**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
**DO RIO GRANDE DO SUL**
**CAMPUS RESTINGA**

# TESTE DE SOFTWARE PARA AUMENTO DA QUALIDADE: TIMESHEET DA SOFTDESIGN

**KEVIN BITENCOURT DA SILVA**

**Porto Alegre**
**2024**

**KEVIN BITENCOURT DA SILVA**

# TESTE DE SOFTWARE PARA AUMENTO DA QUALIDADE: TIMESHEET DA SOFTDESIGN

Trabalho de Conclusão de Curso apresentado, junto ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal Educação, Ciência e Tecnologia do Rio Grande do Sul, como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Rafael Pereira Esteves
Co-orientador: Raphael Rodrigues da Silva

**Porto Alegre**
**2024**

**KEVIN BITENCOURT DA SILVA**

# TESTE DE SOFTWARE PARA AUMENTO DA QUALIDADE: TIMESHEET DA SOFTDESIGN

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Rafael Pereira Esteves
Co-orientador: Raphael Rodrigues da Silva

Aprovado em setembro, 2024.

_____

Rafael Pereira Esteves

_____

Raphael Rodrigues da Silva

_____

Membro da Banca - Professor Gleison Nascimento – IFRS Campus Restinga

_____

Membro da Banca – Professora Gilberto Pavani – IFRS Campus Restinga

Dedico este trabalho a minha filha, que vai nascer em 2025 e a minha esposa que me apoiou em todos os momentos, compreendendo a importância desta conquista para nossa família.

# AGRADECIMENTOS

Agradeço aos meus colegas da SoftDesign pelo apoio, aos meus professores pela atenção e colaboração prestados e por minha família que me motiva a seguir em frente.

*“Não tá morto quem peleia.”*
*(Expressão gaúcha que indica persistência, mesmo diante de dificuldades.)*

# RESUMO

Este trabalho tem como objetivo aprimorar a qualidade e a maturidade do código da ferramenta de controle e apontamento de horas Timesheet da SoftDesign, assegurando que o produto funcione de forma eficiente e eficaz, características essenciais para o sucesso de uma empresa. O Timesheet foi desenvolvido após a pandemia de Covid-19, quando todos os colaboradores precisaram migrar para o home office, e o sistema de ponto da SoftDesign, na época, não atendia a essa nova realidade. Inicialmente, o sistema atendia bem às necessidades, mas com o tempo, novas funcionalidades foram adicionadas e o número de colaboradores aumentou. A combinação desses fatores resultou em problemas como travamentos, lentidão e até perda de dados. Diante disso, surgiu a necessidade de identificar e tratar as causas desses problemas. O objetivo é realizar testes de software na ferramenta para identificar falhas no código, utilizando a ferramenta de testes automatizados Cypress, recomendada pela SoftDesign. O intuito é dar início a um processo contínuo de melhorias, a ser executado futuramente pelos QA Engineers disponíveis.



**Palavra-chave:** qualidade, automação de testes, SoftDesign, Cypress, Timesheet.

## *ABSTRACT*

*This work aims to improve the quality and maturity of the code of SoftDesign's Timesheet timekeeping and control tool, ensuring that the product works efficiently and effectively, essential characteristics for the success of a company. Timesheet was developed after the Covid-19 pandemic, when all employees needed to migrate to home office, and SoftDesign's time system, at the time, did not meet this new reality. Initially, the system met the needs well, but over time, new features were added and the number of employees increased. The combination of these factors resulted in problems such as crashes, slowdowns and even data loss. Therefore, there was a need to identify and treat the causes of these problems. The objective is to perform software tests on the tool to identify errors in the code, using the Cypress automated testing tool, recommended by SoftDesign. The aim is to begin a continuous process of improvements, to be carried out in the future by the available QA Engineers.*



***Keyword:*** *quality, test automation, SoftDesign, Cypress, Timesheet.*

# LISTA DE TABELAS

# SUMÁRIO

# 1. INTRODUÇÃO

Softwares e sistemas estão presentes no dia a dia de todas as pessoas atualmente e se fazem essenciais pela praticidade. Estão presentes nas atividades mais simples, na compra do pão para o café da manhã, por exemplo, temos o uso de pelos menos 3 softwares, na balança, do caixa e no pagamento se não for em espécie. Para um software ser comercializado é necessário assegurar o funcionamento correto do sistema. Para isso é realizado uma série de testes em todas as funcionalidades. Os testes de software foram desenvolvidos para analisar se o comportamento do programa está conforme o esperado. Guilherme Wergutz Muller (2020) fala que "o teste de software é uma maneira de avaliar a qualidade da aplicação e reduzir o risco de falha em operação. Testar não consiste apenas em executar testes (executar o software e verificar os resultados). Executar testes é apenas umas das atividades. Planejamento, análise, modelagem e implementação dos testes, relatórios de progresso, resultado e avaliação da qualidade, também são partes de um processo de testes.", para isso, é possível realizar testes de duas maneiras: manualmente – o qual é realizado por uma pessoa analista de testes, e de forma automatizada – em que são utilizados *frameworks* para gerar *scripts* simulando um usuário interagindo com o sistema ou através da simulação do comportamento da API (Lorena Garcia, 2024). Atualmente testes manuais são de extrema importância para verificação da usabilidade, acessibilidade e responsividade. Assim, muitas vezes, é necessária a automação para auxiliar as atividades de testes manuais, para evitar que tarefas repetitivas sejam feitas, resultando em economia de tempo e como consequência a redução de custos. (Rony Lourenço, 2022).

## 1.1 Problema

A SoftDesign é uma empresa multinacional com sede na cidade de Porto Alegre no Rio Grande do Sul, é uma empresa prestadora de consultoria e desenvolvedora de sistemas para clientes dos mais variados nichos, conta atualmente com aproximadamente 200 colaboradores em diversos lugares do país e afora, e isso gerou a demanda de se criar um sistema de controle de horas para ser realizado a bilhetagem aos clientes que pagam por hora de serviços prestados e controle de trabalho dos colaboradores para com os projetos dos clientes. Conforme Karina Hartmann Head of innovation & Continuous Improvement na SoftDesign, "O desenvolvimento do Timesheet ocorreu por uma confluência de oportunidades. Por um lado, precisávamos de

um sistema melhor e mais moderno para o controle das horas. Por outro, estávamos buscando um desafio realista para a turma de trainees de 2019, ano em que formamos uma turma de 10 trainees. Usamos uma abordagem de aprendizado baseado em projeto. Primeiro os trainees passam por 2 meses de treinamento. Depois, trabalham em um projeto piloto com supervisão de mentores." por ser criado pela SoftDesign esteve em constante evolução até final de 2023 e início de 2024 quando houve um aumento no número de clientes e todos os times de desenvolvimento foram alocados para clientes e o sistema começou a apresentar alguns problemas tais como lentidão, travamentos e o mais grave: Tendo sido uma aplicação desenvolvida inicialmente no contexto da formação de estagiários, ela não foi pensada para escalabilidade ou resiliência. Por isso, nos períodos de pico de uso é comum a aplicação apresentar lentidão e instabilidade. Também há alguns incidentes recorrentes relativos a regras de negócio.

Mesmo já tendo outros profissionais ajustando os incidentes e melhorando o código gerado, ainda gera instabilidades e insegurança na criação de novas funcionalidades no Timesheet.

### 1.2 Objetivo geral e específico

Este trabalho tem como objetivo geral realizar testes automatizados na ferramenta Timesheet de forma a aumentar a qualidade e segurança do comportamento correto da aplicação. Os seguintes objetivos específicos foram desenvolvidos para este trabalho:

- Realizar uma análise dos casos em que ocorrem os problemas.
- Realizar um plano de testes automatizados.
- Gerar uma base de automação de testes usando a ferramenta Cypress, para ser usado futuramente por equipes que venham trabalhar na ferramenta Timesheet.
- Entregar o projeto em um ambiente de CI/CD**.**
- Realizar a passagem de conhecimento para que a equipe do Timesheet dê continuidade ao projeto de testes automatizados.

### 1.3 Estrutura do Documento

O capítulo 2 apresenta a fundamentação teórica do trabalho, mostrando em conceitos teste, testes automatizados, Cypress e Timesheet.

O capítulo 3 trará trabalhos relacionados a este e mostrará as diferenças que este trabalho possui dentre os outros selecionados.

O capítulo 4 apresenta a metodologia utilizada no trabalho, mostrando os processos e ambientes que ficará condicionado para a realização deste trabalho, além de mostrar os testes utilizando a ferramenta Cypress no Timesheet.

O capítulo 5 mostra a finalização junto as evidências com os casos de erros encontrados durante os testes.

Por último, o capítulo 6 mostra a conclusão deste trabalho com as considerações de representantes da SoftDesign relatando o impacto que este trabalho trouxe para o aumento da qualidade da ferramenta Timesheet.

# 2. FUNDAMENTAÇÃO TEÓRICA

## 2.1. Teste de Software:

O teste do software nada mais é do que a investigação do software a fim de identificar e coletar informações sobre sua qualidade em relação ao contexto em que ele deve operar, se relaciona com o conceito de verificação e validação. O que, na prática, é ligado exclusivamente ao uso da aplicação.

Uma aplicação foi criada para cumprir um objetivo, inicia-se então o planejamento do teste, levando em consideração todos os aspectos da empresa que podem de alguma forma interferir nos resultados dos testes, pôr fim a execução.

Há uma concordância generalizada acerca da imprescindibilidade do controle de qualidade no desenvolvimento de software. Atrasos na entrega ou a presença de defeitos podem comprometer a reputação de uma marca, resultando em clientes insatisfeitos e perdidos. Em situações extremas, um erro ou falha no sistema pode afetar a integridade de sistemas interconectados, ocasionando prejuízos significativos no seu funcionamento.(Objective, 2022)

Considere uma montadora de veículos multinacional de grande porte tendo que fazer um recall de mais de um milhão de carros devido a um defeito de software nos detectores dos sensores dos airbags ou um bug de software que causou a falha de um lançamento de satélite militar de bilhões de Dólares, os números falam por si. Falhas de software nos EUA custaram à economia US$ 1,1 trilhão em ativos em 2016. Além disso, eles afetaram 4,4 bilhões de clientes. (IBM, 2024)

Embora os testes em si custem dinheiro, as empresas podem economizar milhões por ano em desenvolvimento e suporte se tiverem uma boa técnica de testes e processos de controle de qualidade em vigor. Os primeiros testes de software descobrem problemas antes que um produto seja lançado no mercado. Quanto mais cedo as equipes de desenvolvimento receberem feedback dos testes, mais cedo poderão resolver problemas. (IBM, 2024)

Sabendo da importância dos testes de software, temos a problemática de testar várias vezes a mesma aplicação, sempre que ela é alterada. Quando testamos manualmente o teste se torna muito caro e a médio e longo prazo a confiança do teste é reduzida, já que a pessoa realizando o teste tende a ter menor atenção aos detalhes conforme repete muitas vezes o mesmo cenário. Para resolver essas demandas foram criadas ferramentas para automatizar os testes, garantindo maior confiabilidade do teste e redução do custo no longo prazo.

## 2.2. Pirâmide de Testes:

É uma representação visual elaborada para simplificar os testes de software, seus níveis, velocidade e complexidade.

