em que os cenários serão executados. Da linha 8 a 15, são informados os comandos que devem ser executados antes de cada cenário de teste. Sendo um deles o **cy.acesar** que tem como finalidade buscar do arquivo *commands* a função de **acesar** que possui as ações necessárias para o acesso a plataforma SIG Extensão e o **cy.fixture('login')** que busca da pasta *fixture* o *login*, arquivo esse que possui os dados de acesso para serem utilizados nos testes.

FIGURA 9 - LOGIN.SPEC.JS

```

1  /// <reference types="cypress" />
2  import {commands} from "../support/commands";
3
4  let nomeUsuario;
5  let senhaUsuario;
6
7  context('Login', () => {
8    beforeEach(() => {
9
10     cy.acesar()
11
12     cy.fixture('login').then(login => {
13       nomeUsuario = login.username;
14       senhaUsuario = login.password
15     });
16
17   })

```

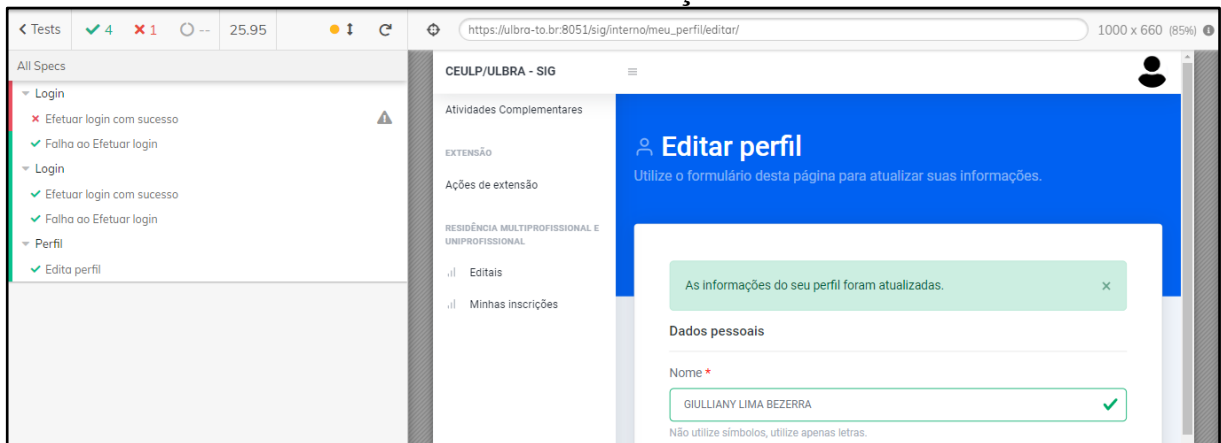
Na Figura 10, na linha 20 a 33, foi descrito o cenário do teste TC001 - Efetuar login sucesso. Assim como o comando **cy.acesar** dentro do **beforeEach** já realiza o acesso a tela

de login, o cenário parte do preenchimento dos campos de login e senha. Na linha 22, é atribuída a variável `login` o elemento do campo login por meio do ID. Na linha 23, é passado o comando `type` para escrever os dados de nome do usuário dentro do campo usuário apresentado na tela de login. Na 25 e 26 foi realizado o mesmo procedimento feito em login para o campo senha. Na linha 28, o comando realiza a captura do botão de entrar, e em seguida realiza a ação de clicar no botão.

FIGURA 10 - LOGIN.SPEC.JS

```
20  it('Efetuar login com sucesso', () => {
21
22      var login = cy.get('#id_username');
23      login.type(nomeUsuario);
24
25      var senha = cy.get('#id_password');
26      senha.type(senhaUsuario);
27
28      cy.get(".btn:contains('Entrar')").click();
29
30      cy.contains('.sidenav-footer-subtitle', /^Você está logado como:/ );
31      cy.contains('.sidenav-footer-title', /^ 125881110 /);
32
33  })
```

A execução dos cenários na máquina local é realizada por meio do comando `$ npx cypress open` como já foi apresentado anteriormente. Ao executar o comando é aberto o navegador de execução do *Cypress*. Em que usuário poderá executar todos os arquivos de teste ou apenas um em específico. A Figura 11, representa a execução de todos os arquivos com seus respectivos contextos e cenários de teste. Do lado esquerdo da imagem é listado os contextos e em seguida os cenários de cada contexto. Do lado direito é apresentado a tela do sistema que está sendo realizado os testes e as ações executadas pelo *Cypress*.

