LILIANA SÁBATO TEODORO

RELATÓRIO DE ESTÁGIO DE UMA ANASLISTA DE QA

LAVRAS – MG

2023

LILIANA SÁBATO TEODORO

RELATÓRIO DE ESTÁGIO DE UMA ANALISTA DE QA

Relatório de Estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para obtenção do título de Bacharel.

Prof. Dr. Antônio Maria Pereira de Resende
Orientador

LAVRAS - MG
2023

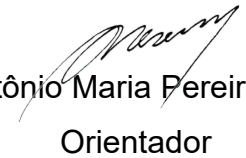
LILIANA SÁBATO TEODORO

RELATÓRIO DE ESTÁGIO DE UMA ANALISTA DE QA

Relatório de Estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para obtenção do título de Bacharel.

APROVADA em 27 de fevereiro de 2023.

Prof. Dr. Antônio Maria Pereira de Resende	DCC/UFLA
Prof. Dr. Bruno de Abreu Silva	DCC/UFLA
Prof. Dra. Renata Lopes Rosa	DCC/UFLA



Prof. Dr. Antônio Maria Pereira de Resende
Orientador

LAVRAS - MG

2023

Dedico à minha mãe Lilian, ao meu pai Delmário e à minha irmã Ana Luiza que sempre me deram força e suporte para chegar até aqui.

RESUMO

Este relatório de estágio descreve as atividades realizadas na área de qualidade de software, em particular, na realização de testes em uma aplicação Web. O estágio foi realizado entre agosto de 2021 e maio de 2022 na YouxGroup, empresa com sede em Lavras (MG), que trabalha diretamente com desenvolvimento de soluções tecnológicas, tais como aplicativos móveis e websites. Para esse estudo, foi realizada uma revisão bibliográfica e durante o estágio, realizaram-se atividades no projeto denominado “Observatório”, responsável por apresentar por meio de tabelas, gráficos, e representações gráficas informações de interesse da Agropecuária brasileira. Nele, foram realizados diversos testes nos painéis desenvolvidos, durante as *sprints*. Estes painéis são divididos em Plataforma Estatística e Plataforma Geoespacial. Ademais, foram realizados testes de Caixa-Preta no sistema, onde foram verificados os seus funcionamentos. Tais testes foram executados através dos cenários de testes escritos antes do desenvolvimento das *features*. Neste relatório serão descritas as atividades realizadas e as metodologias utilizadas para contextualização das práticas executadas.

Palavras-chave: Teste de Software; Testes Manuais; Qualidade de Software; Teste de sistemas WEB.

ABSTRACT

This work placement report describes the activities carried out in the area of software quality, in particular, in carrying out tests on a Web application. The work placement was performed out during the August 2021 until May 2022 at YouxGroup, a company based in Lavras (MG), which works directly with the development of technological solutions, such as mobile applications and websites. For this study, a bibliographic review was carried out and during the work placement, activities were carried out in the project called "Observatory", responsible for presenting, through tables, graphs, and graphic representations, information of interest to Brazilian Agriculture. In it, several tests were carried out on the panels developed during the sprints. These steps are divided into Statistical Platform and Geospatial Platform. In addition, Black Box tests were carried out on the system, where its functioning was verified. Such tests were performed through test scenarios written before the development of features. This report will describe the activities carried out and the methodologies used to contextualize the practices carried out.

Keywords: Software Testing; Manual Testing; Quality Analysis; Test of WEB systems.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

FEATURE	Nova Funcionalidade
V&V	Verificação e Validação
QA	Quality Assurance
SQL	Structured Query Language

SUMÁRIO

1 INTRODUÇÃO

A autora deste relatório realizou estágio na YouxGroup, entre agosto de 2021 e maio de 2022, empresa com sede em Lavras, que trabalha diretamente com desenvolvimento de soluções tecnológicas como aplicativos móveis e websites. Neste relatório de estágio, descrevem-se as atividades realizadas na área de qualidade de software, em particular, em testes manuais em uma aplicação Web denominada *Observatório da Agropecuária Brasileira*. Essa aplicação trata-se de uma plataforma que visa prover dados, informações, conteúdos e painéis dinâmicos sobre o setor agropecuário no país.

Durante o estágio, cenários de testes foram escritos e aplicados manualmente para identificar defeitos, após o desenvolvimento de funcionalidades para essa plataforma. Para isso, foram verificadas as regras de negócio, definidas juntamente ao cliente, e o fluxo de execução das funcionalidades do sistema Web. Foram utilizadas ferramentas como o Sharepoint para escrita online e compartilhada dos cenários de testes e o GitLab para organizar o *backlog*, para auxiliar no gerenciamento do Scrum com Kanban, além de registrar as *issues* como novas funcionalidade ou *bugs*.

Nesse sentido, após um (1) ano de estágio, ficou claro para a estagiária que ela foi contratada para somar homem/hora na linha de produção e garantir a execução dos testes em tempo adequado. Também foi possível notar que é preciso repensar o teste dentro da empresa, automatizando-o, pois, atualmente, são mais de 10 pessoas executando manualmente os testes.

Para tais fins, este relatório está organizado nos seguintes capítulos: o segundo capítulo apresenta uma contextualização do ambiente organizacional da empresa onde foi realizado o estágio e alguns de seus projetos; O terceiro capítulo apresenta um referencial teórico sobre testes de software e as tecnologias utilizadas no decorrer da trajetória do estágio e o quarto e último capítulo detalha o trabalho desenvolvido na empresa.

2 SOBRE A YOUXGROUP

Este capítulo tem o intuito de descrever a organização em que o estágio foi realizado, citando uma breve contextualização de sua história, seus processos organizacionais e seus produtos relacionados à área do autor, sendo essa área, a de tecnologia.

2.1 YouxGroup

A empresa YouxGroup[1] tem clientes por todo Brasil, sendo eles órgãos públicos, tais como o Governo Federal e o governo do estado do Pará, mas também possui clientes do setor privado, como por exemplo a Suzano - empresa brasileira de papel e celulose - e a Munich Re, uma das maiores companhias de resseguro do mundo.

A empresa YouxGroup, fundada em 2020 a partir da junção de 3 empresas situadas na cidade de Lavras, visa fomentar a inovação e o desenvolvimento tecnológico no Brasil, seguindo valores, tais como: Reconhecimento Pessoal, Inovação e Sustentabilidade. Sendo assim, a YouxGroup trata-se de um grupo de empresas que, trabalhando juntas, conta com mais de 200 funcionários trabalhando de vários lugares do Brasil, majoritariamente na modalidade *home office*, tendo também alguns funcionários da sede na modalidade híbrida.

A YouxGroup é dividida em três unidades de negócio, sendo elas *Tech*, *Flow* e *Analytics*. A unidade *Tech* é a responsável por multiplicar *expertises*, por meio de soluções tecnológicas; a *Flow* atua com modelagem de negócio, *design* de serviços e inovação; e a *Analytics* atua com geotecnologia, ciência de dados e inteligência territorial. Dado isso, neste relatório, será dada ênfase à Unidade *Tech*, mais especificamente na área de testes de software utilizando testes manuais, visto que foi a área de atuação do estágio supervisionado realizado.,

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

3 REFERENCIAL TEÓRICO

Este capítulo traz discussões realizadas por outros estudiosos (as) sobre a temática desse estudo, contribuindo para seu embasamento e articulação.

3.1 Definições de testes e sua importância

Nas últimas décadas, a engenharia de software tem crescido significativamente, buscando estabelecer técnicas, padrões, métodos e ferramentas para a produção de software. Devido ao crescente uso de sistemas baseados em computador em quase todas as áreas da atividade humana, resultou em uma demanda crescente pela qualidade e produtividade tanto do ponto de vista do processo produtivo quanto do produto que será produzido (FERNANDES, 2020).

A engenharia de software pode ser definida como a disciplina de aplicar princípios de engenharia para produzir software de alta qualidade a baixo custo. Por meio de uma série de etapas que envolvem o desenvolvimento e aplicação de métodos, técnicas e ferramentas, a engenharia de software fornece os meios para atingir esses objetivos (BERNARDO, 2008).

Desse modo, o teste de software é um elemento de um tema mais amplo, muitas vezes conhecido como verificação e validação (V&V). Verificação refere-se ao conjunto de tarefas que garantem que o software implemente corretamente uma função específica. Validação refere-se ao conjunto de tarefas que asseguram que o software foi criado e pode ser rastreado segundo os requisitos do cliente (PRESSMAN, 2021).

Ademais, as atividades de VV&T - verificação, validação e teste não se restringem ao produto final, elas podem e devem ser conduzidas durante todo o processo de desenvolvimento do software, desde a sua concepção, e englobam diferentes técnicas (DELAMARO, 2013).

Nesse sentido, as atividades agregadas em nome da garantia de qualidade de software são introduzidas ao longo do processo de desenvolvimento, incluindo atividades de VV&T, com o objetivo de minimizar a ocorrência de erros e riscos associados. Dentre as técnicas de verificação e validação, as atividades de teste são uma das mais utilizadas e constituem um dos elementos que comprovam a

confiabilidade do software, além de outras atividades como revisões de uso e especificação e verificação formal e rigorosa (DELAMARO, 2013).