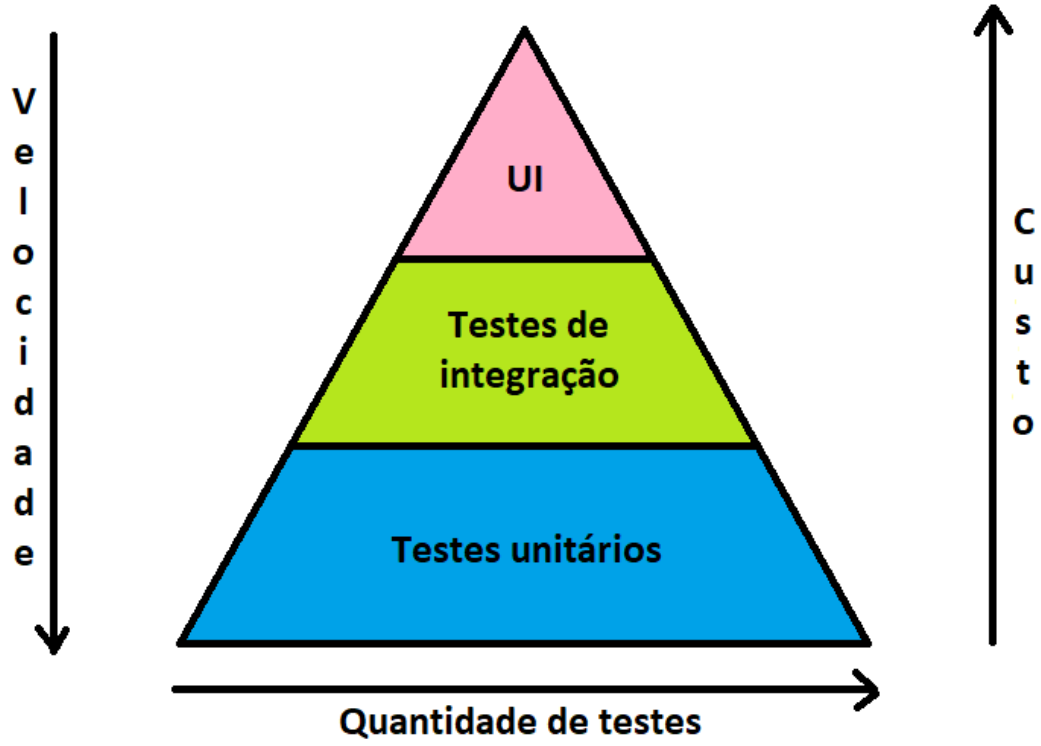


*Figura 1, Pirâmide de testes https://www.alura.com.br/artigos/assets/por-que-e-o-que-e-possivel-testar/imagem1.png*

**Base: Testes de Unidade:**

Na base da pirâmide de testes estão os Testes de Unidade, também conhecidos como testes unitários, esses testes são realizados na menor parte testável de uma aplicação, independentemente de sua interação com outras partes do código. Em um contexto de programação orientada a objetos, uma unidade pode ser classificada como um método público disponível em uma determinada classe, devido ao foco em testar os componentes mais granulares do código de forma isolada, os testes unitários tendem a ser muito pequenos e rápidos de criar e executar. Essa abordagem permite identificar com precisão o local de uma falha caso um dos testes não passe, para viabilizar esse tipo de teste, é comum criar objetos falsos, também conhecidos como *fakes, mocks, stubs* e similares, que imitam o comportamento de objetos reais, dessa forma, não é necessário que outras unidades sejam reais para que o teste seja executado.

É possível criar esses objetos falsos para simular seu comportamento e testar a unidade de forma isolada. (Marcos Rangel, 2024)

**Meio: Testes de Integração:**

De maneira prática os testes de integração verificam caminhos de comunicação externa e as interações entre os componentes para detectar defeitos na interface. Validam a integração com módulos externos, validam conexões com banco de dados, nesse caso geralmente se sobe o banco em memória e simula conexão. (Nathalya Muniz, 2020)

Conforme publicado no blog "casa do desenvolvedor, 2024" Os testes de integração são uma prática crucial no desenvolvimento de software, concentrando-se na validação da interação eficaz entre diferentes módulos ou componentes de uma aplicação. Essa fase de teste visa garantir que as partes individualmente funcionem corretamente quando integradas, evitando falhas e conflitos no sistema como um todo. Assim, ao examinar a interface e a colaboração entre os elementos, os testes de integração proporcionam uma visão abrangente do desempenho da aplicação. Ademais, identificam potenciais problemas de interoperabilidade que podem surgir na fase de implementação, estabilidade e confiabilidade do software.

**Topo: Testes UI (ou Testes de Interface):**

Os testes de interface são uma parte fundamental do desenvolvimento de software, uma vez que garantem que as diferentes partes do sistema se comuniquem corretamente. (Armando Couto, 2023) Testa funcionalidades do sistema afim de que eventos externos não os modifiquem.

> **Usabilidade:** Focado na experiência do usuário, consistência da interface, layout, acesso às funcionalidades etc.
>
> **Acessibilidade:** Subconjunto de usabilidade, visa atender pessoas de diferentes habilidades: deficiência visual, física, auditiva, comprometimento cognitivo, dificuldade de aprendizagem entre outros. (Nathalya Muniz, 2020)

Usado para determinar completamente a eficácia e a usabilidade geral de uma interface do usuário de aplicativo, ela deve ser testada. O teste expõe o quão fácil ou difícil é usar a interface do usuário para o público mais amplo possível. O tempo que leva para testar um aplicativo vale muito a pena. (Microsoft, 2023)

De acordo com a pirâmide, a quantidade de testes que desenvolvemos está diretamente ligada à velocidade e ao custo deles. Isso porque testes unitários, por exemplo, têm a realização mais barata, logo, uma maior quantidade deles pode ser feita e eles compõem a nossa base. Já os testes de interface (ou UI de User Interface) são mais custosos e lentos de realizar, o que os torna menos viáveis. Então, serão feitos em menor quantidade. Todas as etapas de produção de um software devem ser testadas. Não é um trabalho simples, demanda tempo e é custoso. Entretanto, não o fazer pode acarretar problemas muito maiores. (Larissa Gabriela, 2021)

## 2.3. AUTOMATIZAÇÃO DE TESTES:

O teste automatizado é a aplicação de ferramentas de software para automatizar um processo manual conduzido por humanos de revisão e validação de um produto de software. Quanto maior a qualidade dos scripts de testes melhor é a qualidade do método de testes, existem dezenas de softwares de testes, porém para se ter bons resultados deve-se criar o máximo de caso de testes para a determinada aplicação. Além de evitar retrabalho a automação de testes traz muitos benefícios:

**Escalabilidade:**

Com a automação de testes, nós conseguimos manipular nossas verificações para atender praticamente qualquer número de alterações. Também podemos escalar a execução dos testes, pois com os devidos cuidados (automações e gerência de configuração e dados), os testes se tornam portáveis para praticamente qualquer ambiente.

Também podemos contar com a possibilidade de paralelismo. Em muitos cenários, é possível executar testes em paralelo, diminuindo consideravelmente o tempo dispensado para executar uma bateria completa de testes. (Qualidade de Software #2: As vantagens da automação de testes, 2018)

**Confiabilidade:**

Com um ambiente devidamente controlado e uma gerência de configuração e de dados adequada, podemos dizer que os testes sempre são executados da mesma maneira. E isso os torna naturalmente confiáveis, ainda mais se comparados a procedimentos manuais.

Se com testes manuais nós dependemos da atenção e conhecimento de uma pessoa, com testes automatizados nós temos tudo isso codificado.

Com o devido processo de revisão do código dos testes e a determinação adequada do procedimento de verificação, podemos confiar que o software foi devidamente coberto e os resultados representam o certo ou errado absoluto. (Qualidade de Software #2: As vantagens da automação de testes, 2018)

**Ciclos de feedback mais rápidos:**

No começo da aplicação de teste era quase que normal as aplicações chegarem ao usuário final com bugs, falhas e erros, e cabia ao usuário final retornar o feedback para realizar correções e esse processo poderia levar dias até mesmo semanas para ser concluído, mas com a automatização esse processo é muito raro de acorrer, pois a maioria dos testes é feito desde a concepção até a entrega para garantir uma aplicação com um percentual próximo a zero de falhas.

**Redução de custos**

A automação de teste exige um esforço inicial como qualquer outra atividade dentro do desenvolvimento de software. Mas, esse esforço é logo compensando. As reduções de custo se apresentam na forma de:

- maior facilidade na obtenção de feedback dos testes - promove uma execução facilitada, pois possibilita diversos meios de disparo (desde botões em interfaces de ferramentas até uma simples chamada de comando em um terminal). E torna possível uma análise de resultados proativa, dado que muitas ferramentas oferecem relatórios consolidados e métodos de disparo para outras tarefas automatizadas.
- menor investimento financeiro em pessoas executando tarefas repetitivas - as pessoas deveriam ser pagas para realizar tarefas de alto valor, que exijam capacidades cognitivas e raciocínio que só humanos possuem (coisas que máquinas não podem fazer). É um grande desperdício manter pessoas fazendo tarefas repetitivas e enfadonhas, que aumentam a chance de falhas decorrentes naturalmente de esforços repetidos.

- escalabilidade para execução a cada alteração - escala, tanto em pessoas, quanto em tempo, pois possibilita menos pessoas envolvidas com o processo de execução de testes e promove um tempo menor em etapa de setup. Isso permite executar os testes com uma frequência muito maior, se comparado com uma abordagem manual.

- diminuição do retrabalho - com verificações podendo ser executadas continuamente, um menor número de defeitos chega até a entrega final do software, diminuindo consideravelmente o custo em retrabalhos.

O paper End-to-end test automation – A behavior-driven and tool-agnostic approach, publicado pela *Infosys*, traz uma abordagem que busca reduzir a complexidade da automação de testes e torná-la ainda mais útil na promoção de feedback contínuo. Com essa abordagem, eles chegam a citar reduções entre 40% e 60% no custo de geração de casos de teste automatizados, em comparação com testes manuais. (Qualidade de Software #2: As vantagens da automação de testes, 2018)

## 2.4. Cypress

Segundo a documentação o Cypress é um *framework Javascript* para aplicações web voltado para testes *frontend* automatizados. Nele é possível escrever, executar e depurar os testes através de mensagens legíveis, fornecendo assim para os testadores um ambiente de automação completo (CYPRESS, 2022).

Enquanto ferramentas como o Selenium operam executando fora do navegador, o Cypress é exatamente o oposto: ele é executado no mesmo loop de execução que a aplicação, permitindo acesso a comandos nativos. O *framework* é executado sobre o Node, ele se comunica, sincroniza e executa tarefas tornando a experiência da escrita e execução dos testes muito mais ágil. (CYPRESS, 2022).

Para a SoftDesign a escolha da ferramenta certa para testes automatizados é fundamental para ganhar em agilidade e eficiência operacional. O Cypress se destaca nessa função pelos seguintes os motivos:

- Cypress possui instalação simples e uma curva de aprendizado mais leve, permitindo que, mesmo Desenvolvedores iniciantes, possam começar a escrever testes rapidamente;

- Graças à sua arquitetura e execução no mesmo ambiente do navegador, os testes são extremamente rápidos;

- Sua interface interativa facilita a visualização e a depuração de testes, permitindo que os Desenvolvedores identifiquem e corrijam problemas rapidamente;

- Os recursos de simular e testar interações de usuário garantem que todas as partes da aplicação funcionem corretamente;

- Cypress possui uma comunidade forte e com documentação detalhada, e isso facilita a busca por recursos e suporte para tirar dúvidas, como os que você vai encontrar ao longo deste conteúdo! (Cypress: passo a passo para começar a usar, 2021).

Rafael Berçam (Medium.com, 2019) nos mostra um exemplo prático do uso do Cypress de maneira bem simplificada, nas figuras 1-6 abaixo ele inspeciona elementos da página web para identificar os elementos a serem usados para os testes.

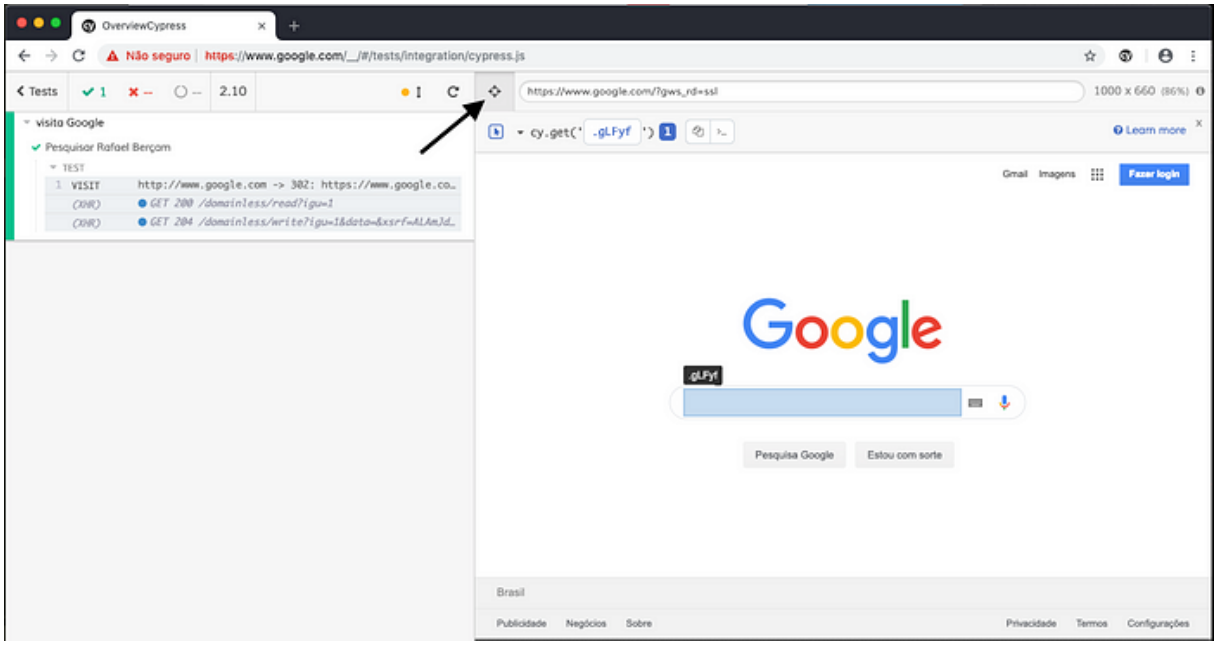


*Figura 2, Fonte:https://medium.com/@faelbercam/um-overview-sobre-cypress-io-framework-de-automa%C3%A7%C3%A3o-de-testes-end-to-end-dc438b9ee7a1*

Na figura 3 tela ele pega o identificador ("id") do campo no qual ele deseja interagir.