FIGURA 11 - EXECUÇÃO LOCAL

Após realizar todos esses passos, foi realizado o comando de verificação para identificar se na tela está sendo apresentado o texto informando que o usuário se encontra logado no sistema.

4.3 CRIAÇÃO DO REPOSITÓRIO

Para o desenvolvimento deste trabalho foi escolhido o GitLab como repositório. Essa escolha se deu pelo fato do GitLab possuir ferramentas de integração e entrega contínua, além de permitir a integração com outros serviços de nuvem. Na Figura 12 é apresentada a estrutura de pastas e arquivos do repositório criado.

FIGURA 12 - REPOSITÓRIO GITLAB

The screenshot displays the GitLab repository interface for 'sig-automacao-cypress'. The repository is owned by 'Giulliany Lima' and was last updated 15 hours ago. The file list shows the following structure:

Name	Last commit	Last update
📁 cypress A	Envio automacao	15 hours ago
📄 .gitignore B	Envio automacao	15 hours ago
📄 .gitlab-ci.yml C	Envio automacao	15 hours ago
📄 cypress.json D	Envio automacao	15 hours ago
📄 package.json E	Envio automacao	15 hours ago
..		
📁 fixtures F	correções de versão	1 day ago
📁 integration G	Envio automacao	15 hours ago
📁 plugins H	correções de versão	1 day ago
📁 support I	correções de versão	1 day ago

O repositório segue uma estrutura de uma pasta e cinco arquivos. Na pasta *cypress*, se encontra as pastas de *download* (caso seja necessário a utilização de algum arquivo para os testes), *fixtures* em F (onde são armazenados os arquivos .js com dados para utilização nos cenários de teste), *integration* em G (onde são armazenados os arquivos de cenários de teste .spec.js), *plugins* em H e *support* em I. Seguindo a estrutura foi adicionado o arquivo *.gitignore*, onde contém os arquivos a serem ignorados ao realizar o envio do projeto ao repositório. O *.gitlab-ci.yml* (em C) contém a configuração realizada para que os cenários de teste sejam executados na nuvem. No arquivo *cypress.json* e *package.json* estão as configurações, versionamento e URL base do projeto.

4.4 CONFIGURAÇÃO DO CI NA AUTOMAÇÃO

Para realizar a configuração do CI na automação foi necessária a criação do arquivo `.gitlab-ci.yml`. A sua estrutura está sendo apresentada na Figura 13.

FIGURA 13 - ARQUIVO DE CONFIGURAÇÃO CI

```
1 stages:
2   - test
3
4 test:
5   image: cypress/browsers:node12.14.1-chrome85-ff81
6   stage: test
7   script:
8     # Instala as dependências
9     - npm install cypress@9.5.3 ci
10    # Inicia o serviço em background
11    - npm run start:ci &
12    # Executa os cenários de teste no navegador chrome
13    - npx cypress run --browser chrome
```

Na Figura 13, nas linhas 1 e 2 são definidas as *stages* ou estágios de trabalho, que consistem em agrupar os trabalhos a serem executados por parte da entrega contínua. Sua execução é realizada por ordem, onde todos os trabalhos que compõem a *stage* são executados em paralelo. Para este trabalho, foi necessário apenas a *stage* de teste que foi nomeada como “test”. Entre as linhas 4 a 13 são passados os comandos: Instalação de dependências, inicialização do servidor e execução, assim como as necessidades para o estágio de teste que consiste na versão do Cypress a ser utilizada, o navegador que será utilizado e a referência do estágio.

Portanto, na linha 5 foi informada a imagem padrão do *docker* do navegador Chrome. Essa imagem possui um pacote com um sistema operacional e o Chrome instalado. Isso foi necessário pois o *Cypress* é uma plataforma que realiza testes end-to-end, logo ele realiza uma busca por um navegador, com isso o GitLab utilizando o *docker* cria esse ambiente na nuvem para que os cenários sejam executados. Além da imagem é passado também o *script*, na linha 9, para realizar a instalação das dependências para execução dos testes. Na linha 11 foi passado o comando para a inicialização do servidor, tendo em vista que para executar os cenários é necessário preparar a estrutura de execução no ambiente criado na nuvem. Por fim, na linha 13 é passado o comando de execução do cypress no navegador que será utilizado, que é o Chrome.

4.5 EXECUÇÃO DA AUTOMAÇÃO EM AMBIENTE DE TERCEIROS

Uma vez que os cenários de testes estão automatizados e com a configuração do CI realizada, para que a execução destes testes fosse iniciada foi necessário realizar o envio do projeto para o repositório de códigos do Gitlab.