Assim, as atividades de teste incluem análise dinâmica do produto e são atividades relacionadas para identificar e eliminar erros persistentes. Do ponto de vista da qualidade do processo, o teste do sistema é uma atividade essencial para subir ao Nível 3 do Modelo CMM do *Software Engineering Institute* (SEI). Ademais, o conjunto de informações obtidas a partir das atividades de teste é importante para as atividades de depuração, manutenção e estimativa da confiabilidade do software. Deve-se notar que as atividades de teste têm sido identificadas como uma das atividades mais caras no desenvolvimento de software (DELAMARO, 2013).

Para tanto, segundo Bernardo (2008), o teste de um produto de software consiste basicamente em quatro etapas: planejamento do teste, projeto do caso de teste, execução e avaliação do resultado do teste. Essas atividades devem percorrer todo o próprio processo de desenvolvimento de software, e geralmente são realizadas em três fases de teste: unidade, integração e sistema.

3.2 Principais tipos de testes

Em outros tempos, a realização de testes e a garantia da qualidade eram de responsabilidade do programador, não possuindo grande preocupação com os testes. As atividades relacionadas a testes eram simplesmente tarefas de navegar pelo código e corrigir problemas já conhecidos. Essas tarefas eram realizadas pelos próprios desenvolvedores, não existindo recursos dedicados para a realização dessas tarefas como atualmente (RIOS, 2006).

Assim, devido ao avanço tecnológico e a necessidade cada vez maior de sistemas e ferramentas que funcionem com qualidade e resposta rápida para o usuário, os testes se tornaram uma etapa muito importante no processo de desenvolvimento. Atualmente, usuários e organizações possuem uma demanda muito alta por seus softwares e ferramentas, não podendo tolerar falhas e erros que possam ocorrer, resultando em perdas e prejuízos (RIOS, 2006). Dado isso, os testes se tornaram cada vez mais importantes para as organizações, e consequentemente houve um início ao investimento na área para garantir que seus softwares ou ferramentas funcionem perfeitamente. Assim minimizando, as chances de qualquer tipo de erro ocorrer. Desse modo, pode-se dizer que a garantia de

qualidade de software é uma atividade essencial para qualquer negócio que ofereça produtos para uso de terceiros, podendo até ser utilizada como forma de marketing onde uma organização demonstra sua preocupação em garantir a qualidade de seus produtos (TRIGO, 2017).

Dessa forma, segundo Rios (2006), os testes são um processo de detecção de erros de software que verificam o comportamento do software através de diversos cenários de testes, descobrindo *bugs.*

3.2.1 Teste Dinâmico versus Teste Estático

Os estudos realizados por Delamaro (2013), propuseram duas abordagens de teste baseadas no metamodelo. A primeira abordagem denominada estática, analisa caminhos de navegação ou fluxo de informações fornecidas pelo usuário; a segunda abordagem, chamada de dinâmica, avalia a aplicação por meio da execução de casos de teste. Para as pesquisas implementadas por Pressman (2021), o teste estático é uma técnica de verificação de software focada na revisão, não em testes executáveis.

Logo, alguns exemplos de testes estáticos propostos por Delamaro (2013), são:

a) Páginas inalcançáveis – páginas disponíveis no servidor que não são alcançadas da página inicial;
b) Páginas fantasmas – páginas associadas a ligações pendentes, ou seja, que indicam páginas inexistentes;
c) *Frames* alcançados – *frames* nos quais páginas podem ser carregadas.

O teste dinâmico permite revelar defeitos relacionados aos fluxos de controle e de dados entre as páginas da aplicação Web. O caso de teste é composto de uma sequência de páginas da aplicação associadas aos dados de entrada. A sequência de páginas é gerada por meio da computação de expressão de caminho, representação algébrica dos caminhos de um grafo (DELAMARO, 2013).

Nessa concepção, dá-se que o teste dinâmico para sistemas de IA (Inteligência Artificial) é uma técnica de validação que exercita o código-fonte com casos de teste. A intenção é mostrar que o sistema de IA se conforma com os comportamentos especificados por especialistas humanos (PRESSMAN, 2021).

Nesse sentido, observa-se que os testes Dinâmicos, Delamaro (2013), descrevem:

a) Teste de Página (*page testing*) – cada página na aplicação deve ser visitada pelo menos uma vez em algum caso de teste;
b) Teste de Ligação (*hiperlink testing*) – cada ligação de cada página na aplicação deve ser percorrida pelo menos uma vez.

Assim, em geral, dividem-se as atividades de VV&T em estáticas e dinâmicas. As estáticas são as que não requerem a execução ou mesmo a existência de um programa ou modelo executável para serem conduzidas. As dinâmicas são as que se baseiam na execução de um programa ou de um modelo (DELAMARO, 2013).

3.2.2 Teste Funcional e Não funcional

O teste funcional visa propor e especificar casos de teste, considerar possíveis mudanças e garantir que todos os requisitos descritos na especificação funcional, documentação contendo todas as regras de negócios, comportamento esperado ou não intencional e outros comportamentos de software sejam atendidos com sucesso (LESSA; LESSA JÚNIOR, 2009). Isso requer um profundo conhecimento das regras de negócio do sistema para garantir que sejam devidamente consideradas nos cenários que serão testados (FUKUSAWA *et al.,* 2015). Se detalhes incorretos de regras de negócios não forem descobertos durante a execução do teste, o sistema pode não se comportar conforme o esperado (PRESSMAN, 2011).

Dada a perspectiva, a ideia é garantir que não haja diferenças entre os requisitos funcionais e o comportamento do software construído (FUKUSAWA *et al.,* 2015). Esses testes podem ser executados validando as seguintes situações:

a) As pré-condições de uma transação de negócios;
b) O fluxo de dados de uma transação de negócios;
c) O cenário primário de uma transação de negócios;
d) Os cenários alternativos de uma transação de negócios;
e) Os cenários de exceção de uma transação de negócios;
f) As pós-condições de uma transação de negócios.

Para tanto, é necessário conhecer um pouco sobre os requisitos não funcionais do software primeiramente para que depois, verifique os conceitos relacionados a testes não funcionais. Dessa forma, de acordo com os estudos realizados por Santana (2022), os requisitos não funcionais, também denominados requisitos de qualidade, incluem tanto limitações no produto (desempenho, confiabilidade e segurança) como limitações no processo de desenvolvimento (custos, métodos a serem adotados no desenvolvimento e componentes a serem reutilizados).

Ainda segundo Santana (2022), os testes não funcionais são aqueles que visam apenas os requisitos não funcionais como bases dos testes. Estes testes são divididos em alguns subtipos de testes. São eles: Teste de Desempenho, Teste de Carga e Teste de Estresse, Teste de Usabilidade.

3.2.3 Teste de Desempenho, Carga e Volume

O teste de carga é projetado para simular situações incomuns no uso do software, analisando o comportamento caso algo não planejado possa acontecer (FUKUSAWA *et al.,* 2015). A ideia é avaliar como todo o conjunto de soluções lida com as mudanças contínuas de processamento (BERNARDO, 2008). Esses testes são recomendados para software de missão crítica. Pode ser executado da seguinte forma:

a) Elevando e reduzindo sucessivamente o número de transações simultâneas;
b) Aumentando e reduzindo o tráfego de rede;
c) Aumentando o número de usuários simultâneos.

Dado isso, o teste de volume consiste em determinar as restrições de processamento e infraestrutura da solução (TRIGO, 2017), além dessa função, também visa analisar o desempenho do software com cada aumento na carga de processamento. A ideia é entender as limitações da solução e avaliar a proximidade que esses requisitos estão dessa limitação (BERNARDO, 2008). Nessa concepção, pode ser executado da seguinte forma:

a) Aumentando sucessivamente o volume de transações;

b) Aumentando sucessivamente o tamanho de arquivos a serem processados.
c) Aumentando sucessivamente o volume de consultas.

Retendo-se ao teste de performance, esse tem por objetivo, analisar como o sistema se comporta diante de situações previstas de pico máximo e concorrência. Pressman (2021), afirma que o critério de sucesso estabelecido é empregar o volume de transações e tempo de resposta obtidos nos testes e compará-los com os valores-limite especificados. Podendo ser executados da seguinte forma:

a) Validar todos os requisitos de desempenho identificados;
b) Simular n% de tráfego de rede;
c) Combinar todos esses elementos.
d) Simular *n* usuários acessando a mesma informação, de forma simultânea;
e) Simular n usuários processando a mesma transação, de forma simultânea.

3.2.4 Teste de Caixa Preta

O teste Caixa Preta foca nos requisitos funcionais do software, portanto, visa garantir que o sistema esteja em conformidade com a especificação funcional, satisfazendo assim todos os requisitos do sistema. O objetivo principal é garantir que o que foi desenvolvido seja o esperado e só precisa conhecer as possíveis entradas e saídas do programa e diferentemente do teste de caixa branca, ignora a arquitetura interna do software (PRESSMAN, 2011).

O teste de Caixa Preta procura descobrir erros nas seguintes categorias: (1) funções incorretas ou ausentes; (2) erros de interface; (3) erros nas estruturas de dados ou no acesso a banco de dados externos; (4) erros de desempenho; e (5) erros de inicialização e término (PRESSMAN, 2011, p.56).