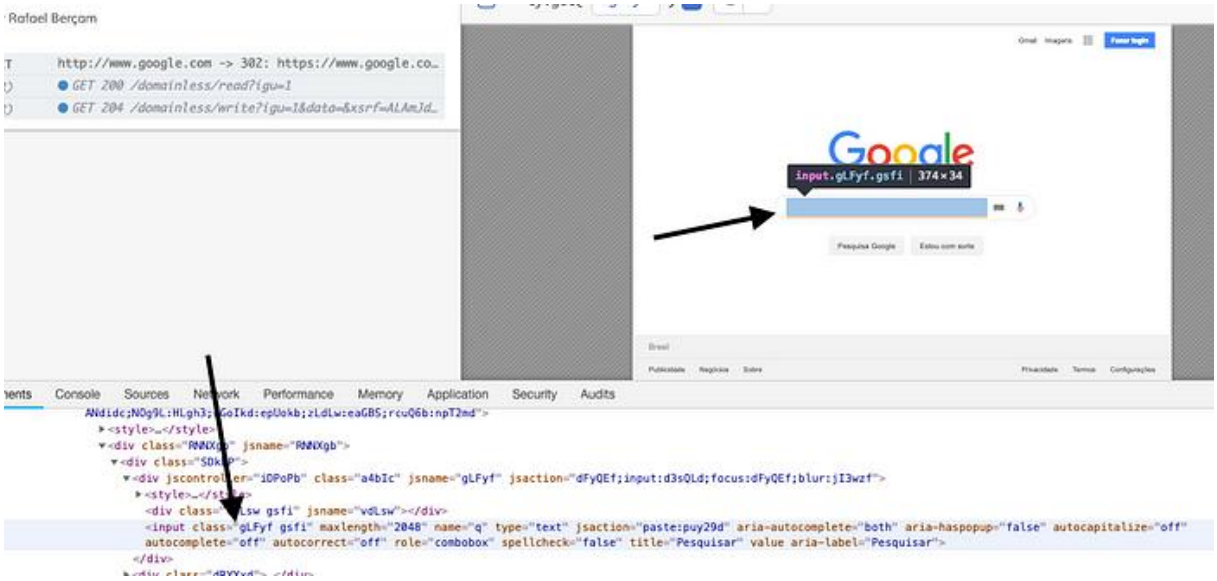


*Figura 3, Fonte:https://medium.com/@faelbercam/um-overview-sobre-cypress-io-framework-de-automa%C3%A7%C3%A3o-de-testes-end-to-end-dc438b9ee7a1*

Com os campos devidamente identificados, basta atribuir os valores a serem preenchidos no código do Cypress, na figura 4 em questão o objetivo é fazer uma pesquisa pelo nome do autor no buscador do Google.

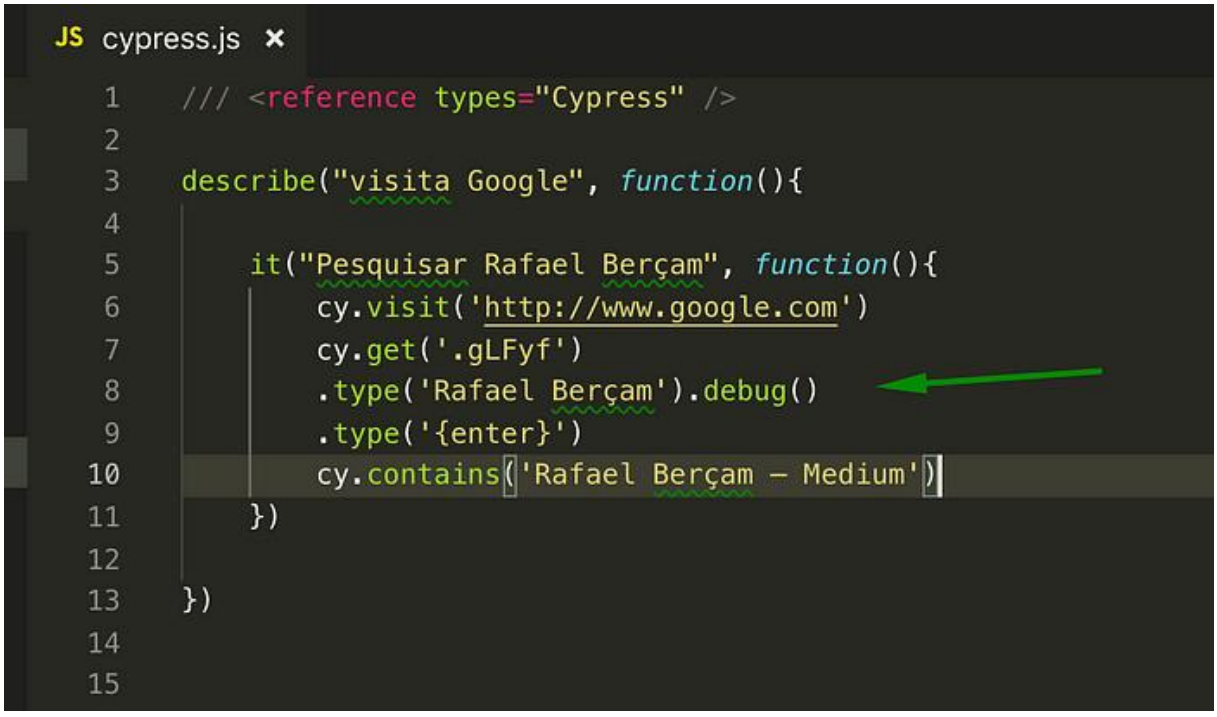


*Figura 4, Fonte:https://medium.com/@faelbercam/um-overview-sobre-cypress-io-framework-de-automa%C3%A7%C3%A3o-de-testes-end-to-end-dc438b9ee7a1*

A ferramenta de *Debugger* do Chrome para exatamente quando digita o nome do campo de busca do Google e a partir dali você consegue depurar todo os passos da aplicação e acompanhar

o processo, e ao final ele exibe no console o resultado conforme mostrado na figura 5:

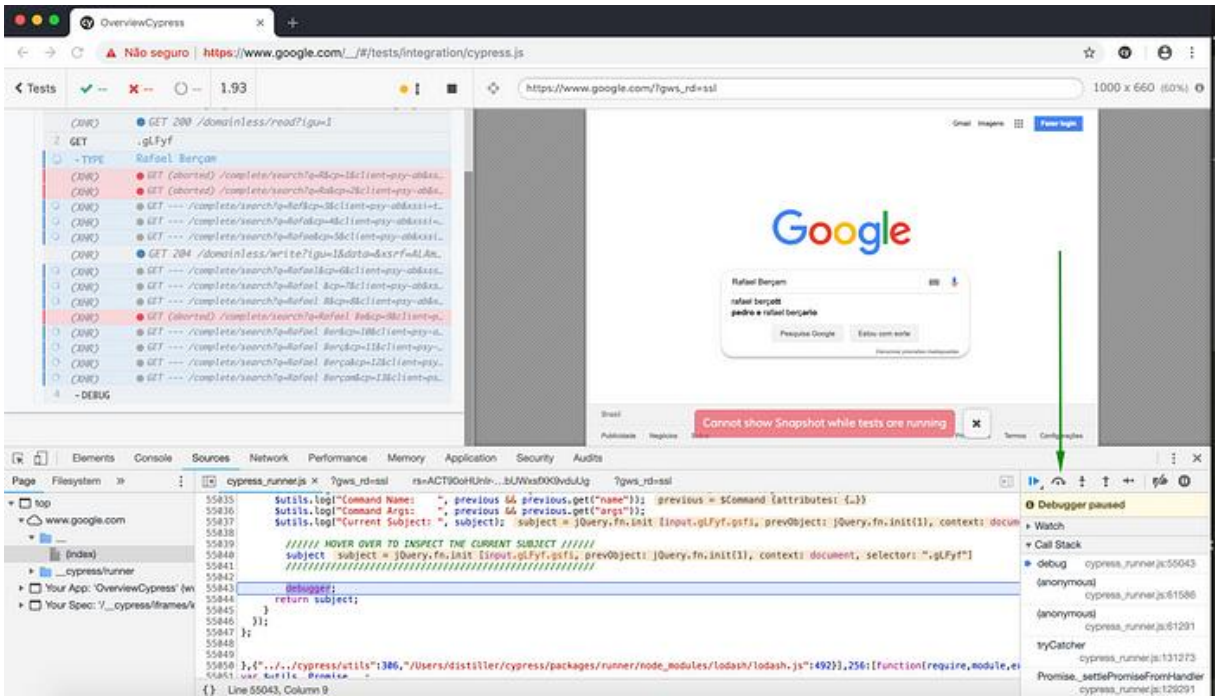


*Figura 5, Fonte:https://medium.com/@faelbercam/um-overview-sobre-cypress-io-framework-de-automa%C3%A7%C3%A3o-de-testes-end-to-end-dc438b9ee7a1*

A figura 6 mostra o console onde exibe o nome do comando debugado, os argumentos enviados e qual elemento foi explorado no caso que foi a classe do campo de busca do Google.

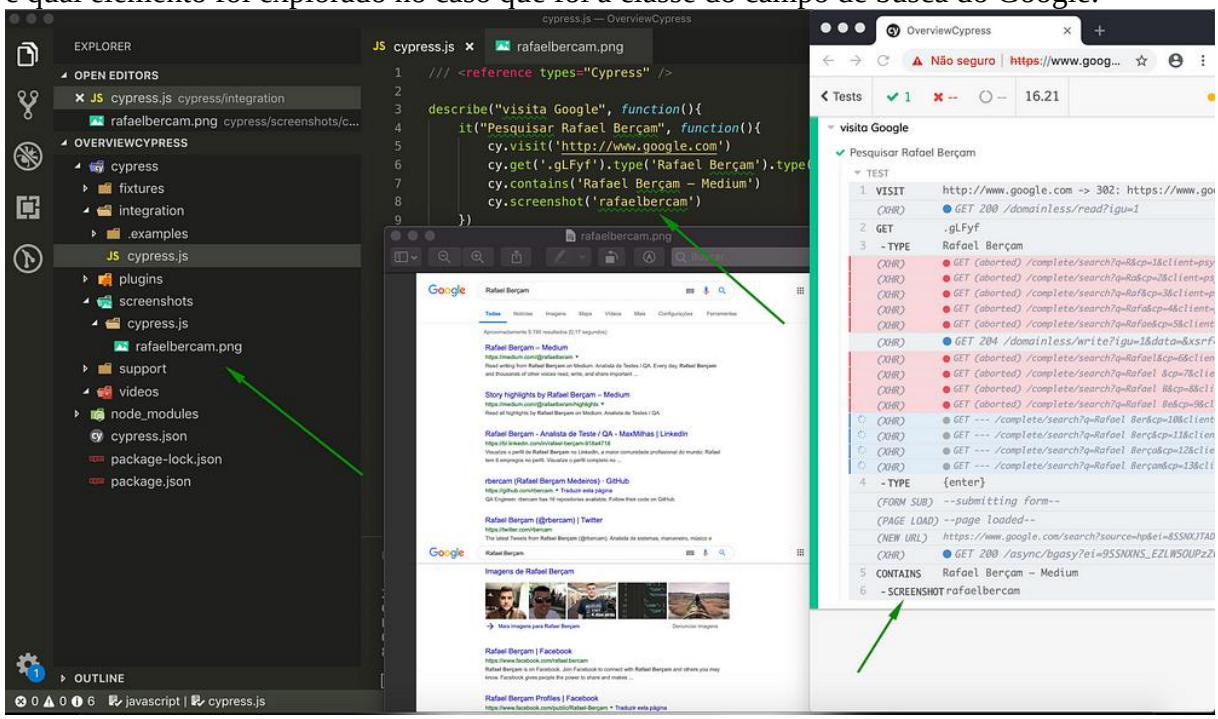


*Figura 6, Fonte:https://medium.com/@faelbercam/um-overview-sobre-cypress-io-framework-de-automa%C3%A7%C3%A3o-de-testes-end-to-end-dc438b9ee7a1*

E o resultado desse exemplo é a identificação do poder da ferramenta Cypress e o desempenho que pode ser extraído, ficando muito claro que a facilidade de trabalhar com esse *framework* é muito grande, e não precisa de fazer tantas configurações para obter um teste robusto e monitorado.