FIGURA 14 - ENVIO AO REPOSITÓRIO

```
PS C:\Automacao-SIG-Extensao> git status
PS C:\Automacao-SIG-Extensao> git add .
PS C:\Automacao-SIG-Extensao> git status
PS C:\Automacao-SIG-Extensao> git commit -m "Envio automacao"
PS C:\Automacao-SIG-Extensao> git push origin teste
```

Para realização deste envio foi necessário executar alguns comandos no terminal. Foi realizado a verificação das pastas e arquivos que foram criados por meio do comando **\$ git status**, realizando essa verificação foi realizado o comando **\$ git add .** comando este responsável por adicionar os arquivos e pastas ao commit que será realizado. Para ter certeza de que todos os arquivos necessários foram adicionados foi executado o comando de verificação novamente. Considerando que todos os arquivos foram adicionados, foi realizado a gravação das alterações por meio do comando **\$ git commit -m "Envio automacao"**. Diante disso foi realizado o envio para o repositório por meio do comando **\$ git push origin teste**.

FIGURA 15 - INICIALIZAÇÃO DA PIPELINE NO GITLAB

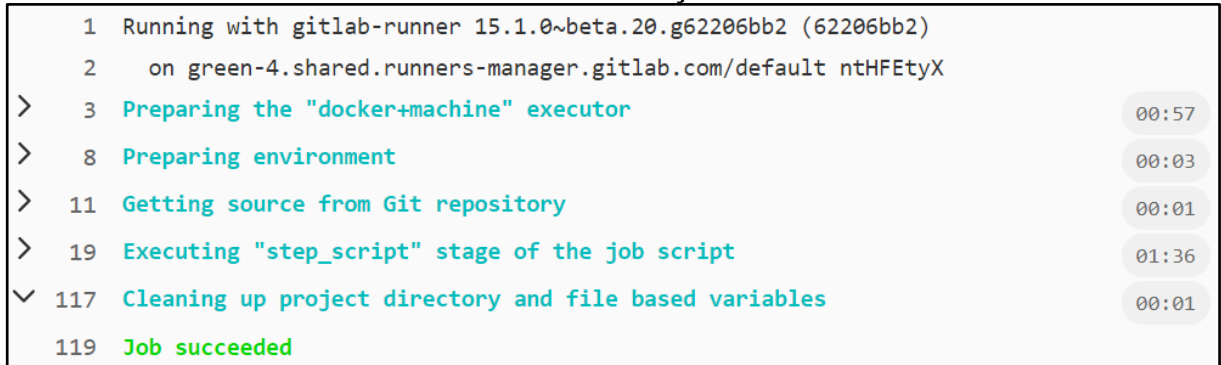
The screenshot shows the GitLab Pipelines page for the project 'SIG-Automacao-Cypress'. The interface includes a navigation bar with 'All 13', 'Finished', 'Branches', and 'Tags'. There are buttons for 'Clear runner caches', 'CI lint', and 'Run pipeline'. A search bar labeled 'Filter pipelines' and a 'Show Pipeline ID' dropdown are also visible. The main content area displays a table of pipelines with columns for Status, Pipeline, Triggerer, and Stages. One pipeline is shown in a 'running' state, with a status icon, a timer showing '00:02:41', and a message '20 minutes ago'. The pipeline name is 'correções de versão', the ID is '#565069328', and the triggerer is 'teste' with commit hash '0d4680cb'. The stages column shows a 'latest' tag and a 'running' status icon.

Status	Pipeline	Triggerer	Stages
running 00:02:41 20 minutes ago	correções de versão #565069328	teste 0d4680cb	latest running

Ao ser enviada as alterações no GitLab, a *pipeline* que corresponde a execução de um conjunto de instruções e etapas passadas no arquivo *.gitlab-ci.yml* é iniciada. No painel são

apresentadas as informações de *status*, tempo de execução, *commit* relacionado e autor. Ao acessar a *pipeline* é possível visualizar as *stages* criadas e suas execuções. Para este trabalho foi criado a *stage* de *test*.