Ao aplicar técnicas de Caixa Preta, derivamos um conjunto de casos de teste que satisfaz aos seguintes critérios: (1) casos de teste que reduzem, numa contagem que seja maior do que um (1), o número de casos de testes adicionais que devem ser projetados para se conseguir testes razoáveis; e (2) casos de teste que nos dizem algo sobre a presença ou ausência de classes de erros, em vez de um

erro associado somente ao teste específico que se tem em mãos (PRESSMAN, 2011).

3.2.5 Teste de Caixa Branca

Segundo Rosa (2017), o teste de Caixa Branca é baseado na arquitetura interna do software. São empregadas técnicas destinadas a identificar falhas na estrutura interna dos programas. Para realizar os testes, o programador deve conhecer a tecnologia utilizada pelo software e ter conhecimento suficiente da arquitetura interna do software necessário para acessar o código-fonte e a estrutura do banco de dados.

Desse modo, quando um programador realiza testes de Caixa Branca, ele deve garantir que todos os caminhos independentes dentro de um componente sejam executados pelo menos uma vez, testar todas as condições lógicas com valores verdadeiros e falsos e testar todos os *loops* para garantir a validade do componente (FUKUSAWA *et al.,* 2015).

Deste ponto de vista, erros de lógica podem ocorrer ao implementar funções, condições ou controles fora da função principal, o manuseio é bem entendido ao testar situações cotidianas, mas pode haver casos especiais que podem ser detectados no momento da execução. Executando um teste. em alguns casos, pensamos que um caminho lógico nunca será executado, quando na verdade esse é executado periodicamente. O fluxo lógico de um programa pode ser contraintuitivo, ou seja, assume-se que o fluxo de dados seguirá um caminho específico, entretanto, só é possível descobrir durante os testes. Ademais, erros de ortografia também podem ocorrer quando o mecanismo de verificação gramatical não detecta e só serão detectados no início do teste (FUKUSAWA *et al.,* 2015).

Por esses e outros motivos, a realização do teste de Caixa Branca é importante, pois somente com ele podem ser detectados erros dessa natureza. Desse modo, os testes de Caixa Preta, mesmo sendo muito bem realizados, podem não revelar esses erros (FUKUSAWA *et al.,* 2015).

3.2.6 Comparativo entre Testes de Caixa Preta e Testes de Caixa Branca

De acordo com Pressman (2021), os testes estão se tornando cada vez mais importantes e aparecendo nas organizações. Identificamos alguns dos tipos de teste usados e suas funções. O teste de Caixa Branca concentra-se principalmente na estrutura interna do sistema (FERNANDES, 2020). Esse tipo de teste é baseado no código-fonte e se esforça para cobrir casos de teste e garantir a funcionalidade dos componentes de software. Ao contrário do teste de Caixa Preta, que busca garantir que o software seja desenvolvido de acordo com as especificações funcionais, ou seja, desenvolvido conforme requisitos esperados (FERNANDES, 2020). Dessa maneira, as técnicas de Caixa Branca são vistas como complementares às técnicas de Caixa Preta, e as informações obtidas pela aplicação desses padrões são consideradas relevantes para atividades de manutenção, depuração e confiabilidade de software.

Ainda segundo os estudos de Pressman (2021), o teste de Caixa Branca comumente é feito em técnicas de teste de unidade e teste de integração. O teste de unidade é feito na fase inicial do processo de teste e os componentes são testados individualmente. Já nos testes de integração, que é quando os componentes são construídos ou modificados, são testados em conjuntos, na qual devem manter a compatibilidade com outros componentes existentes. Esses testes geralmente são de responsabilidade dos desenvolvedores do sistema, pois eles sabem que o código gerado está consistente (FERNANDES, 2020). Pressman (2011), explana que o teste de caixa preta é normalmente realizado em técnicas de teste funcional e teste de integração.

3.2.7 Teste de Unidades

Teste de unidade considerado no presente trabalho, de acordo com Somerville (2020, p. 59), “é o processo de testar componentes de programa, como os métodos ou as classes”. O teste unitário faz parte da fase inicial do processo de verificação e é usado para testar o aplicativo, realizando os testes nos componentes individualmente. O teste é guiado com base na estrutura interna do componente, descrito como teste de caixa branca (PRESSMAN, 2021).

Segundo Souza (2016), para encontrar *bugs*, é necessário testar em caminhos de controle importantes, guiado pela descrição detalhada do projeto. A relativa complexidade dos testes e os erros detectados por esses, são limitados pelo

estreito escopo estabelecido para a realização de testes unitários. Sendo assim, por ser sempre baseado em uma Caixa Branca, esse teste pode ser realizado em vários componentes em paralelo.

De acordo com Somerville (2020), sempre que possível, o teste de unidade deve ser automatizado, com um *framework* de automação de teste. Estes frameworks de teste de unidade fornecem classes de teste genéricas que podem ser estendidas para criar casos de teste específicos. Após, eles podem executar todos os testes que foram implementados e informar por meio de uma interface gráfica com o usuário, o sucesso ou não dos testes.

No teste de unidade, interfaces e componentes são testados para garantir que as informações fluam corretamente para dentro e para fora da unidade do programa em teste. As estruturas de dados são verificadas para garantir que os dados armazenados temporariamente não tenham sido modificados por qualquer motivo durante qualquer etapa do processamento do algoritmo (PRESSMAN, 2021).

Portanto, caminhos independentes são executados para garantir que todas as instruções do componente sejam executadas pelo menos uma vez, e todos os caminhos de tratamento de erros também são testados para garantir que os erros sejam tratados e que o programa os esteja tratando corretamente (PRESSMAN, 2021).

3.2.8 Teste de Integração

O teste de integração é o próximo passo após o teste de unidade, pois quando os componentes são construídos ou modificados, eles devem manter a compatibilidade com outros componentes existentes (PRESSMAN, 2021). Neste teste, vários componentes são colocados para executá-los, testar e verificar sua interface. Se todos os componentes são testados com sucesso e funcionam individualmente, por que eles não funcionam juntos?

Sendo assim, Pressman (2021), argumenta que o problema de juntá-los é que existe uma interface, pois os dados podem ser perdidos, eventos inesperados adversos podem ocorrer, as funções de cada componente podem não ter os resultados esperados da função principal quando combinadas, e, vários outros problemas. Há uma tendência de realizar testes por meio de integração não incremental, ou seja, construir programas usando uma abordagem *big bang*.

Neste tipo de abordagem, todos os componentes são pré-montados e todo o programa é testado. Na grande maioria dos casos, os resultados são confusos porque um conjunto de erros é descoberto e sua correção é complicada pela dificuldade de isolar a causa ao combinar componentes. Mesmo depois que os erros são corrigidos, novos erros surgem e o processo de correção de erros é repetido com a mesma dificuldade (PRESSMAN, 2011).

A integração incremental é o oposto da abordagem *big bang*, pois nesse caso, o programa é construído em pequenos segmentos, e é possível isolar e corrigir erros de forma mais fácil além de realizar testes mais completos nas interfaces. A integração incremental permite então, a aplicação de uma abordagem sistemática ao teste e podem-se utilizar diferentes abordagens como a integração *top-down* e integração *bottom-up* (PRESSMAN, 2011).

Em uma abordagem de cima para baixo, o ciclo de integração segue a mesma linha de pensamento do desenvolvimento estrutural, com as unidades de nível mais alto sendo criadas e passando por um processo de refinamento e decomposição contínuos até atingirem o nível estrutural mais baixo.

Segundo Rezende (2006), na abordagem *bottom-up*, os componentes no nível mais básico da arquitetura da aplicação são testados em conjunto com controladores desenvolvidos para simular as interações e interfaces dos componentes. Assim, à medida que o teste é executado com sucesso, o componente substitui o controlador e cria um novo controlador de teste para simular a interface e a interação. Repita este processo até que não seja possível atingir mais níveis de integração.

3.2.9 Teste de Sistema

De acordo com as pesquisas realizadas por Delamaro (2013), depois que se tem o sistema completo, com todas as suas partes integradas, inicia-se o teste de sistema. O objetivo é verificar se as funcionalidades especificadas nos documentos de requisitos estão todas corretamente implementadas.

Já para Sommervile (2020), o teste de sistema durante o desenvolvimento envolve a integração dos componentes para criar uma versão do sistema e depois testar o sistema integrado. Esse teste verifica se os componentes são compatíveis, se interagem corretamente e se transferem os dados certos no momento certo por

meio de suas interfaces. Desse modo, o autor comenta que o teste de sistema deve se concentrar em testar as interações entre os componentes e objetos que compõem um sistema.

Na maioria dos sistemas, é difícil saber o quanto de teste de sistema é essencial e quando se deve parar de testar. É impossível testar exaustivamente, ao ponto em que toda a sequência possível de execução do programa seja testada. Portanto, os testes devem se basear em subconjuntos possíveis de casos de teste (DELAMARO, 2013).

Aspectos de correção, completude e coerência devem ser explorados, bem como requisitos não funcionais como segurança, performance e robustez. Muitas organizações adotam a estratégia de designar uma equipe independente para realizar o teste de sistemas (DELAMARO, 2013).

3.4 Automação de testes

A busca por maior produtividade com eficiência e qualidade é meta de toda empresa que tem o crescimento como objetivo. Em uma empresa de software essa busca vai além, pois alcançar mais eficiência passa pela investigação e implementação das evoluções, quase diária, de novas tecnologias que possibilitam melhorar a qualidade de seus produtos. A automação veio para auxiliar o setor de testes dessas empresas a crescer justamente em produtividade e eficiência, sem de forma alguma renunciar à qualidade (FUKUSAWA *et al.,* 2015).