## 2.5. Sistema Timesheet:

O sistema Timesheet foi desenvolvido para atender uma demanda interna da SoftDesign, inicialmente pensado para ser um sistema de controle de horas para registro do período trabalhado. Criado durante a pandemia onde os sistemas ponto existentes na soft ficaram obsoletos devido à nova realidade do trabalho home office e híbrido adotado pela empresa.

O sistema tem muitas funcionalidades implementadas para gerenciar dados dos colaboradores, realizar o controle de horas em projetos, e ainda facilitar a coleta de dados para posterior processamento de folha. O sistema apresenta ainda integrações com outros sistemas, como, por exemplo, o sistema de departamento pessoal.

No controle de horas, o sistema possibilita que um colaborador alocado em 2 ou mais projetos faça o registro das suas horas trabalhadas no dia, informando em qual projeto e qual(is) atividades atuou dentro do projeto. Exemplo: o colaborador Imortal está alocado nos projetos Libertadores, Copa do Brasil e Brasileirão. Então, ele pode registrar que no dia trabalhado ele atuou 2h no Libertadores, 3 horas na Copa do Brasil e 4 horas no Brasileirão. Desta forma, ele teve 9 horas de trabalho no dia, sendo 1h extra, que por regras de negócio internas da empresa deve ser aprovada previamente com o gestor do projeto. Dependendo do contrato do cliente com a empresa, é necessário emitir relatórios mensais das horas trabalhadas a fim de serem aprovadas pelo cliente.

Ou seja, o sistema Timesheet tem o papel de realizar o registro de horas trabalhadas por projeto e por atividade, auxiliar o Administrativo em suas atividades de departamento pessoal e também gerar relatórios para diversos fins.

## 2.6. Clean Code:

As técnicas de clean code referem-se a práticas de programação que visam a criação de um

código legível, compreensível e fácil de manter. Entre os princípios mais importantes estão a simplicidade, a clareza e a modularização, com foco em evitar redundâncias e complexidade desnecessária.

Essas práticas são amplamente reconhecidas e recomendadas por especialistas na área de desenvolvimento de software, como Robert C. Martin, autor de "Clean Code: A Handbook of Agile Software Craftsmanship", que defende que a qualidade do código impacta diretamente na eficiência e na sustentabilidade do desenvolvimento de software a longo prazo. Quando aplicadas aos testes automatizados, as técnicas de clean code trazem diversos benefícios significativos, como a facilidade de manutenção, a redução de erros e a reutilização de código. Ao criar funções de teste bem estruturadas e nomeadas de forma clara, torna-se possível reduzir a duplicidade de código, facilitando a leitura e compreensão dos testes, o que minimiza a probabilidade de falhas durante a execução deles. Testes escritos com esses princípios em mente permitem que novos testes sejam incorporados com maior eficiência, além de serem mais flexíveis e adaptáveis a mudanças no sistema. Isso garante não apenas a integridade dos testes, mas também sua escalabilidade à medida que o projeto cresce.

Adicionalmente, a organização do código de testes segundo as diretrizes de clean code facilita a identificação e a correção de falhas, tornando o processo de depuração mais rápido e preciso. Como resultado, a aplicação dessas técnicas contribui significativamente para a confiabilidade e a agilidade no ciclo de desenvolvimento de software, aumentando a confiança dos desenvolvedores na qualidade do sistema e reduzindo custos a longo prazo, já que erros são detectados e corrigidos com mais rapidez.

## 2.7. Desenvolvimento Guiado por Comportamento (DBB)

Behavior Driven Development (BDD) é uma prática de desenvolvimento de software que se concentra em definir o comportamento esperado de um software antes que ele seja realmente desenvolvido. Para aplicar eficientemente o BDD, é necessário descrever os cenários em linguagem natural, de modo que todas as partes envolvidas no projeto possam compreendê-los facilmente. Depois que os cenários são escritos, o time de desenvolvimento inicia os trabalhos de codificação com apoio dos cenários. O ideal é que o desenvolvedor além de produzir o

código, também produza um teste automatizado que atenda as regras previstas no cenário produzido (Objective, 2023). Essa abordagem ajuda a garantir que o software atenda aos requisitos de negócios e funcione como esperado, promovendo uma colaboração mais estreita entre desenvolvedores, testadores e stakeholders. O uso de frameworks como Cucumber ou SpecFlow facilita a implementação do BDD, permitindo que os testes sejam escritos de forma legível para todos os envolvidos, sem a necessidade de entender profundamente detalhes técnicos de programação. Com o BDD, o desenvolvimento se torna mais alinhado às necessidades do cliente, reduzindo a probabilidade de falhas e aumentando a qualidade do produto final.

# 3. TRABALHOS RELACIONADOS

Em pesquisas feitas através do Google Acadêmico tendo como frase utilizada na barra de pesquisa: “aplicação de testes automatizados em sistema com Cypress”, três trabalhos semelhantes a este foram selecionados, onde os três aproximam-se com o tema deste trabalho por realizarem testes automatizados em sistemas web com o objetivo de otimizar, melhorar, encontrar bugs (problemas/falhas de código) ou proporcionar mais qualidade ao código de seus respectivos sistemas base.

O primeiro dos trabalhos selecionado é o de Rony Lourenço (AUTOMAÇÃO DE TESTES PARA UM SISTEMA DE E-COMMERCE, 2022.), nele o objetivo é aplicar o Cypress em uma plataforma de venda online para reduzir o tempo dos testes com uma cobertura ampla de testes possibilitando o reuso, a conclusão dele foi de que com o Cypress o tempo para realizar os testes foi 4 vezes mais rápido trouxe também de redução de custos operacionais. Maxwell Silva (Desenvolvimento de Testes Automatizados e Testes Manuais para um Sistema Web: Uma Análise Comparativa, 2021.) faz um trabalho muito similar, porém seu o objetivo é afirmar a necessidade do uso de ferramentas de testes automatizadas. Para atingir esse resultado ele realizou diversas pesquisas além de realizar testes práticos para concluir essa afirmação.

O trabalho de Thiago Moura (Automação de testes em aplicações web utilizando uma abordagem ad-hoc, 2021.) foi inovador onde foi desenvolvida uma ferramenta que simula o usuário a efetuar o preenchimento e a submissão dos dados, a validação da ferramenta foi feita através do desenvolvimento de uma aplicação web para cadastros, na qual os testes gerados tiveram sua cobertura de interfaces averiguada, e foi avaliado sua utilidade em indicar as falhas, além da capacidade de utilização com testes de regressão.

Estes trabalhos servem de base inspiradora para a apresentação visual e elaboração das produções textuais contidas neste trabalho, além de inspirar modelos de testes de código, ferramentas, os meios utilizados para realização de testes, os resultados apresentados e quaisquer outros pontos apresentados nestes trabalhos não foram considerados visto que estes são únicos para seu devido sistema base de testes, para se ter uma visão melhor na Tabela Comparativa. Neste trabalho as tanto o sistema Timesheet quanto a ferramenta de testes automatizados Cypress, foram requisitos da empresa SoftDesign, principal beneficiária dos resultados obtidos neste documento, para segundo solicitado: “Manter o padrão para que qualquer equipe quando disponível, trabalhar no Timesheet usando o material que será produzido assim podendo aprimorá-lo e incrementá-lo como bem quiser visto que estamos habituados com o uso do Cypress”.

**Tabela Comparativa**

| **Trabalhos** | **Ferramentas** | **Objetivo** | **Cenário** | **Testes Automatizados** |
|---|---|---|---|---|
| Este TCC | Cypress | Aumentar a qualidade de código com uso de Cypress. | Timesheet (sistema de controle de horas) | Sim |
| AUTOMAÇÃO DE TESTES PARA UM SISTEMA DE E-COMMERCE | Cypress | Aplicar o Cypress em uma plataforma de venda online para reduzir o tempo dos testes. | E-commerce | Sim |
| Desenvolvimento de Testes Automatizados e Testes Manuais para um Sistema Web: Uma Análise Comparativa. | Selenium, Xampp, Testlink e Cucumber. | Afirmar a necessidade do uso de ferramentas de testes automatizados. | Aplicação Web | Sim |
| Automação de testes em aplicações web utilizando uma abordagem ad-hoc. | N/D | Desenvolver uma ferramenta de automação de testes. | Aplicação Web | Sim |

Fonte: Elaborado por Kevin Bitencourt (2024).

# 4. METODOLOGIA

Com acesso ao ambiente de homologação: web-timesheet-stable.ingress.softdesign.com.br, utilizando a IDE Microsoft Visual Studio Code com o Cypress se iniciam os testes básicos de usabilidade padrão para conhecer o funcionamento. Para configurar o login no ambiente a SoftDesign designou um desenvolvedor para configurar a autenticação com um usuário pré-definido disponibilizado pela empresa.

Em reunião com as profissionais responsáveis pelo produto, foi levantado as seguintes funcionalidades para testes na modalidade CLT: apropriar horas, submeter a planilha e validar se o cálculo de horas extras está correto na folha. As profissionais elencaram essas funcionalidades pelo motivo que a maior parte dos colaboradores atua de carteira assinada.

## 4.1. Modelo

Foi necessário desenvolver um código base para uso na maioria dos testes, este realiza o preenchimento das horas, validando o dia da semana, quantidade de 8 horas semanais, além de fazer tratamentos de finais de semana e intervalo de almoço. Calcula o dia da semana através do preenchimento do mês atual, o código identifica se o mês possui 30 ou 31 dias tendo também um tratamento especial para o mês de fevereiro, calculando se possui 28 ou 29 dias, além desse tratamento ele também identifica os finais de semana os quais ele não deve preencher. O preenchimento de horas funciona de maneira randômica em intervalos de horas, foi colocado por padrão entrada entre 8:30 e 9h, ou seja, ele gera um horário entre esse intervalo com saída para o almoço 4 horas após essa entrada, o intervalo para o almoço também possui um tratamento tendo como horário de almoço variando entre 1 hora até 1:30 de almoço com retorno somado a mais 4 horas de trabalho para fechar 8 horas de trabalho diário conforme o contrato CLT.