FIGURA 16 - EXECUÇÃO DA JOB



```
1 Running with gitlab-runner 15.1.0~beta.20.g62206bb2 (62206bb2)
2   on green-4.shared.runners-manager.gitlab.com/default ntHFETyX
> 3 Preparing the "docker+machine" executor 00:57
> 8 Preparing environment 00:03
> 11 Getting source from Git repository 00:01
> 19 Executing "step_script" stage of the job script 01:36
✓ 117 Cleaning up project directory and file based variables 00:01
119 Job succeeded
```

Ao acessar a *stage* na Figura 16 é possível observar a *job* de *test* sendo executada. Na linha 1 o GitLab inicia a execução do *commit* realizado, na linha 3 inicia a preparação para a execução do *docker*, estando inicializado, na linha 8 é realizado a preparação do ambiente, na linha 11 é obtido a referência do repositório do projeto do GitLab, logo na linha 19 é instalado a imagem *docker* do *Cypress* e os *scripts* passados no Figura 13 - Arquivo de configuração CI.

4.6 CRIAÇÃO DO RELATÓRIO DE RESULTADOS

Com a execução dos testes no ambiente na nuvem, é gerado um relatório que contém informações de versão *Cypress*, navegador, Node.js utilizada na execução dos testes e a quantidade de arquivos de especificação de testes.

FIGURA 17 - RELATÓRIO DA EXECUÇÃO DA JOB SUCESSO

```

| Cypress:      9.5.3
| Browser:     Chrome 85 (headless)
| Node Version: v12.14.1 (/usr/local/bin/node)
| Specs:       2 found (login.spec.js, perfil.spec.js)

Running: perfil.spec.js (2 of 2)
Perfil
  ✓ Edita perfil (13566ms)
1 passing (14s)
(Results)

| Tests:      1
| Passing:    1
| Failing:    0
| Pending:    0
| Skipped:    0
| Screenshots: 0
| Video:     true
| Duration:   13 seconds
| Spec Ran:   perfil.spec.js

```

No relatório é especificado para cada arquivo de execução de teste o seu contexto. Na Figura 16 o contexto apresentado é o de perfil e dentro desse contexto é apresentado o cenário de teste “editar perfil”. Após a execução de cada cenário de um contexto é informado a quantidade de tempo levado para sua execução, assim como o seu resultado. Para o contexto de perfil, foi executado 1 cenário de teste e 1 cenário de teste passou, levando um tempo de 13s para sua execução como um todo.

FIGURA 18 - RELATÓRIO DA EXECUÇÃO DA JOB SUCESSO

(Run Finished)						
Spec	Tests	Passing	Failing	Pending	Skipped	
✓ login.spec.js	00:14	2	2	-	-	-
✓ perfil.spec.js	00:13	1	1	-	-	-
✓ All specs passed!	00:28	3	3	-	-	-

Após a execução de cada arquivo de teste é gerado um relatório. Esse relatório abrange as informações de quantos cenários de teste foram executados, a quantidade de cenários que passaram e falharam, assim como o tempo gasto para execução por cenário e todos os cenários do contexto do arquivo. Com essa visão a equipe de desenvolvimento poderá identificar e mapear onde as alterações no código impactou, além de monitorar o tempo de performance para a execução dos cenários.

FIGURA 19 - RELATÓRIO DA EXECUÇÃO DA JOB FALHA

```

Running: login-falha.spec.js (1 of 3)
Login
  1) Efetuar login com sucesso
     ✓ Falha ao Efetuar login (6173ms)
  1 passing (22s)
  1 failing
  1) Login
     Efetuar login com sucesso:
       AssertionError: Timed out retrying after 4000ms: Expected to find content: '/^ 125881115 /' within the selector:
'.sidenav-footer-title' but never did.
       at Context.eval (https://ulbra-to.br:8051/_cypress/tests?p=cypress/integration/login-falha.spec.js:119:8)
  (Results)
  Tests:      2
  Passing:    1
  Failing:    1
  Pending:    0
  Skipped:    0
  Screenshots: 1
  Video:      true
  Duration:   21 seconds
  Spec Ran:   login-falha.spec.js
  
```

A Figura 19, representa um cenário de falha proposital, em que o *Cypress* procura determinado elemento durante tanto tempo mas não encontra o elemento passado, assim resultando em uma falha. Para que o usuário tenha ciência de onde se encontra o erro, no relatório é apresentada a descrição de erro de tempo esgotado, o cypress esperou durante

4000ms mas o elemento de texto 125881115 não foi apresentado na tela. A causa do erro se deu pois o texto passado em código para identificação se encontrava divergente do que foi apresentado na tela.