Existem duas razões originais para automatizar o teste de software: O primeiro argumento tem a ver com o alto crescimento de sistemas e aplicativos onde se torna difícil controlar e criar testes manuais. A segunda razão é baseada na alta complexidade do software que está sendo desenvolvido atualmente. Tendo a situação como hipótese, e levando em conta os pontos levantados, em alguns sistemas a única solução para a realização de testes é utilizar a automatização dos testes, considerando o tipo de testes que serão realizados e a funcionalidade a realizar, pelo que por vezes executar testes manuais não é uma opção (FUKUSAWA *et al.,* 2015).

Sendo assim, o profissional que trabalha com automação de testes, precisa entender da programação do sistema desenvolvido e a lógica de implementação que existe por trás do sistema, sendo que anteriormente, essa necessidade pertencia

aos programadores. Outra necessidade é entender as funcionalidades e características dos sistemas que irão utilizar a automação, pois muitas vezes possuem linguagens proprietárias para desenvolvimento (SANTOS, 2022).

Para realizar essas automações descritas existem ferramentas e frameworks que auxiliam o profissional nessa etapa. Segundo Santos (2022), as ferramentas de automação de testes funcionam com a execução de um conjunto de instruções redigidas em uma linguagem de programação, que na maioria das vezes é chamada de linguagem *script*. As instruções para as ferramentas são muito detalhadas como por exemplo, informações sobre as entradas específicas, ordem das entradas, valores específicos utilizados nas entradas e saídas esperadas. Esse movimento, portanto, pode deixar *scripts* detalhados suscetíveis a alterações no software em teste (SUT, na sigla em inglês), particularmente quando a ferramenta interage com a interface gráfica de usuário (GUI, na sigla em inglês).

No contexto do desenvolvimento de um sistema, um software ou uma aplicação, existem diversas ferramentas que podem ser utilizadas para realização da automação de testes. Entre elas estão o *Selenium, Cucumber, Apium, TestComplete, Ranorex, Robotium*, entre outras. Cada uma possui elementos e funcionalidades que as tornam específicas para determinados tipos de teste e ambientes.

Quadro 1 - Ferramentas de Automação mais utilizadas

Ferramenta	Ambiente	Linguagem
SELENIUM	Web	Java, Perl, JavaScript, PHP, Python, C#, Ruby
CUCUMBER	Web	Ruby,Java, NET, Scala, Groovy
TESTCOMPLETE	Desktop, Mobile, Web	JavaScript, Python, JScript, Delphi, C++ e C#
TELERIK TS	Mobile, Web	Angular, Asp.Net, HTML5, JavaScript, Ajax, Silverlight, Ruby, PHP, VB.Net e C#
ROBOTIUM	Mobile	Java
RANOREX	Desktop, Mobile, Web	Não utiliza linguagem
APIUM	Mobile	Java
TESTINGWHIZ	Web, Mobile, BDD	Não utiliza linguagem

Fonte: Castro, 2022, p.163

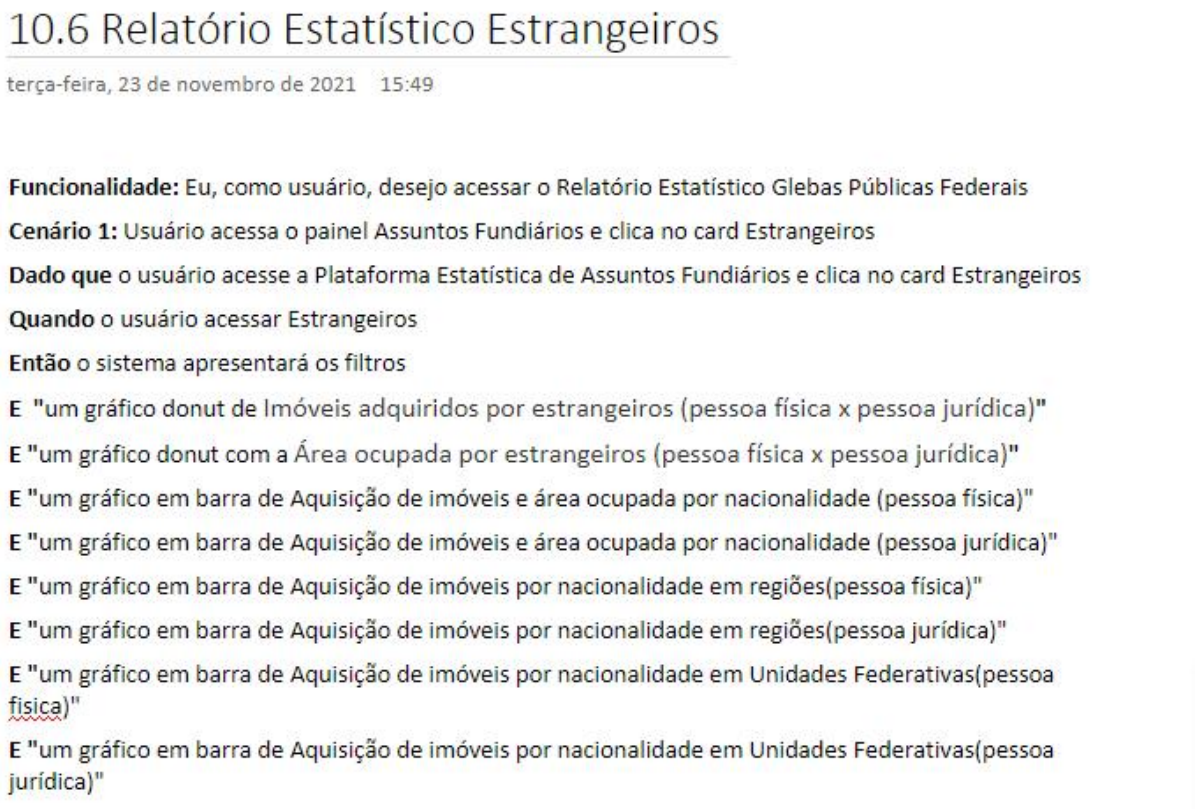
3.5 Cenários de teste

Um cenário de teste é uma descrição de um objetivo que o usuário pode encontrar ao utilizar o programa (VICTORINO *et al.,* 2009).

Na figura 1 é exibido um cenário escrito para uma das telas do projeto Observatório. Nele é especificado os passos para realização dos testes.

O conhecimento de causa e as habilidades em testes podem facilitar os testadores a transformar os cenários em ideias concretas, escolher a abordagem mais lógica e rodar os testes que podem extrair os problemas importantes (FERNANDES, 2020).

Figura 1 - Exemplo de cenário desenvolvido



10.6 Relatório Estatístico Estrangeiros

terça-feira, 23 de novembro de 2021 15:49

Funcionalidade: Eu, como usuário, desejo acessar o Relatório Estatístico Glebas Públicas Federais

Cenário 1: Usuário acessa o painel Assuntos Fundiários e clica no card Estrangeiros

Dado que o usuário acesse a Plataforma Estatística de Assuntos Fundiários e clica no card Estrangeiros

Quando o usuário acessar Estrangeiros

Então o sistema apresentará os filtros

E "um gráfico donut de Imóveis adquiridos por estrangeiros (pessoa física x pessoa jurídica)"

E "um gráfico donut com a Área ocupada por estrangeiros (pessoa física x pessoa jurídica)"

E "um gráfico em barra de Aquisição de imóveis e área ocupada por nacionalidade (pessoa física)"

E "um gráfico em barra de Aquisição de imóveis e área ocupada por nacionalidade (pessoa jurídica)"

E "um gráfico em barra de Aquisição de imóveis por nacionalidade em regiões(pessoa física)"

E "um gráfico em barra de Aquisição de imóveis por nacionalidade em regiões(pessoa jurídica)"

E "um gráfico em barra de Aquisição de imóveis por nacionalidade em Unidades Federativas(pessoa fisica)"

E "um gráfico em barra de Aquisição de imóveis por nacionalidade em Unidades Federativas(pessoa jurídica)"

Fonte: YouxGroup (2022)

Em um ambiente de desenvolvimento de software, devem existir documentações com descrições do sistema e informações sobre ele. Nesta documentação, utilizando boas práticas, é importante que existam cenários de testes (PIMENTEL, 2005).

3.6 SCRUM

Segundo Schwaber e Sutherland (2020), Scrum é um método ágil de gerenciamento de projetos que ajuda pessoas, equipes e organizações a criar valor por meio de soluções adaptativas para problemas complexos, e consiste em artefatos, eventos e equipes Scrum. Os artefatos do Scrum representam trabalho ou valor, sendo projetados para maximizar a transparência das principais informações e são separados em três, sendo eles, *Product Backlog, Sprint Backlog* e Incremento. Dessa maneira, cada artefato contém um compromisso para garantir que ele forneça informações que aumentem a transparência e o foco contra o qual o progresso pode ser medido: Para o Product Backlog, sendo a Meta do produto; para o Sprint Backlog, a Meta da Sprint e para o Incremento, é a Definição de Pronto (SCHWABER; SUTHERLAND, 2020).