A figura 7 abaixo mostra o código descrito acima.

timesheet-test > cypress > support > JS preenche.js > ...

```
/// <reference types="cypress" />
import * as calculoDia from "../../cypress/support/calculoDia";

Cypress.Commands.add('preencheHora', ()=> {

    let mes = 12;
    let ano = 2024;
    let totalDiasMes = calculoDia.obterTotalDiasMes(mes, ano);
    let dia = 1;

    for (let linha = 1; linha <= totalDiasMes * 2; linha++) {
      let horaEntradaManha = gerarHorarioEntre(8, 30, 9, 0);
      let horaSaidaManha = somarHoras(horaEntradaManha, 4);


      let intervaloAleatorio = gerarIntervaloAleatorio(1, 1, 30);
      let horaEntradaTarde = somarHorasComMinutos(horaSaidaManha, intervaloAleatorio);
      let horaSaidaTarde = somarHoras(horaEntradaTarde, 4);

      if (!calculoDia.verificarSeFimDeSemana(dia, mes, ano)) {
        preencherHoras(linha, 1, horaEntradaManha, "ENTRADA");
        preencherHoras(linha, 1, horaSaidaManha, "SAIDA");
        preencherHoras(linha, 2, horaEntradaTarde, "ENTRADA");
        preencherHoras(linha, 2, horaSaidaTarde, "SAIDA");
      }

      linha++;
      dia++;
    }
    cy.get('.primaryButton').click()
  });

  const preencherHoras = (linha, coluna, valor, acao) => {
    let acaoUser = 1;
    if (acao === "SAIDA") {
      acaoUser = 2;
    }
    cy.get(`:nth-child(${linha}) > .col-7 > :nth-child(1) > :nth-child(${coluna}) > :nth-child(${acaoUser}
      .clear().type(valor);
  };

  const somarHoras = (horaInicial, horasParaAdicionar) => {
    let [horas, minutos] = horaInicial.split(':').map(Number);
    horas += horasParaAdicionar;
    if (horas >= 24) {
      horas -= 24;
    }
    return `${horas.toString().padStart(2, '0')}:${minutos.toString().padStart(2, '0')}`;
  };


  const somarHorasComMinutos = (horaInicial, minutosParaAdicionar) => {
    let [horas, minutos] = horaInicial.split(':').map(Number);
    minutos += minutosParaAdicionar;
    if (minutos >= 60) {
      horas += Math.floor(minutos / 60);
      minutos = minutos % 60;
    }
    if (horas >= 24) {
      horas -= 24;
    }
    return `${horas.toString().padStart(2, '0')}:${minutos.toString().padStart(2, '0')}`;
  };

  const gerarHorarioEntre = (horaInicio, minutoInicio, horaFinal, minutofinal) => {
    const inicioMinutos = horaInicio * 60 + minutoInicio;
    const fimMinutos = horaFinal * 60 + minutofinal;
    const minutosAleatorios = Math.floor(Math.random() * (fimMinutos - inicioMinutos + 1)) + inicioMinutos
    const horas = Math.floor(minutosAleatorios / 60);
    const minutos = minutosAleatorios % 60;
    return `${horas.toString().padStart(2, '0')}:${minutos.toString().padStart(2, '0')}`;
  };

  const gerarIntervaloAleatorio = (horaMin, minutoMin, minutoMax) => {
    const intervaloMinutos = horaMin * 60 + minutoMin;
    const intervaloMaxMinutos = horaMin * 60 + minutoMax;
    return Math.floor(Math.random() * (intervaloMaxMinutos - intervaloMinutos + 1)) + intervaloMinutos;
  };
```

*Figura 7 Print do código do modelo no Visual Studio Code.*

## 4.2. Cenários de Teste

Em uma reunião no formato de brainstorm com a QA do projeto, sem questionários ou roteiros apenas com a pergunta inicial: “Quais os cenários precisam ser testados?”, foi levantado as funcionalidades as quais deveriam ser focadas nesse início, os cenários de testes foram se formando e maturando conforme a reunião evoluía, visando atender as principais funcionalidades para o funcionamento esperado do Timesheet. Após um refinamento chegou-se nos seguintes formatos de cenários de teste no formato de BDD.

1. **Lançamento e Apropriação de Horas:**
   Dado que usuário esteja na tela Planilha de Horas
   Quando informar um conjunto de horas
   Então o sistema deve ser possível realizar a apropriação de horas preenchidas

2. **Validar apropriação pendente:**
   Dado que usuário tenha informado horas para todo o mês corrente
   Quando não realizar a apropriação das horas inseridas
   Então o sistema deve mostrar o ícone de "Apropriação pendente"

3. **Validar apropriação correta:**
   Dado que usuário tenha informado horas para todo o mês corrente
   Quando realizar a apropriação das horas inseridas
   Então o sistema **não deve** mostrar o ícone de "Apropriação pendente"

4. **Validar edição de apropriação:**
   Dado que usuário tenha informado um conjunto de horas com suas respectivas apropriações
   Quando horas estiverem com o status <status>, sendo clicado no botão <ação>
   Então o sistema deve <situação>
   <status> | <ação> | <situação>
   Rascunho | Excluir | retirar o registro da apropriação
   Reprovado | Editar | permitir alterar a apropriação

5. **Validar bloqueio de edição de apropriação:**
   Dado que usuário tenha informado um conjunto de horas com suas respectivas apropriações
   Quando horas estiverem com o status <status> no mês
   Então o sistema deve exibir os campos de lançamento E apropriação bloqueados
   <status>
   Submetido
   Aprovado

6. **Validar preenchimento das horas com data de vínculo distinta:**
   Dado que usuario possua intervalo entre vínculos de projeto / atividade
   Quando informar horas na planilha, com valor em data sem vínculo

Então o sistema não deve permitir salvar para a data em questão, exibindo mensagem de erro

7. **Validar aviso ao informar preenchimento de horas sobrepostas:**
   Dado que usuário informe mais de um conjunto de horas
   Quando os valores informados forem sobrepostos
   Então o sistema deve exibir ícone de alerta referente as horas sobrepostas

8. **Submissão de Planilha de Horas:**
   Dado que usuário tenha informado suas horas, com apropriação preenchida corretamente
   Quando informar a data do período da submissão, clicando no botão "Submeter"
   Então o sistema deve exibir modal de confirmação

9. **Validar submissão da planilha com sucesso:**
   Dado que usuário tenha informado suas horas, com apropriação preenchida corretamente
   Quando confirmar submissão das horas
   Então o sistema deve recarregar a página exibindo o status para o período selecionado como "Submetido", exibindo a mensagem "Planilha submetida com sucesso"

10. **Validar submissão da planilha com intervalo de data diferente:**
    Dado que usuário esteja submetendo planilha
    Quando informar data limite de submissão, sendo submetido com sucesso
    Então deve recarregar a tela com as horas apropriadas submetidas desabilitadas para edição, contendo status como "Submetido" até a data enviada

11. **Validar bloqueio de apropriação para horas sobrepostas:**
    Dado que usuário tenha informado horas sobrepostas
    Quando abrir aba de apropriação de horas
    Então deve bloquear a seleção de valores para apropriação das horas sobrepostas

## 4.3. Testes

Serão apresentados os casos de teste para efeito de ilustração visual da aplicação do Cypress no ambiente de teste do Timesheet.

1. **Lançamento e Apropriação de Horas**

   Neste caso de teste, informa-se um conjunto de horas e testa-se se o sistema permite realizar a apropriação de horas preenchidas. A figura 8 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

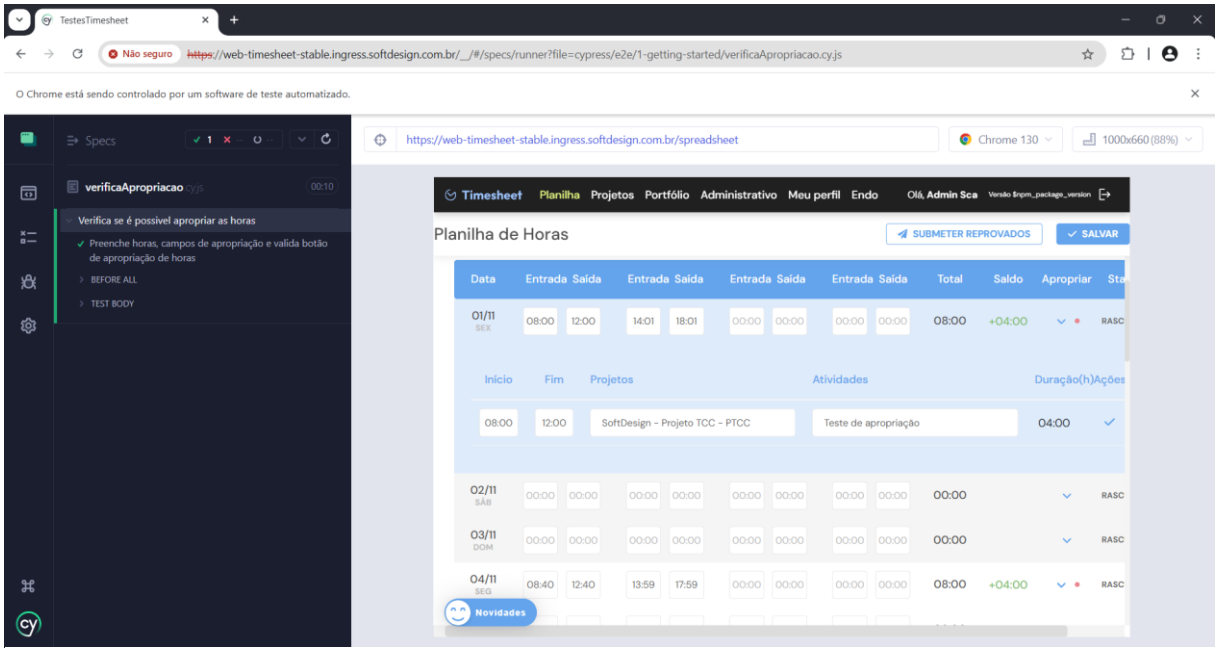


*Figura 8 Print do teste rodando com Cypress no navegador Chrome.*

## 2. Validar apropriação pendente

Neste caso de teste, informa-se um conjunto de horas para todo o mês corrente sem realizar a apropriação das horas inseridas testa-se o sistema devendo mostrar o ícone de "Apropriação pendente" representado por um círculo vermelho. A figura 9 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

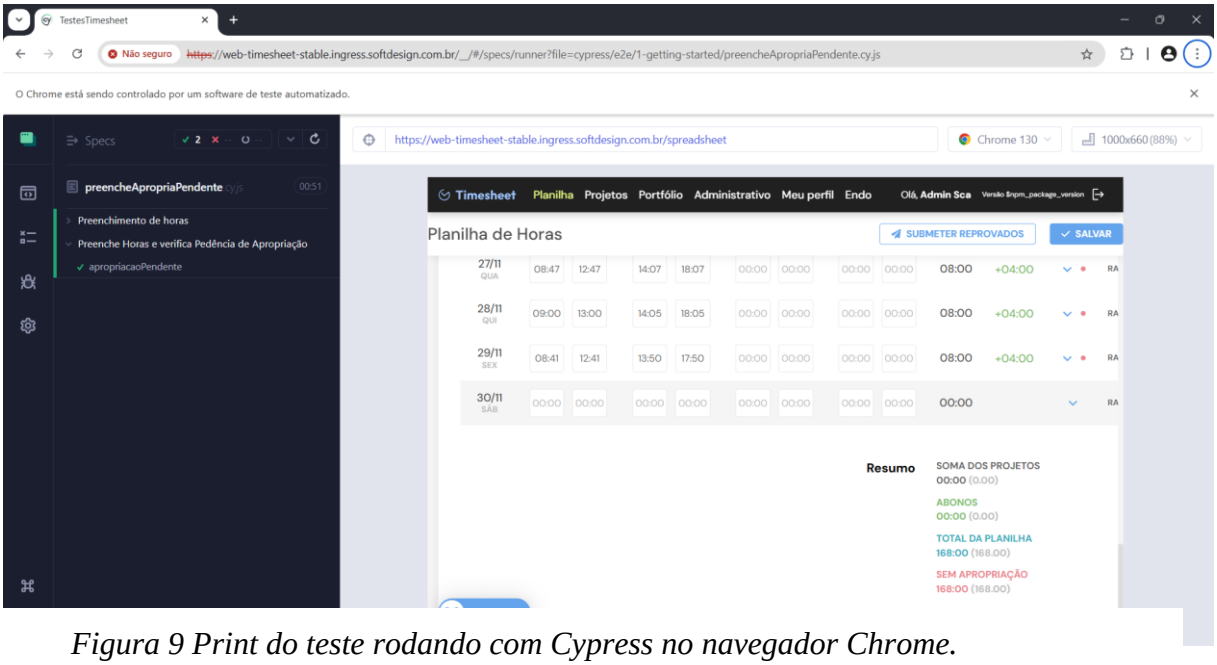


*Figura 9 Print do teste rodando com Cypress no navegador Chrome.*

### 3. Validar apropriação correta

Neste caso de teste, informa-se um conjunto de horas para todo o mês corrente sem realizar a apropriação das horas inseridas, realizando a apropriação das horas inseridas testa-se o sistema devendo **não** mostrar o ícone de "Apropriação pendente" representado por um círculo vermelho. A figura 10 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