FIGURA 20 - RELATÓRIO DA EXECUÇÃO DA JOB FALHA

(Run Finished)						
Spec	Tests	Passing	Failing	Pending	Skipped	
✘ login-falha.spec.js	00:21	2	1	1	-	-
✔ login.spec.js	00:12	2	2	-	-	-
✔ perfil.spec.js	00:17	1	1	-	-	-
✘ 1 of 3 failed (33%)	00:51	5	4	1	-	-

Ao final do teste o que muda do relatório total da execução sucesso é para a execução com falha é apenas a quantidade da coluna de *failing*, que recebe um incremento a mais no seu valor e em vez de apresentar que todas as especificações foram passadas com sucesso, é apresentado um percentual de falha.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo aplicar TaaS utilizando *Cypress*. Como resultado foi criada uma estrutura que permite a execução de cenários de testes automatizada em *Cypress*, que possibilitou a sua execução na nuvem.

Para chegar nos resultados foi realizado o levantamento de qual sistema seria utilizado para automação dos cenários de teste, o serviço da nuvem que seria utilizado, sendo escolhido o SIG Extensão como sistema e o serviço do GitLab por ser um ambiente que realiza serviços na nuvem, além de ser bastante utilizado para esse tipo de serviço pelos profissionais da área de TI. Com o sistema e o serviço definido foi possível partir para o planejamento dos cenários de testes que seriam utilizados.

Nos cenários de testes foi descrito as ações do usuário e ações de resposta que deve-se esperar da plataforma do SIG Extensão. Tendo os cenários planejados foi possível realizar a automação destes cenários, utilizando como base o que foi planejado. Com isso, foi realizada a instalação do *Cypress* e criado um arquivo de teste(.spec). Nesse arquivo, foi descrito os passos para serem seguidos, desde a identificação de elementos a ações a serem feitas. Com os cenários automatizados e rodando localmente, partiu-se para a configuração do CI, criando um arquivo de configuração(.yml) e passando os comandos de instalação de dependências, inicialização do servidor e execução. Deste modo foi possível executar os cenários de teste automatizados em um ambiente de terceiro do GitLab.

Durante o processo de desenvolvimento deste trabalho, houve a atualização do *Cypress* para a versão 10.0, considerando que a estrutura desenvolvida se encontrava em uma versão inferior, logo quando a aplicação fosse executada no ambiente de terceiro, ela iria se deparar com uma falha de compatibilidade. Deste modo como lição aprendida foi a de que tinha-se a necessidade de estipular uma versão específica para que quando fosse realizar a instalação das dependências no ambiente de terceiros, seja utilizada a versão do projeto ao invés da versão mais atual.

Como trabalhos futuros, propõe-se a criação de CD (*Continuous Delivery*) para dar continuidade ao projeto. Também pode ser realizado a implementação de mais cenários de testes, cenários que possam abranger mais as funcionalidades do SIG Extensão.

6 REFERÊNCIAS

- BARTIÉ, Alexandre (org.). **Garantia da qualidade de software: adquirindo maturidade**. 13. ed. Rio de Janeiro: Elsevier Editora Ltda, 2002. 291 p.
- BERTOLINO, Antonia; NAUTIYAL, Lata; MALIK, Preeti. **Annotated buzzwords and key references for software testing in the cloud**. 2017 International Conference On Computing, Communication And Automation (Iccca), [S.L.], n. 9, p. 893-900, maio 2017. Disponível em: https://www.researchgate.net/publication/322004124_Annotated_buzzwords_and_key_references_for_software_testing_in_the_cloud. Acesso em: 30 jun. 2022.
- CANDEA, George *et al.* **Automated software testing as a service**. In: THE 1ST ACM SYMPOSIUM, 1., 2010, New York, New York, Usa. **Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10**. New York, New York, Usa: Acm Press, 2010. p. 155-160. Disponível em: <https://dl.acm.org/doi/abs/10.1145/1807128.1807153>. Acesso em: 12 out. 2021.
- COSTA, Mozart Guerra. **Estratégia de Automação em Testes: requisitos, arquitetura e acompanhamento de sua implantação**. 2004. 93 f. Dissertação (Mestrado) - Curso de Computação, Instituto de Computação Universidade Estadual de Campinas, Campinas, 2006. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/276289>. Acesso em: 27 out. 2021.
- CYPRESS (org.). **Why Cypress?** 2021. Disponível em: <https://docs.cypress.io/guides/overview/why-cypress#In-a-nutshell>. Acesso em: 25 out. 2021.
- CYPRESS (org.). **Key Differences** 2021. Disponível em: <https://docs.cypress.io/guides/overview/why-cypress#In-a-nutshell>. Acesso em: 25 out. 2021.
- FANTINATO, Marcelo; CUNHA, Adriano C. R. da; DIAS, Sindo V.; MIZUNO, Sueli A.; CUNHA, Cleida A. Q.. **AutoTest – Um Framework Reutilizável para a Automação de Teste Funcional de Software**. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE (SBQS), 3., 2004, Brasília. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2004. p. 219-233.
- GAO, Jerry Z. *et al.* **Testing as a Service (TaaS) on Clouds**. In: SEVENTH INTERNATIONAL SYMPOSIUM ON SERVICE-ORIENTED SYSTEM ENGINEERING, 7., 2013, San Francisco. **Proceedings....** San Francisco: Ieee, 2013. p. 212-223.
- HUMBLE, Jez; FARLEY, David. **Entrega Contínua: como entregar software de forma rápida e confiável**. 9. ed. Porto Alegre - Rs: Bookman, 2014.
- MOBARAYA, Fatini; ALI, Shahid. **Technical Analysis of Selenium and Cypress as Functional Automation Framework for Modern Web Application Testing**. 9Th International Conference On Computer Science, Engineering And Applications (Iccsea 2019),