Para Schwaber e Sutherland (2020), o Product Backlog é uma lista ordenada e emergente do que é necessário para melhorar o produto, e única fonte de trabalho realizado pelo Scrum Team. Dado isso, seu compromisso é a Meta do Produto, que descreve um estado futuro do produto que pode servir como um alvo para o Scrum Team planejar. O Sprint Backlog é composto pela Meta da Sprint (por que), o conjunto de itens do Product Backlog selecionados para a Sprint (o que), bem como um plano de ação para entregar o Incremento (como). E por fim, um incremento é um trampolim concreto em direção a Meta do Produto. Cada incremento é adicionado a todos os anteriores e completamente verificados, garantindo que todos funcionem juntos (SCHWABER; SUTHERLAND, 2020).

Nessa concepção, segundo Schwaber e Sutherland (2020), um Time Scrum consiste em Scrum Masters, Product Owners e Desenvolvedores. O Scrum Master é responsável por construir o Scrum conforme definido no Guia do Scrum, e ele faz isso ajudando todos na equipe e organização Scrum a entender a teoria e a prática do Scrum.

Desse modo, o Product Owner é responsável por maximizar o valor do produto resultante do trabalho do Scrum Team. Por fim, os desenvolvedores são as pessoas do Scrum Team que estão comprometidas em criar qualquer aspecto de um incremento utilizável a cada Sprint. Eventos são usados no Scrum para criar regularidade e minimizar a necessidade de reuniões que não são definidas no Scrum.

Cada evento no Scrum é uma oportunidade formal para revisar e ajustar um artefato Scrum. Esses eventos são projetados especificamente para obter a transparência necessária (SCHWABER; SUTHERLAND, 2020). A Sprint é um evento de duração fixa de um mês ou menos para criar consistência de trabalho, cada Sprint pode ser considerado um projeto curto em que algo de valor é gerado ao fim. A Sprint Planning inicia a Sprint ao definir o trabalho a ser realizado na Sprint. Este plano resultante é criado pelo trabalho colaborativo de todo o Scrum Team (SCHWABER; SUTHERLAND, 2020).

Nessa sequência, o Daily Scrum é uma atividade de no máximo 15 minutos para o Scrum Team Developer verificar o progresso em relação ao Sprint Goal e ajustar o Sprint Backlog conforme necessário para o próximo trabalho planejado. O objetivo da Revisão do Sprint é examinar os resultados e identificar ajustes futuros. As equipes Scrum apresentam os resultados de seu trabalho para as principais partes interessadas e discutem o progresso em direção às metas do produto. Em relação à Sprint Retrospective, seu propósito é planejar maneiras de aumentar a qualidade e a eficácia (SCHWABER; SUTHERLAND, 2020).

4 ATIVIDADES DESENVOLVIDAS

No presente capítulo, serão apresentadas as atividades desenvolvidas durante o período do estágio bem como as tecnologias utilizadas. Para um melhor entendimento, esta seção é quebrada nas seguintes estruturas: Período de Adaptação e Projeto Inicial - SEMAD, Planejamento com o Scrum, Criação de Cenários de Testes e Execução dos Cenários no Projeto Observatório.

4.1 Período de adaptação

O período de estágio se deu do dia 01 de setembro de 2021 e até 29 de abril de 2022. Logo na primeira semana de trabalho, realizou-se um treinamento para os novos estagiários sobre o funcionamento da empresa, sobre metodologias, e como eram realizadas as atividades. Ademais, esse treinamento abordou a realização de um curso de Scrum e a retirada de um certificado, um curso da Udemy sobre Testes de Software e dois Onboardings de apresentação geral da empresa.

Após, na segunda semana, fui direcionada a um projeto chamado SEMAD que consiste em um aplicativo móvel para a realização de registros e acompanhamento de alertas e, ou denúncias ambientais. Com essa função, fui instruída nesse projeto pelo período de 3 semanas.

Nesse sentido, para iniciar as atividades no projeto, uma equipe especializada da empresa, composta por um Analista de QA, um PO e um desenvolvedor realizaram um treinamento de GitLab. Desse modo, Segundo Palestino (2015), Git é um sistema de código aberto para controle de versões, comumente utilizado por desenvolvedores de software, no entanto, o sistema controlará a versão de qualquer tipo de arquivo. Nesse sentido, é essencial que se crie um histórico de alterações de arquivo para que você saiba quais alterações foram feitas e por quem, e tenha total liberdade para voltar às versões anteriores. Assim, o uso do GitLab foi primordial para verificação das tarefas existentes, execução das atividades demandadas e reporte de situações não esperadas ocorridas através de *issues* que são utilizadas no quadro do projeto para definir as Demandas da Sprint.

Nessa perspectiva, foram realizados testes manuais no APP do SEMAD e escritas de *issues* para reportar os erros encontrados. Para tanto, as atividades realizadas nesse projeto foram orientadas por um Analista de Qualidade Pleno da

organização, além de tornar-se um momento também de treinamento para habituar às atividades da empresa.

Logo após esses treinamentos fui direcionada ao projeto que foi mantido até atualmente, após efetivação. Esse projeto é intitulado “o projeto da Agropecuária Brasileira”, e integra e disponibiliza uma grande quantidade de dados e informações da agricultura e pecuária do país e do mundo.

Foram trabalhadas diariamente 6 horas de estágio, no período de 8 às 12h e em seguida no horário de 14h às 16h, podendo haver flexibilidade. As horas contabilizadas somam 960 horas de carga horária total, durante os 8 meses de atividades. E nesse sentido, o objetivo do estágio foi aprender e introduzir na área para a carreira de Analista de Qualidade de Software, na qual a estudante já tinha realizado uma iniciação científica na área de testes de Carga em sistemas WEB.

Ademais, como supervisor houve um Analista de Qualidade Pleno, que auxiliava nas tarefas, ensinava e realizava reuniões mensais para instruir e feedbacks sobre a realização das atividades do estágio.

4.1.1 Projeto Inicial - SEMAD

O SEMAD Alerta é um módulo da Secretaria de Estado de Meio Ambiente e Desenvolvimento Sustentável (SEMAD) do Estado de Goiás. O módulo possui como finalidade proporcionar ao usuário o acesso remoto ao aplicativo e à página WEB, para a realização de registros e acompanhamento de alertas ou denúncias ambientais.

Figura 2 - Tela inicial do aplicativo SEMAD

Fonte: SEMAD (2022)

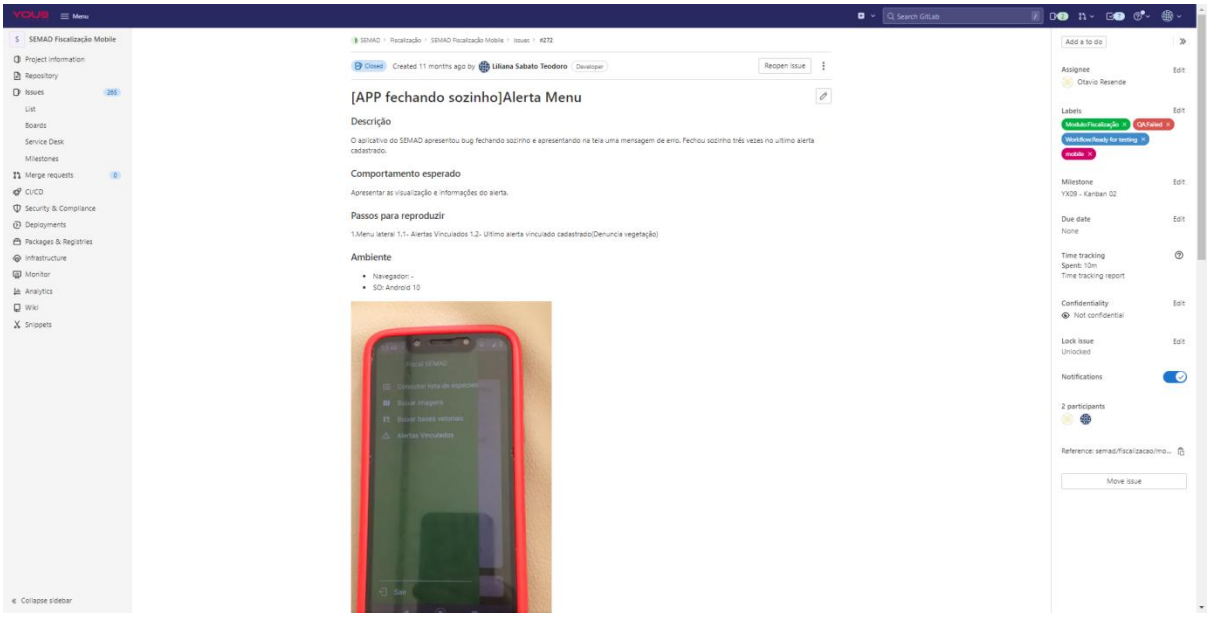
O período de atividades realizadas no projeto citado acima foi de 3 semanas. O início nesse projeto foi importante para ambientação cotidiana na empresa e aprendizado na introdução na área de QA, como por exemplo, reportar erros e *bugs* e como estruturar *issues* de qualidade. Neste momento foi o início da experiência no âmbito da Qualidade de Software.

Desse modo, através do Discord - aplicativo de bate-papo de voz – através de mensagens e compartilhamento de tela, foi realizada toda a comunicação. Ou seja, as equipes permanecem em chamadas de voz e esse momento tona-se então “local de trabalho”.