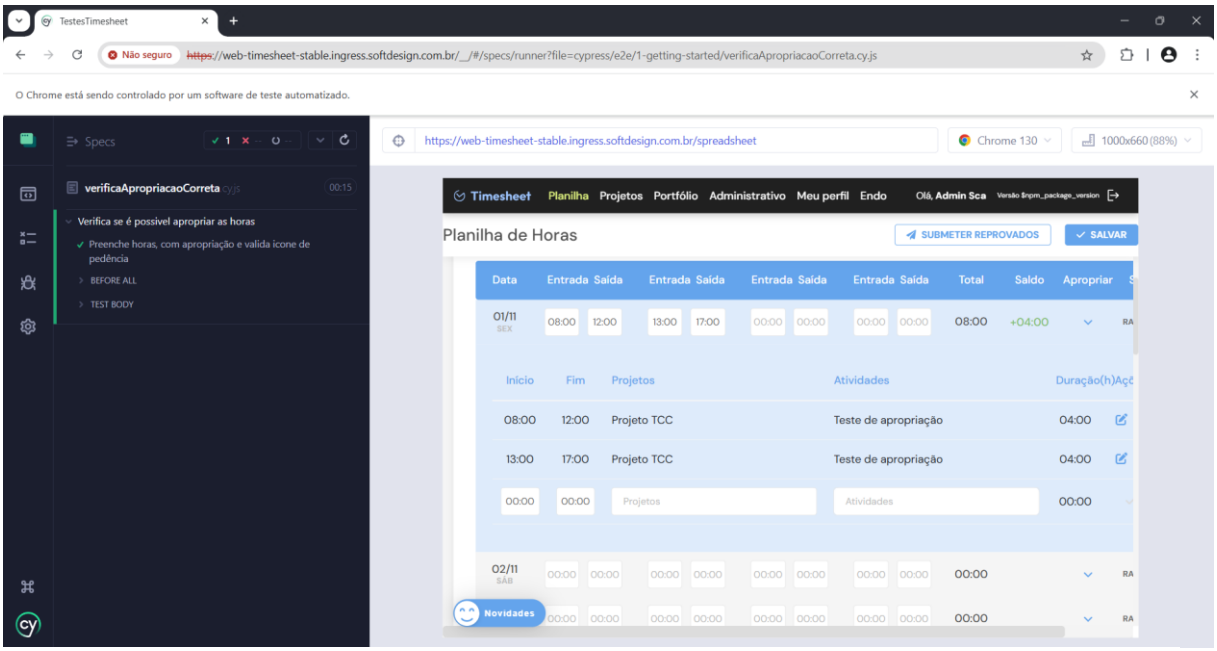


*Figura 10 Print do teste rodando com Cypress no navegador Chrome.*

4. **Validar edição de apropriação**

Neste caso de teste, informa-se um conjunto de horas com a apropriação das horas inseridas, realizando a apropriação das horas inseridas testa-se o sistema devendo permitir a edição e exclusão das horas apropriadas**.** A figura 11 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

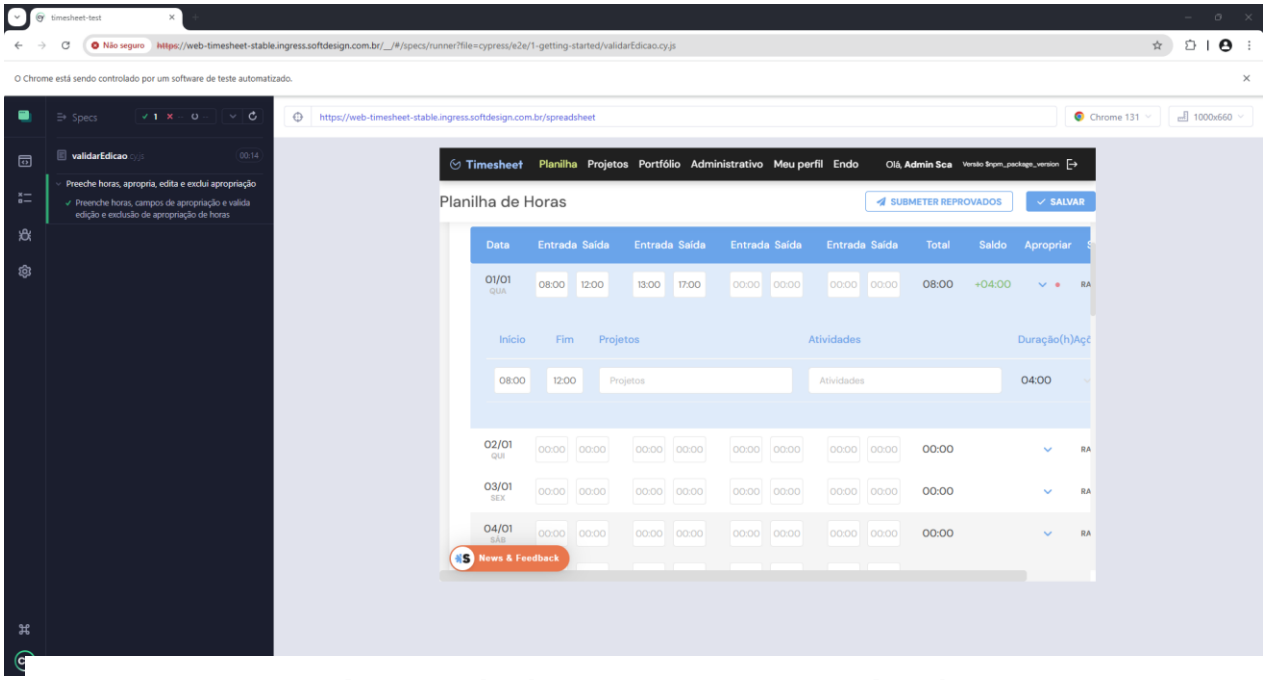


*Figura 11 Print do teste rodando com Cypress no navegador Chrome.*

5. **Validar bloqueio de edição de apropriação**

Neste caso de teste, informa-se um conjunto de horas com a apropriação das horas inseridas, realizando a apropriação e submissão testa-se o sistema devendo **não** permitir a edição e exclusão das horas apropriadas**.** A figura 12 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

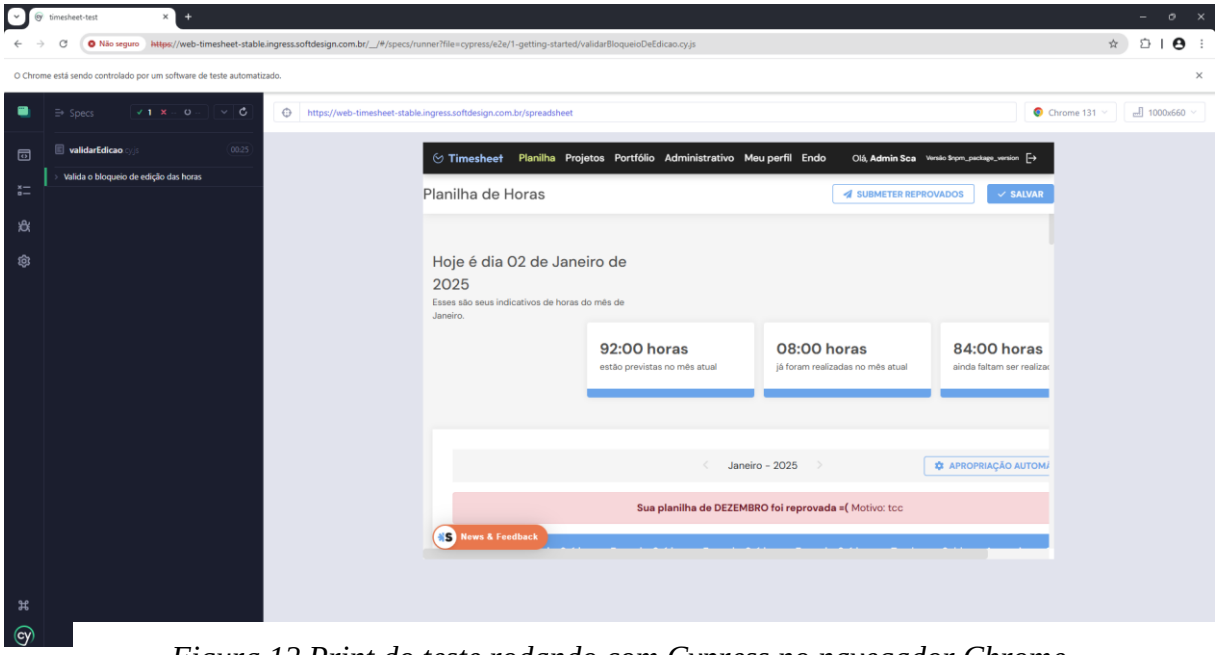


*Figura 12 Print do teste rodando com Cypress no navegador Chrome.*

6. **Validar preenchimento das horas com data de vínculo distinta**

   Dado que usuario possua intervalo entre vínculos de projeto / atividade Quando informar horas na planilha, com valor em data sem vínculo Então o sistema **não** deve permitir salvar para a data em questão, exibindo mensagem de erro. **Não foi viável no período encontrar uma solução para trabalhar com projetos de banco diferentes junto a id da planilha do usuário devido a problemas de complexidade do código do Timesheet.**

7. **Validar aviso ao informar preenchimento de horas sobrepostas:**

Dado que usuario informe mais de um conjunto de horas quando os valores informados forem sobrepostos o sistema deve exibir ícone de alerta referente as horas sobrepostas. A figura 13 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

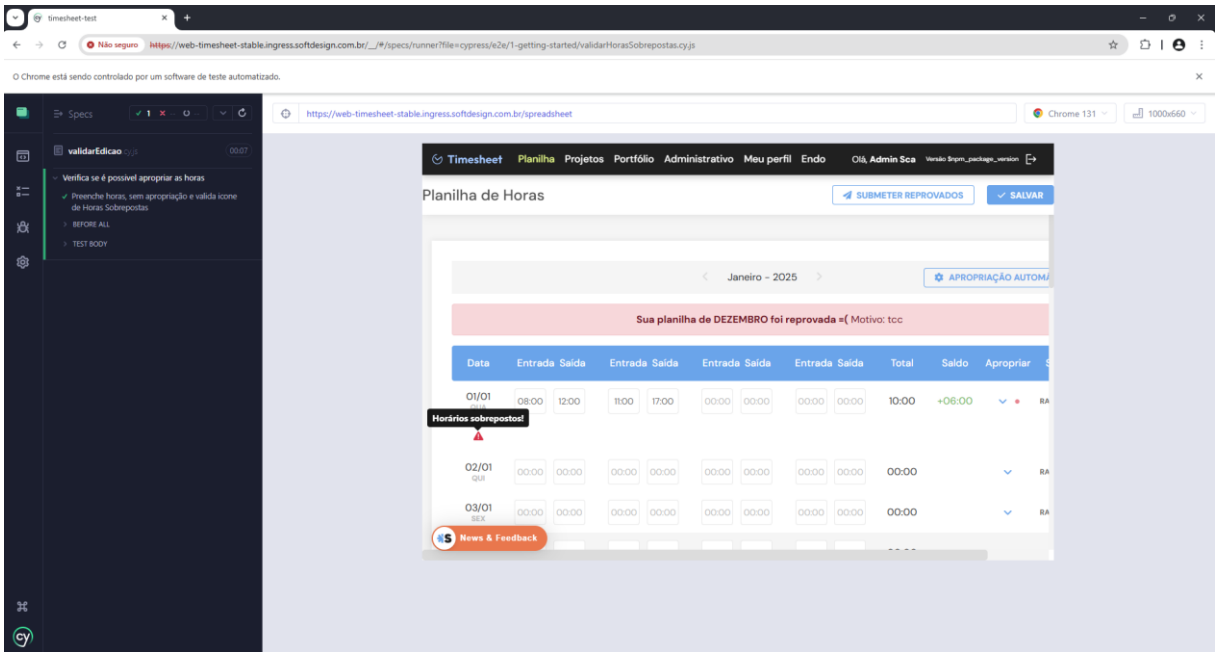


*Figura 13 Print do teste rodando com Cypress no navegador Chrome.*

## 8. Submissão de Planilha de Horas

Dado que usuario tenha informado suas horas, com apropriação preenchida corretamente quando informar a data do período da submissão, clicando no botão "Submeter" então o sistema deve exibir modal de confirmação. A figura 14 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