Nova Zelândia, p. 27-46, 21 dez. 2019. Aircc publishing Corporation.
<http://dx.doi.org/10.5121/csit.2019.91803>.

MWAURA, Waweru. **End-to-End Web Testing with Cypress: explore techniques for automated frontend web testing with cypress and javascript**. Birmingham: Packt Publishing Ltd, 2021.

PAULA FILHO, Wilson de Pádua. **Engenharia de software: produtos**. 4. ed. Rio de Janeiro: Ltc, 2019. 1 v.

ROODENRIJS, Ewald. **Testing Clouds**. Groningen: Sogeti, 2011. 159 p. Disponível em: https://www.capgemini.com/cn-zh/wp-content/uploads/sites/12/2017/07/TMap_NEXT___Testing_Clouds.pdf. Acesso em: 05 dez. 2021.

SELENIUM (org.). **Visão geral**. 2021. Disponível em: <https://www.selenium.dev/documentation/overview/>. Acesso em: 27 out. 2021.

SILVA, Carlos Gabriel Gomes de Melo. **Estudo Comparativo de Ferramentas de Testes de Ponta a Ponta Automatizados em Sistemas Web**. 2019. 56 f. TCC (Graduação) - Curso de Curso de Engenharia de Computação, Departamento de Engenharia de Computação e Automação – Dca, Universidade Federal do Rio Grande do Norte – Ufrn, Natal, 2019. Disponível em: <https://repositorio.ufrn.br/handle/123456789/43656>. Acesso em: 25 out. 2021.

SIQUEIRA, Willian Carlos; FARINA, Renata Mirella. A IMPORTÂNCIA DOS TESTES DE SOFTWARE COMO GARANTIA DE QUALIDADE. **Revista Científica Semana Acadêmica**, Fortaleza, v. 01, n. 000171, p. 2-15, 21 jun. 2019. Disponível em: Engenharia de software produtos by Wilson de Pádua Paula Filho (z-lib.org). Acesso em: 27 out. 2021.

SHAHIN, Mojtaba; BABAR, Muhammad Ali; ZHU, Liming. **Continuous Integration, Delivery and Deployment: a systematic review on approaches, tools, challenges and practices**. **Ieee**, Austrália, v. 6, p. 3909-3943, mar. 2017. Disponível em: <https://ieeexplore.ieee.org/document/7884954>. Acesso em: 02 dez. 2021.

SHARIF, Muddsair; JANTO, Skowronek; LUECKEMEYER, Gero. **COaaS:Continuous Integration and Delivery framework for HPC using Gitlab-Runner**. Proceedings Of The 2020 The 4Th International Conference On Big Data And Internet Of Things, [S.L.], n. 5, p. 54-58, 22 ago. 2020. ACM. <http://dx.doi.org/10.1145/3421537.3421539>. Disponível em: https://dl.acm.org/doi/abs/10.1145/3421537.3421539?casa_token=519Hwmv3zU8AAAAA:C hiwvDLIWQcni3eM1fieYxMW_kG7NmwfGHGpOO2mD178jd0tHwXRIEkyVztNcBvtB2hD2zR3W9mu0yM. Acesso em: 30 jun. 2022.

YU, Lian. *et al.* **Testing as a Service over Cloud**. **Ieee**, Nanjing, China, v. 5, n. 11526938, p. 181-188, set. 2010. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5569908/authors#authors>. Acesso em: 05 dez. 2021.