Dando seguimento, transcorreram, inicialmente, atividades de testes supervisionados de *issues* desenvolvidas do sistema para verificar se a tarefa estava funcionando de acordo com as especificações. Desse modo, caso a tarefa esteja válida, ela é aprovada e o tempo utilizado nos testes é adicionado ao tempo de realizações. Entretanto, se durante a realização da tarefa, encontram-se comportamentos não esperados pelo sistema, é então necessário retornar a *issue* para correção. Assim, é necessário, na *issue* retornada, a descrição minuciosa dos passos para reproduzir tal erro, como onde foi encontrado e como foi encontrado,

para facilitar aos desenvolvedores encontrarem o erro. Na figura 3 exibe um exemplo de uma das primeiras issues.

Figura 3 - Exemplo de *issue* no App do SEMAD



Fonte: Observatório (2022)

A *issue* descrita na figura 3 possui as informações para reproduzir o erro encontrado, além de um vídeo gravando o comportamento da tela no momento da falha relatada. Para mais, foi aprendido sobre definição dos erros e priorização de cada um deles. As *issues* no GitLab nos projetos da YouxGroup são marcadas com *labels* que identificam o que está ocorrendo e qual a prioridade desse acontecimento relatado. Exemplos de *Labels* do processo de Qualidade:

a) QA:Failed: *Label* que indica na *issue* que existe um problema para a Qualidade do sistema.
b) “ERRO”: Situação que retorna uma resposta errada, não esperada, do sistema.
c) “P1”: Ocorrência que impede o uso do sistema; não existe forma de contorná-la.
d) “P2”: Ocorrência não impeditiva, ou seja, pode ser contornada pelo usuário e é relativa ao funcionamento do sistema (não cumprimento de algum requisito).

e) “P3”: Ocorrências relativas à interface, como problemas de renderização, e cores.
f) “P4”: Ocorrência que deve ser verificada, mas tem baixa prioridade, podendo serem realizadas outras implementações.
g) “Melhoria”: Após e durante realizar testes, podem ser encontradas formas de melhor utilização do sistema, que facilita a usabilidade para o usuário.

Essas *labels* devem ser utilizadas juntas, de acordo com o melhor contexto da situação existente que está sendo descrita. O seu uso é de grande relevância nas atividades do projeto, pois ajudam a verificar quais devem ser desenvolvidas com precedência.

Nessa concepção, as atividades realizadas no projeto SEMAD, no início do estágio, foram extremamente importantes para o entendimento da contextualização da área de Qualidade, e também para introdução no contexto profissional.

4.2 Planejamento com o Scrum

O papel da pessoa Analista de Qualidade nos ritos Scrum é muito importante e fundamental. Durante as *plannings* eram realizadas, juntamente com o PO, observações relevantes para entendimento geral das tarefas a serem desenvolvidas. Entendendo bem as demandas da *Sprint*, podem ser feitas estimativas mais corretas do tempo de possível desenvolvimento de uma nova *feature* (novas implementações no sistema).

Durante a *planning* eram estimadas também horas para realização de testes. O gitLab foi utilizado para apontar as *issues* com as estimativas de tempo previstas pelo time e as descrições de atividades para serem desenvolvidas.

Já no início da sprint eram escritos os cenários de testes de acordo com as regras de negócio e de acordo com o protótipo já desenvolvido anteriormente pela equipe de produto.

4.3 Criação dos Cenários de Testes

Os cenários de testes são atividades descritas em um documento que detalham os passos a serem seguidos para realização estruturada de um teste. A

partir das regras de negócio, são avaliadas as atividades e métodos para a escrita desses casos de teste. Esses são escritos, a fim de que quando chegar o momento que o sistema está em ambiente de testes, os casos serão validados. Para escritas dos cenários foi utilizado o *Microsoft SharePoint.*

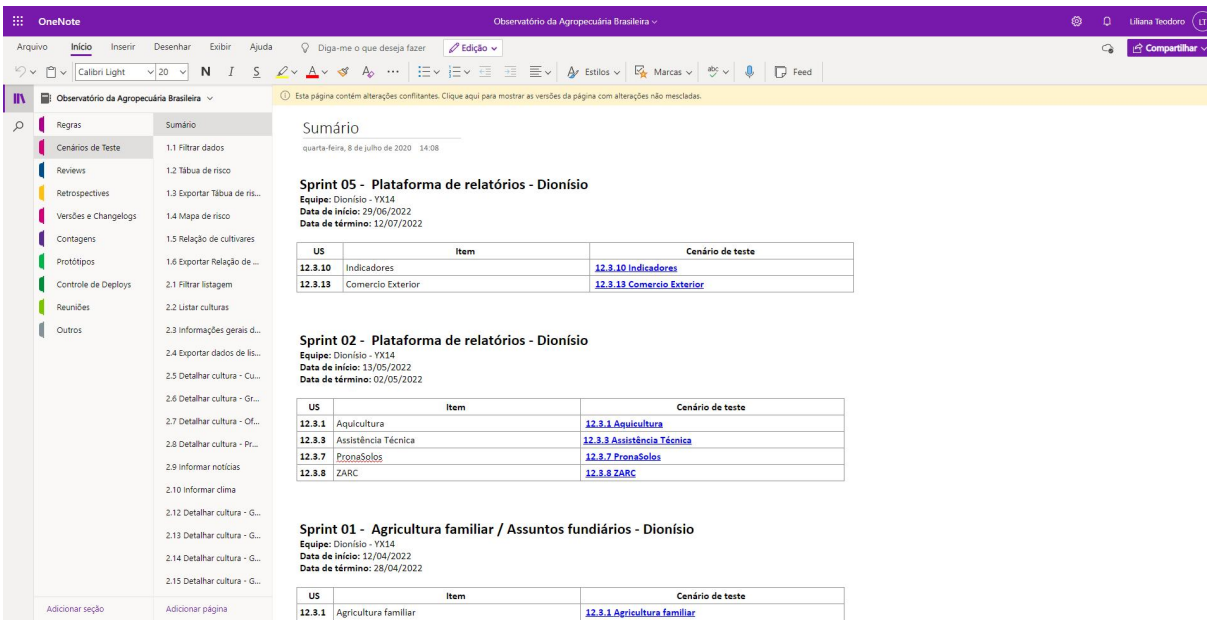
O *Microsoft SharePoint* é uma plataforma multifacetada de colaboração de grupos de aplicativos cruzados da *Microsoft*. Primeiro, é uma intranet móvel e inteligente que pode ser usada como portal corporativo para ajudar a gerenciar informações, projetos, fluxos de trabalho e equipes. Suas características o tornam uma ótima ferramenta para o desenvolvimento de negócios (LATINNE, 2021).

Ademais, é possível compartilhar aplicativos e recursos no portal do Microsoft SharePoint para notificar os colaboradores sobre atualizações e notícias. Por meio da plataforma os colaboradores podem de forma rápida e instantânea compartilhar todas as informações com a equipe de forma segura e isso ajuda a orientar a tomada de decisões. Através dessa ferramenta e a transformação dos processos, é possível acelerar a produtividade e as aprovações (LATINNE,2021).

Nesse sentido, o uso do Shrepoint no projeto foi de extrema importância para compartilhamento entre os colaboradores da escrita dos cenários de testes.

Na figura 4, mostra o sumário com os links para cada cenário descrito.

Figura 4 - Exemplo da utilização do Sharepoint no projeto Observatório



Fonte: Próprio autor (2022)

O programa também possui direitos de acesso para garantir que apenas os colaboradores envolvidos no projeto ou tarefa participem. Dessa forma, a edição e a comunicação ficam centralizadas dentro da equipe de cada requisito, garantindo maior qualidade na entrega (LATINNE,2021).

4.4 Execução dos Cenários

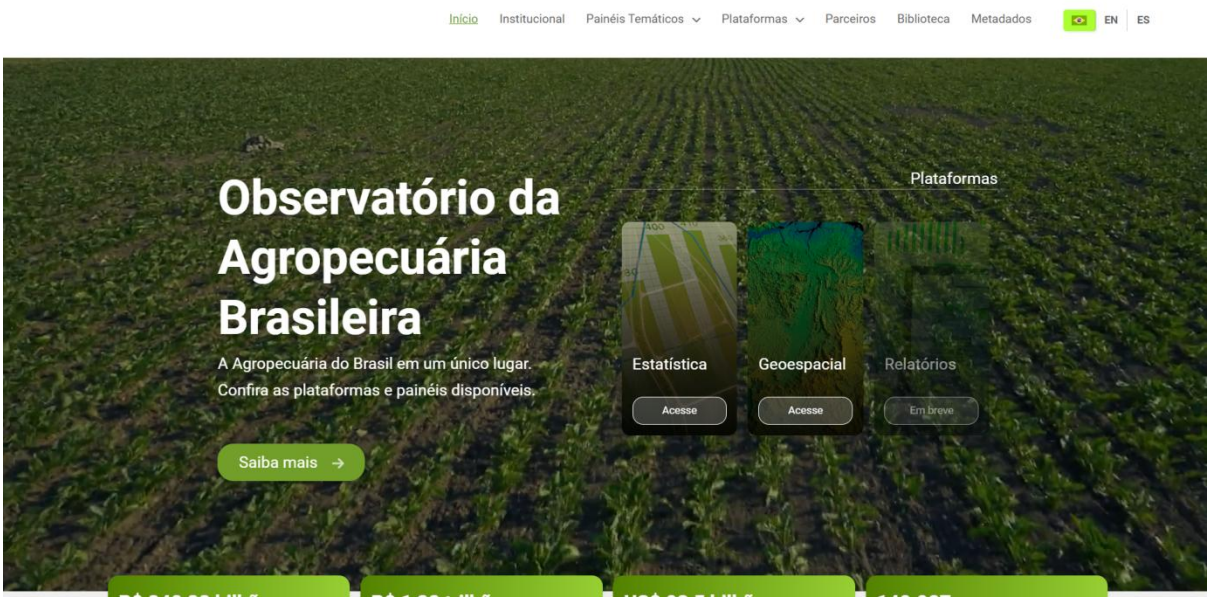
Os testes realizados nos sistemas da empresa YouxGroup são realizados em ambientes de desenvolvimento pelos Analistas de Qualidade, para que, em seguida e após aprovados, serem enviados para ambiente de homologação, no qual os clientes irão validar se também estão de acordo com o esperado. Desse modo, após aprovados por Analistas de Qualidade e pelos clientes, a funcionalidade do sistema poderá ser enviada para ambiente de produção, a qual fica disponível para os usuários acessarem.