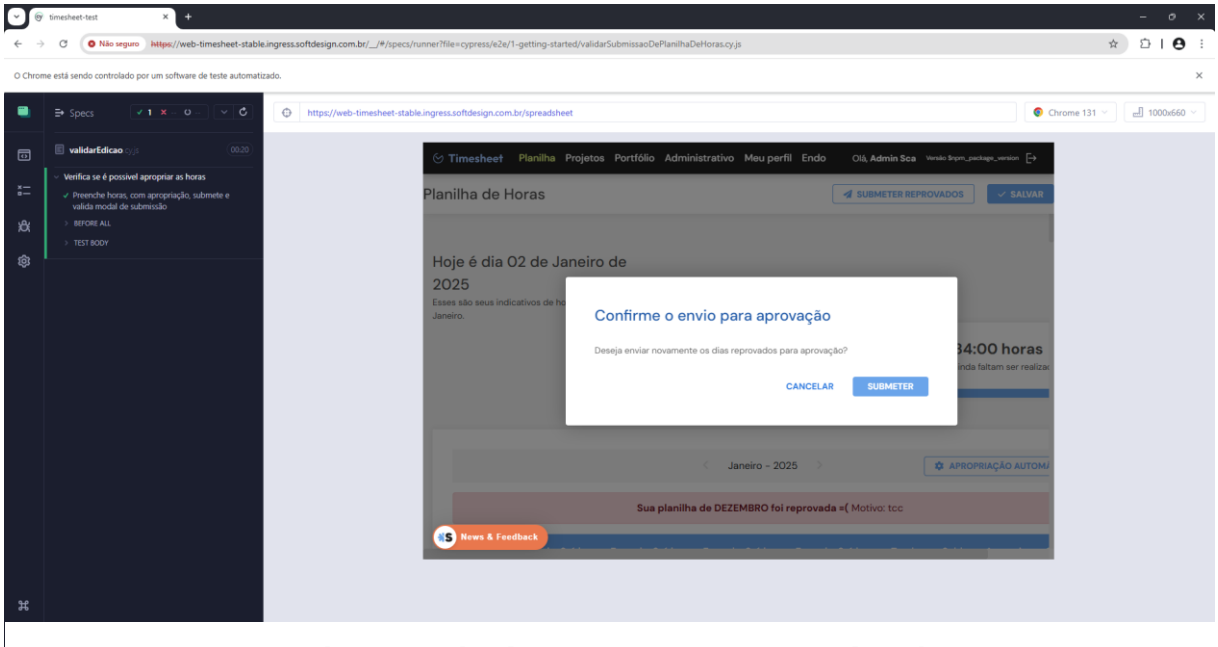


*Figura 14 Print do teste rodando com Cypress no navegador Chrome.*

**9. Submissão de Planilha de Horas com Sucesso**

Dado que usuario tenha informado suas horas, com apropriação preenchida corretamente quando confirmar a submissão das horas, clicando no botão "Submeter" Então o sistema deve recarregar a página exibindo o status para o período selecionado como "Submetido", exibindo a mensagem "Planilha submetida com sucesso". A figura 15 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

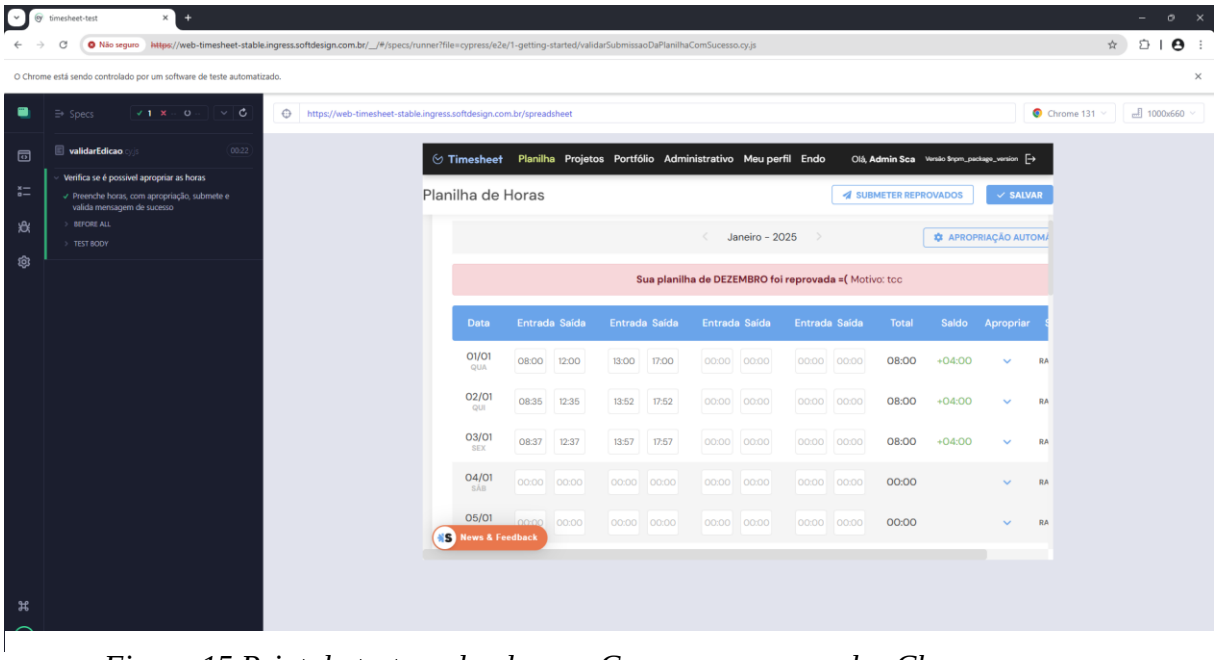


*Figura 15 Print do teste rodando com Cypress no navegador Chrome.*

**10. Submissão de Planilha de Horas com Intervalo Diferente**

Dado que usuario esteja submetendo planilha quando informar data limite de submissão, sendo submetido com sucesso então deve recarregar a tela com as horas apropriadas submetidas desabilitadas para edição, contendo status como "Submetido" até a data enviada. A figura 16 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

**11. Validar bloqueio de apropriação para horas sobrepostas**

Dado que usuario tenha informado horas sobrepostas quando abrir aba de apropriação de horas então deve bloquear a seleção de valores para apropriação as horas sobrepostas. A figura 17 abaixo traz o resultado positivo onde as ações esperadas foram atendidas.

*Figura 16 Print do teste rodando com Cypress no navegador Chrome.*

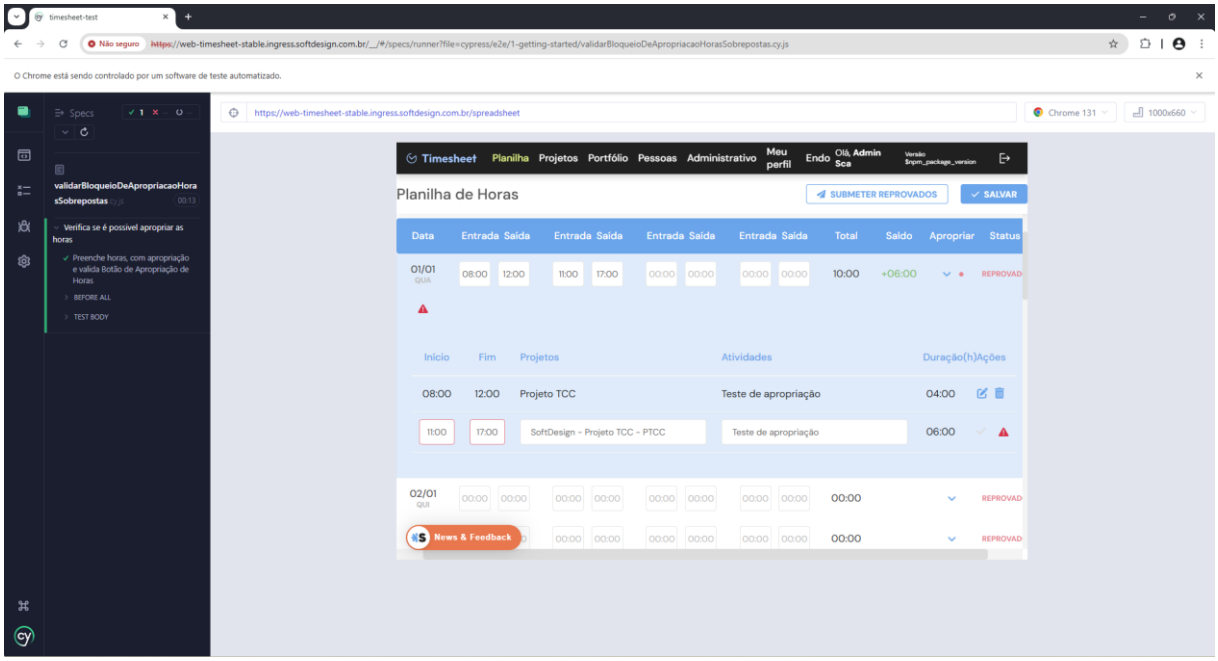


*Figura 17 Print do teste rodando com Cypress no navegador Chrome.*

## 5. RESULTADO

Os testes desenvolvidos neste trabalho de conclusão foram integrados às pipelines de teste da ferramenta. Durante o processo, foram identificados alguns problemas de configuração do banco de dados, que resultavam na perda de horas registradas. Além disso, constatou-se que algumas funcionalidades haviam sido implementadas utilizando injeção de APIs, o que dificultou a realização de testes em determinados campos e funcionalidades.

No decorrer deste trabalho, também foi desenvolvida uma nova funcionalidade chamada "Abono", a qual não foi abordada neste estudo devido ao tempo disponível para sua finalização e à falta de maturidade dessa funcionalidade. Essa nova funcionalidade gerou uma série de problemas nos testes previamente realizados, que passaram por uma nova refatoração.

Os testes foram incorporados ao repositório da SoftDesign na Azure, e servirão como base para o desenvolvimento de novos testes, incluindo os testes de API realizados na primeira semana de janeiro de 2025.

A conclusão da SoftDesign, através do responsável Raphael Rodrigues – Head Porfolio, evidenciou que o trabalho contribuiu significativamente para um processo que estava sem um

responsável há algum tempo e que poderia gerar problemas com o aumento do fluxo de projetos. Ressaltou que a iniciativa trouxe maior confiabilidade, uma vez que os problemas relacionados à perda de dados foram resolvidos, além de contribuir a evidenciar a necessidade de otimizar o código para reduzir os tempos de execução das apropriações. Outro ponto, foi que a iniciativa, apoiou o início do trabalho de automação de testes com Cypress, para a pessoa QA Engineer, que atua no projeto Timesheet, evoluir mais cenários automatizados

A figura 18 abaixo mostra o retorno da pipeline rodando os testes no ambiente da Softdesign, onde foram executados os onze casos de teste acima com sucesso.

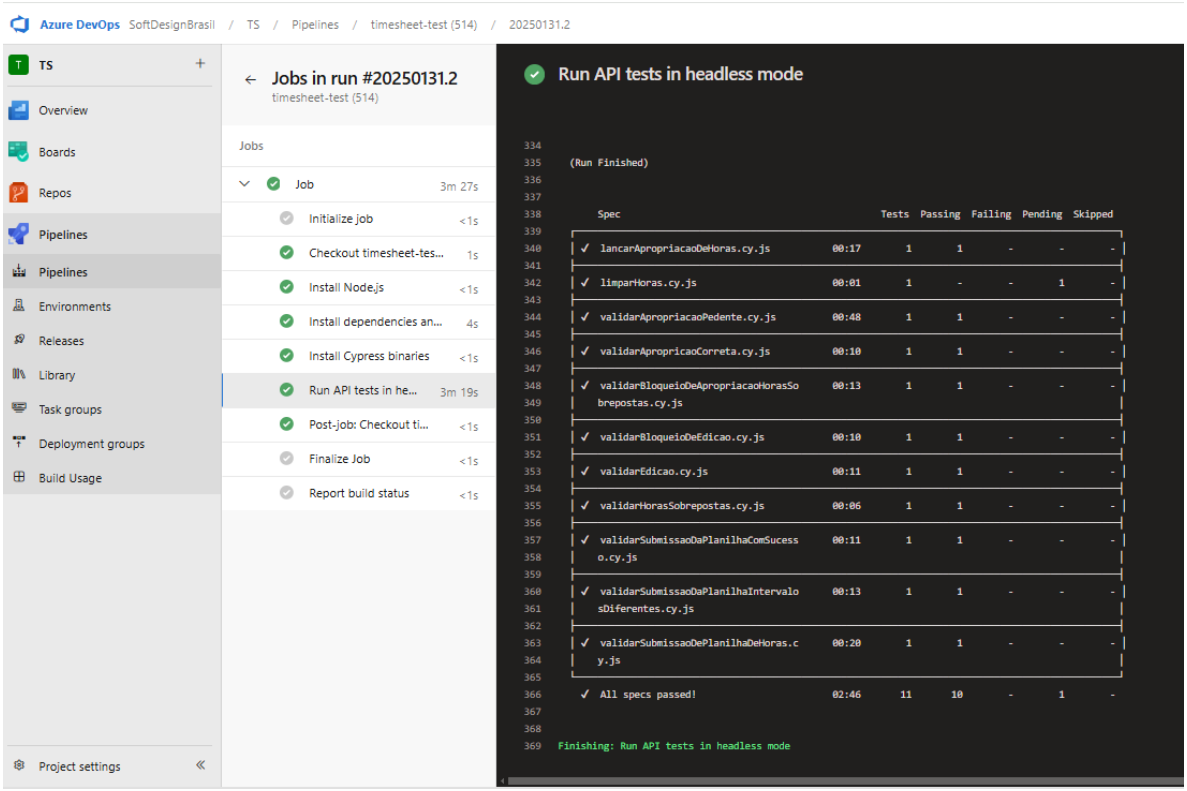


*Figura 18 Print do resultado dos testes rodando com Cypress da Pipeline no navegador Chrome.*

## 5.1. Evidências

### Falha de preenchimento de horas - Gravidade Média

O problema ocorre ao realizar o preenchimento de horas, após o salvamento automático parte das horas preenchidas somem, conforme mostrado na figura 19 abaixo.