4.4.1 Observatório

O sistema do Observatório foi o projeto no qual foi desenvolvida a maior parte das atividades durante o estágio. Esse é um sistema do Ministério da Agricultura, Pecuária e Abastecimento do Brasil e é uma iniciativa para valorizar a Agropecuária Brasileira

O Observatório da Agropecuária Brasileira é uma inovadora solução tecnológica que sistematiza, integra e disponibiliza um gigantesco conjunto de dados e informações da agricultura e pecuária do país - e também mundiais -, provendo subsídios aos processos de tomada de decisão e de formulação de políticas públicas, ao tempo em que revela este virtuoso setor e oportuniza o equacionamento de seus principais desafios (OBSERVATÓRIO, 2023).

Figura 5 - Tela inicial do site Observatório



Fonte: Observatório (2023)

No sistema existem duas vertentes que juntas, o compõem. São elas: a plataforma Estatística e a plataforma GeoEspacial.

A Plataforma Estatística é um ambiente do Observatório que disponibiliza informações de interesse da Agropecuária Brasileira, por meio de dados numéricos, tabulares e representações gráficas acerca dos diversos temas abordados. A Plataforma está organizada com filtros por período, estratificações em nível nacional, estadual e municipal, além de dados quantitativos e qualitativos. Os dados são disponibilizados por fontes oficiais, abertos ao público, com uma abordagem profunda das variáveis apresentadas, em diferentes níveis, permitindo ao usuário melhor interação e estudo das informações selecionadas (OBSERVATÓRIO, 2023).

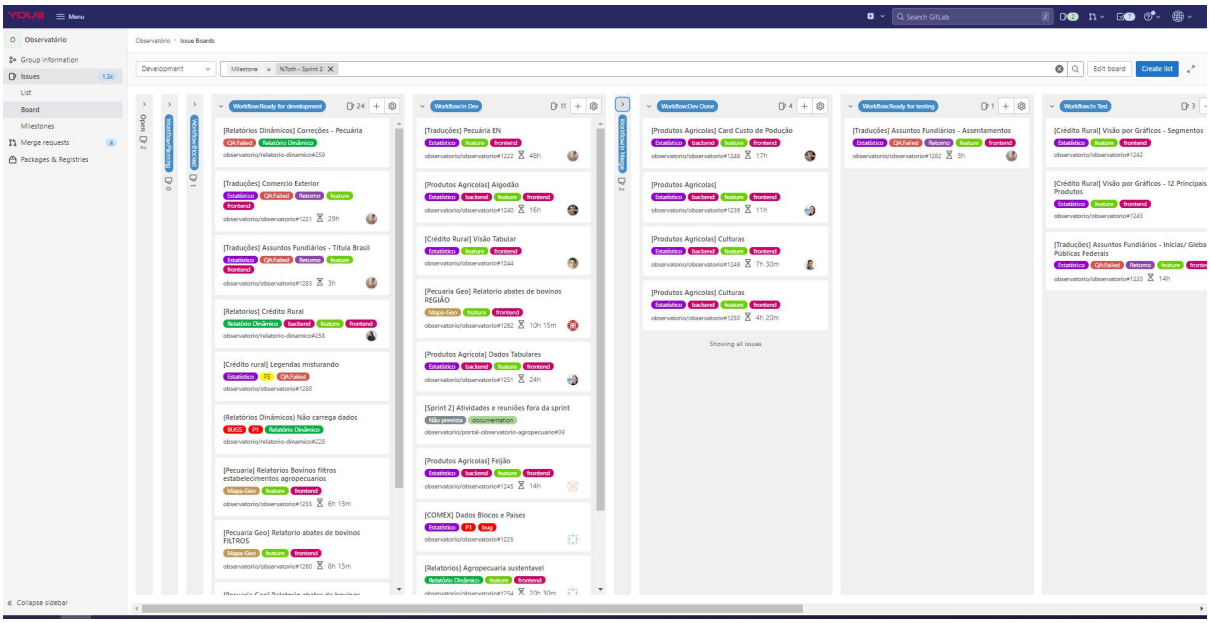
Já a Plataforma Geoespacial é um ambiente do Observatório dedicado à integração de dados e informações territoriais de interesse da agropecuária brasileira, por intermédio dos quais os dados geoespaciais podem ser visualizados e construídos de acordo com a necessidade e interpretação dos usuários (OBSERVATÓRIO, 2023).

Nesse sentido, foram realizadas atividades de teste na vertente do sistema da Plataforma Estatística. Sendo assim, no projeto do observatório, foram realizados diversos testes, em painéis da plataforma estatística, desenvolvidos durante as *sprints* em que foi realizado o estágio.

Após o planejamento da *Sprint*, com as demandas definidas e estimadas, pode-se realizar o desenvolvimento de cada uma dessas *issues* e nesse momento dá - se início também ao desenvolvimento dos cenários de testes.

Dessarte, os testes foram realizados em cada nova tarefa desenvolvida após realização de um merge[2] dessa *issue* para ambiente de teste e de comunicar-se com o desenvolvedor para confirmar a situação de pronta dessa *issue*, para testar. Durante a *sprint*, à medida queocorriam as entregas das tarefas descritas nas *issues*, essas eram movidas para *"Ready for Test*", que significa "Pronta para teste" ou seja, que já poderiam ser realizados os testes desta parte.

Figura 6 - *Issues* no GiLab do projeto Observatório



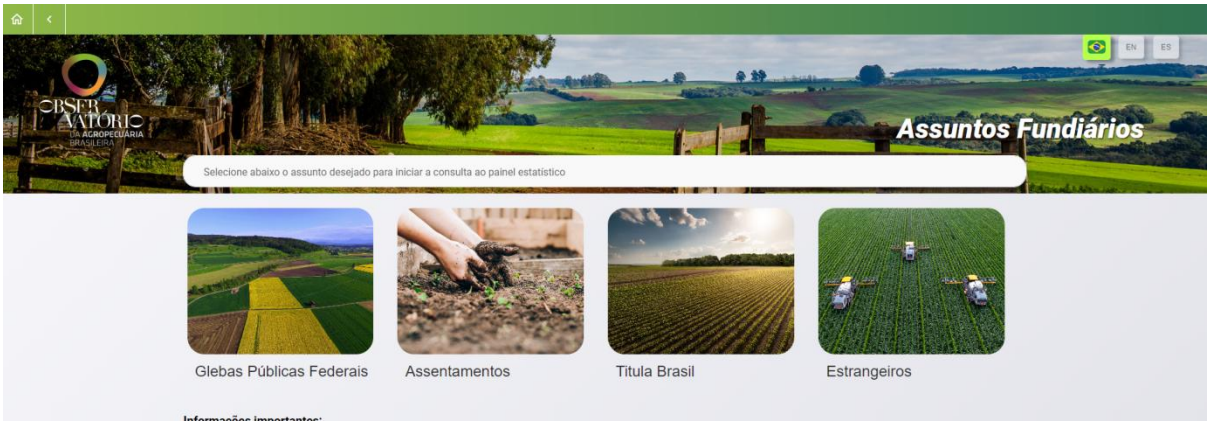
Fonte: YouxGroup (2022)

Nessa concepção, dando continuidade, a partir desse momento são feitas as validações do sistema manualmente e por detalhes, para verificar se está de acordo com o esperado pelo cliente. São verificados se a parte do sistema está de acordo com os cenários de teste da tela, se a regra de negócio definida, e se está de acordo com o protótipo já aprovado pelo cliente. Além disso, verifica-se também, se o sistema está retornando as respostas corretas e esperadas.

Para tanto, nos painéis foram testadas diversas combinações de filtros para verificar a exatidão dos dados exibidos.