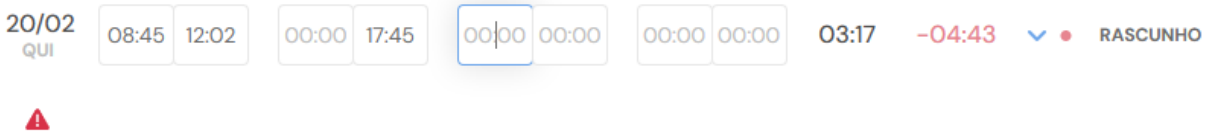


*Figura 19 Print do problema do Timesheet no navegador Chrome.*

**Travamento - Gravidade Alta**

O problema ocorre aleatoriamente durante o uso do Timesheet, o sistema trava e exibe a seguinte mensagem de erro “ATENÇÃO: Não conseguimos fazer seu login porque suas configurações estão incompletas. Entre em contato com o Administrativo”, conforme mostrado na figura 20 abaixo.

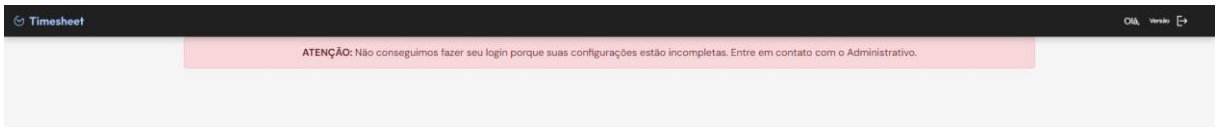


*Figura 20 Print do problema do Timesheet no navegador Chrome.*

**Falha de preenchimento de horas - Gravidade Média**

O problema ocorre ao realizar a apropriação de horas, após as horas serem preenchidas, na aba de apropriação não são reconhecidas as horas corretas, conforme mostrado na figura 21 abaixo.

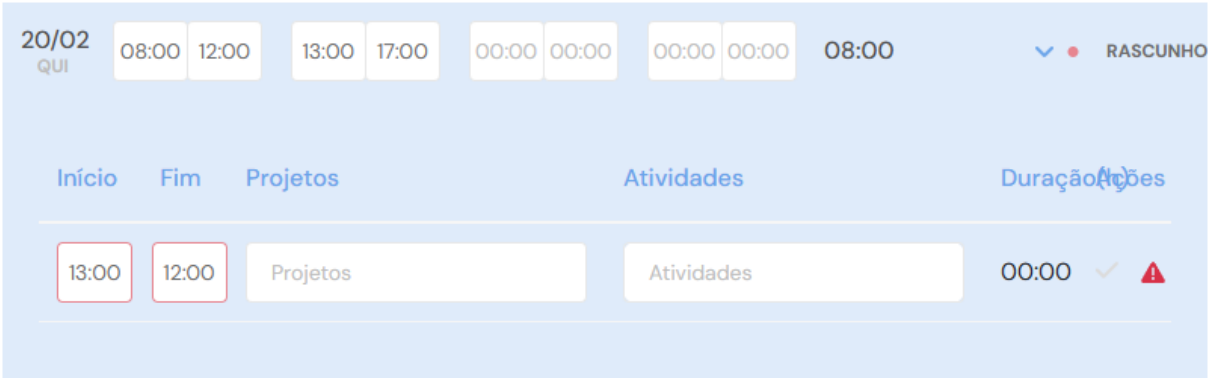


*Figura 21 Print do problema do Timesheet no navegador Chrome.*

# 6. CONCLUSÃO

Este trabalho teve como objetivo a automação de casos de testes funcionais para o sistema de controle e registro de horas **Timesheet** da SoftDesign. Foram desenvolvidos onze casos de teste, dos quais dez foram concluídos, uma vez que algumas funcionalidades do código da ferramenta apresentaram uma alta complexidade.

Foi possível criar modelos de testes que funcionam como modelos para futuras verificações, permitindo a reutilização das funções já desenvolvidas em novos casos de teste. Os testes foram executados no navegador **Google Chrome**, e os registros dos testes realizados estão apresentados no tópico 4.3 deste trabalho.

Para o trabalho foram realizadas pesquisas que incluem o estudo de ferramentas, técnicas de codificação e testes automatizados, bem como a aplicação prática de técnicas relacionadas ao uso de bancos de dados. Como resultado, diversas melhorias no desempenho do sistema foram observadas. Embora não tenha sido possível realizar todos os casos de teste inicialmente previstos, os objetivos propostos neste trabalho foram plenamente atingidos.

Para trabalhos futuros, como a automação foi projetada com foco na reutilização, o código desenvolvido poderá ser refatorado para atender a outros projetos da empresa. A pessoa QA Engineer que for atuar no Timesheet deve desenvolver mais cenários de forma contínua para expandir a cobertura de testes e reduzir falhas e erros.

# REFERÊNCIAS BIBLIOGRÁFICAS

MULLER, GUILHERME. O que é Teste de Software? Por que é necessário?.2020. Disponível em: https://cwi.com.br/blog/o-que-e-teste-de-software-por-que-e-necessario/. Acesso em: 1 set. 2024.

PITTET, STEN. Diferentes tipos de testes de software.2024. Disponível em: https://www.atlassian.com/br/continuous-delivery/software-testing/types-of-software-testing. Acesso em: 1 set. 2024.

IBM, TEAM. O que é teste de software?.2024. Disponível em: https://www.ibm.com/br-pt/topics/software-testing. Acesso em: 1 set. 2024.

BLOG, UMBLER. Qualidade de Software #2: As vantagens da automação de testes.2018. Disponível em: https://blog.umbler.com/br/qualidade-de-software-2-as-vantagens-da-automacao-de-testes/?utm_source=google&utm_medium=cpc&utm_campaign=6456682452&utm_term=&utm_content=80087430391&gad_source=1&gclid=Cj0KCQjwjY64BhCaARIsAIfc7YYQWJ12whce5ouu8ZDI3432QMAcZVWdz3ny3ejlFowf8jH7CrqLIlcaAqh-EALw_wcB. Acesso em: 1 set. 2024.

FILHO, AVELINO. Testes automatizados: como garantir a qualidade do software.2018. Disponível em: https://br.k21.global/gestao-de-times-ageis/qualidade-do-software-testes-automatizados?gad_source=1&gclid=Cj0KCQjwjY64BhCaARIsAIfc7YaIjA8IbQuas0mZ2lP1s33VneRf0RGoc0nO-983YJfqz8Ts78FZelMaAqGBEALw_wcB. Acesso em: 1 set. 2024.

RANGEL, MARCOS. Um breve resumo sobre Pirâmide de testes.2024. Disponível em: https://www.dio.me/articles/um-breve-resumo-sobre-piramide-de-testes. Acesso em: 21 out. 2024.

MUNIZ, NATHALYA. Pirâmide de Testes.2020. Disponível em: https://medium.com/@nmuniz/pir%C3%A2mide-de-testes-328ca50f31cd. Acesso em: 21 out. 2024.

DESENVOLVEDOR, CASA. Testes de Integração: Tipos, Importância e Vantagens.2024. Disponível em: https://blog.casadodesenvolvedor.com.br/testes-de-integracao-tipos-importancia-e-vantagens/?gad_source=1&gclid=Cj0KCQjwmt24BhDPARIsAJFYKk3cg39DjdshkqNV3hjg0NpVvVNx9r6GZ904YKV-z9ZfgAlDdFG_3GsaAhLSEALw_wcB. Acesso em: 21 out. 2024.

COUTO, ARMANDO. Dublês de Teste.2023. Disponível em: https://coutoarmando.medium.com/os-testes-de-interface-s%C3%A3o-uma-parte-fundamental-do-desenvolvimento-de-software-uma-vez-que-3742e8541385. Acesso em: 21 out. 2024.

LEARN, MICROSOFT. Testando uma interface do usuário.2023. Disponível em: https://learn.microsoft.com/pt-br/windows/win32/appuistart/testing-a-user-interface. Acesso em: 21 out. 2024.

GABRIELA, LARISSA. Por que e o que é possível testar?.2021. Disponível em: https://www.alura.com.br/artigos/por-que-e-o-que-e-possivel-testar. Acesso em: 21 out. 2024.

MOURA, Thiago Santos de. AUTOMAÇÃO DE TESTES EM APLICAÇÕES WEB UTILIZANDO UMA ABORDAGEM AD-HOC.2021. Orientador: Prof. Dr. Cláudio de Souza Baptista. Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) – Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

SILVA, Maxwell Anjo da. Desenvolvimento de Testes Automatizados e Testes Manuais para um Sistema Web. Orientador: Prof. Me. Eduardo Henrique da Rocha do Nascimento. Monografia (Monografia em Ciência da computação). – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2021.

LOURENÇO, Rony de Senna. AUTOMAÇÃO DE TESTES PARA UM SISTEMA DE E-COMMERCE.2022. Orientadora: Prof. Dr. Uirá Kulesza Monografia (graduação) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Curso de Engenharia de Computação, Natal, RN, 2022.

Timesheet:
HARTMANN, KARINA. Timesheet – Guia de uso.2020. Disponível em: https://portal.softdesign.com.br/timesheet-guia-de-uso/. Acesso em: 14 set. 2024.

DOCS, CYPRESS. Why Cypress?.2024. Diponivel em: https://docs.cypress.io/guides/overview/why-cypress Acesso em: 5 out. 2024.

SANTOS, EDERSON. Cypress: passo a passo para começar a usar.2021. Disponível em: https://softdesign.com.br/blog/cypress_passo_a_passo_para_comecar_a_usar/#Uso-de-variaveis-de-ambiente. Acesso em: 5 out. 2024.

BERÇAM, RAFAEL. Um Overview sobre Cypress.io — Framework de Automação de Testes end-to-end.2019. Disponível em: https://medium.com/@faelbercam/um-overview-sobre-cypress-io-framework-de-automa%C3%A7%C3%A3o-de-testes-end-to-end-dc438b9ee7a1. Acesso em: 5 out. 2024.

COUTO, LEANDRO. Vamos falar de automação de testes web: Cypress.2020. Disponível em: https://medium.com/@leandrowcs1985/vamos-falar-de-automa%C3%A7%C3%A3o-de-testes-web-cypress-6dfddfcb8d7d. Acesso em: 5 out. 2024.

Martin, Robert. Clean Code: A Handbook of Agile Software Craftsmanship. Massachusetts. James O. Coplien. 431p. 2009. Disponível em: https://kolegite.com/EE_library/books_and_lectures/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5/Clean%20Code%20-%20%20A%20Handbook%20of%20Agile%20Software%20Craftsmanship.pdf. Acesso em: 15 jan. 2025

Garcia, Lorena. Qual a vantagem de utilizar o cypress?. 2024. Disponível em: https://cursos.alura.com.br/forum/topico-duvida-qual-a-vantagem-de-utilizar-o-cypress-427427. Acesso em: 05 jun. 2024.

Objective. Testes de Software: Definição, Conceitos e Exemplos.2022. Disponível em: https://www.objective.com.br/insights/testes-de-software/. Acesso em: 31 jan. 2025.

Objective. Behavior Driven Development (BDD): saiba como aplicar, quais as fases e os benefícios.2023. Disponível em: https://www.objective.com.br/insights/bdd/. Acesso em: 3 fev. 2025.