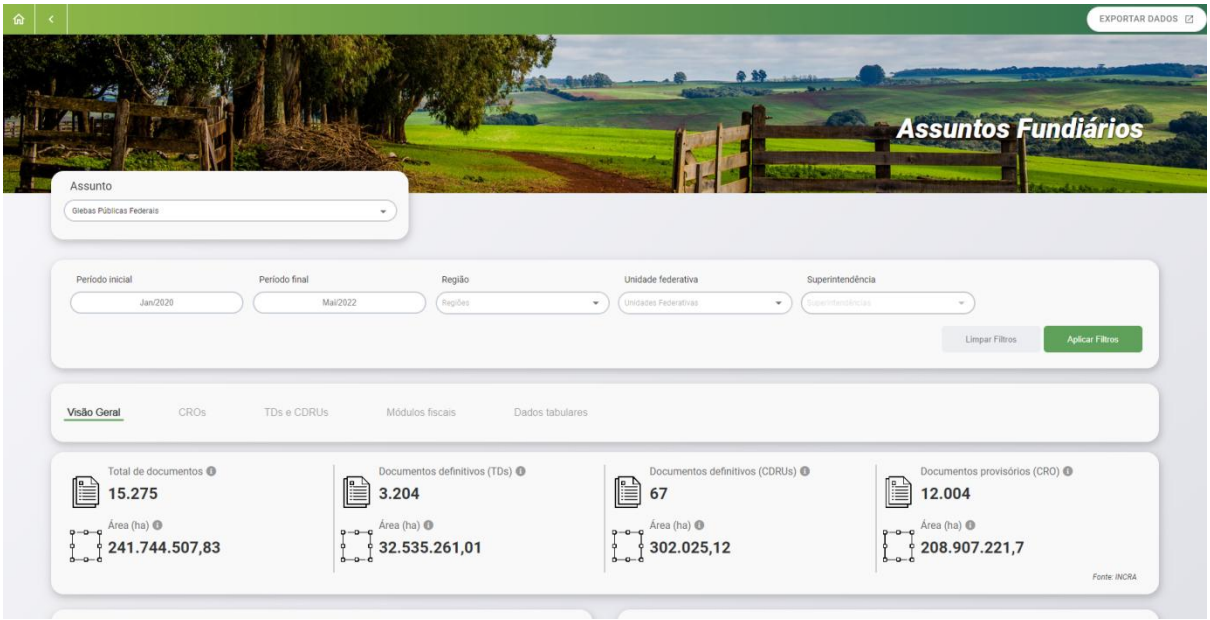
[2] No Git, significa mesclagem de alterações no sistema a uma versão já existente

Figura 7 - Exemplo de painel temático



Fonte: Observatório (2023)

Figura 8 - Filtros do Painel de Glebas Públicas Federais



Fonte: Observatório (2023)

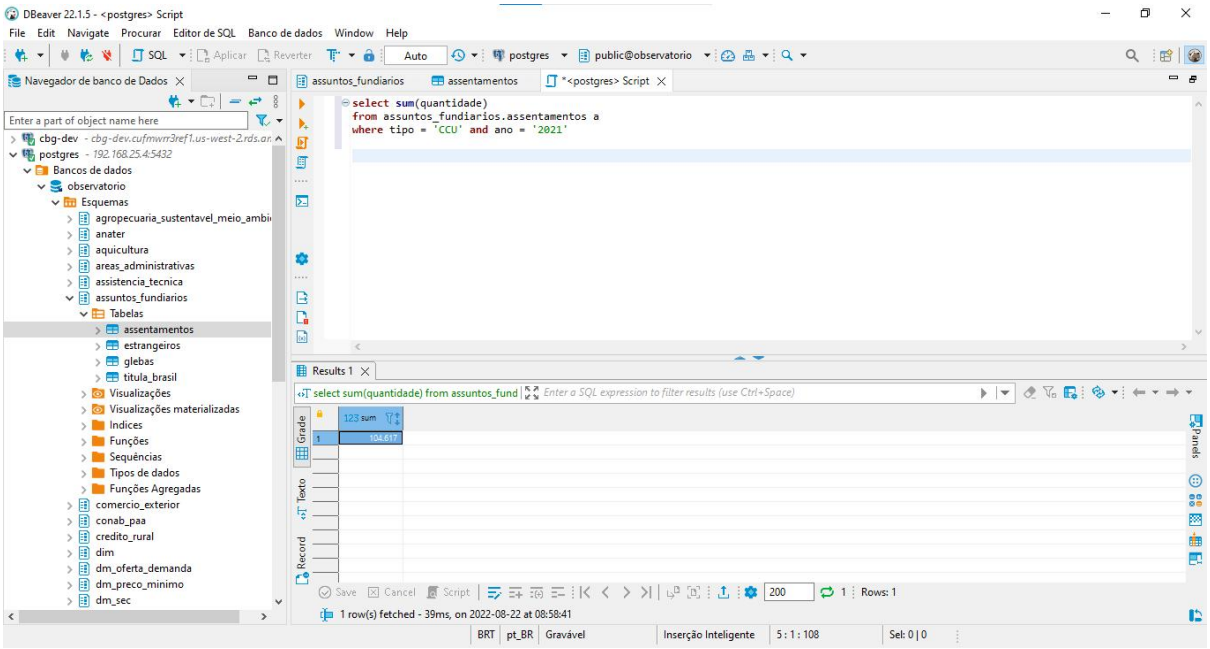
Para validação de valores, exibidos no *front-end* do sistema - página vista pelo usuário - fez se necessário a utilização de banco de dados. Um banco de dados é uma coleção organizada de informações ou dados estruturados, geralmente armazenados eletronicamente em um sistema de computador. Os dados nos tipos mais comuns de bancos de dados em operação hoje são modelados como linhas e colunas em uma série de tabelas para melhorar a eficiência do processamento e consulta de dados. Que podem ser facilmente acessados, gerenciados, modificados,

atualizados, controlados e organizados. A maioria dos bancos de dados usa *Structured Query Language* (SQL) para gravar e consultar dados (ORACLE, 2022).

Durante o Estágio, foi utilizado o banco de dados Dbeaver para realizar as consultas SQL. Kalmukov *et al.,* (2022), explana que o DBeaver é um aplicativo de software cliente SQL e uma ferramenta de gerenciamento de banco de dados.

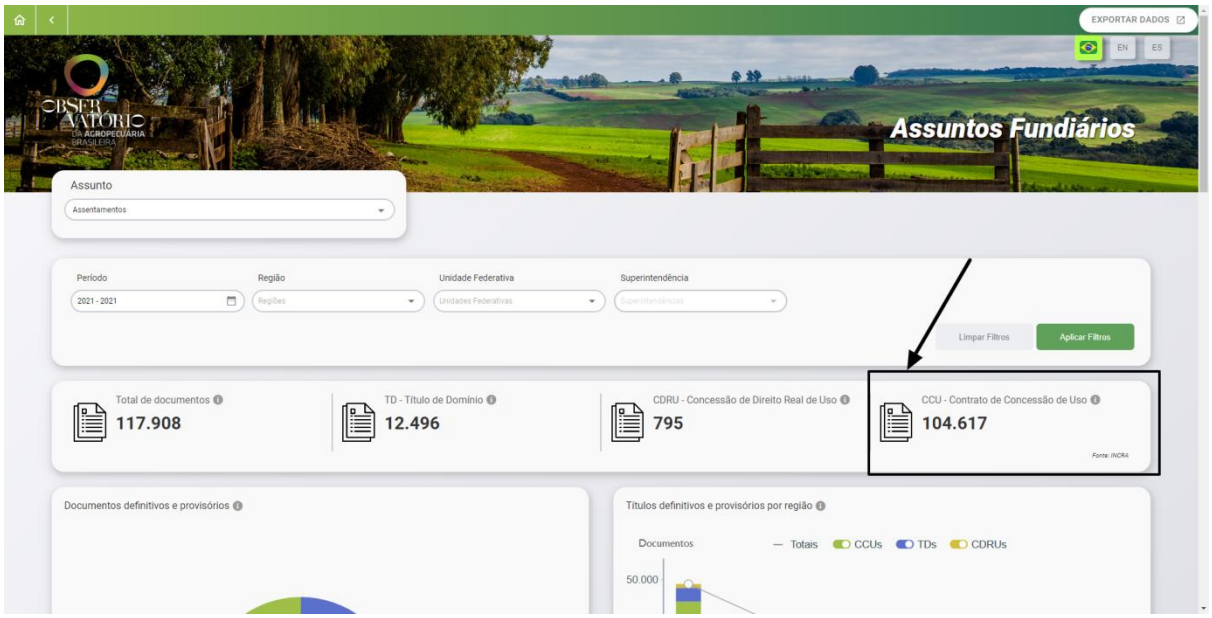
Na linguagem SQL, para consultar e verificar os valores existentes no banco, foram realizadas consultas chamadas “SELECT”. Assim, com o DBeaver, essas consultas foram realizadas para exibir os valores que estão no banco e verificar se de fato os valores retornados do sistema Observatório estão de acordo com os dados existentes no banco.

Figura 9 - Exemplo de consulta realizada no banco



Fonte: Próprio Autor (2023)

Figura 10 - Exemplo do retorno na página



Fonte: Observatório (2023)

Conforme as figuras 9 e 10, a busca é feita no banco de dados e em seguida é feita uma comparação com os valores retornados no sistema.

Desse modo, são realizadas as conferências dos valores, de acordo com um conjunto de filtros realizados no sistema, para garantir que os dados estejam vindo corretamente.

Após a execução dos testes, caso as tarefas fossem aprovadas eram movidas para "*Teste Done*", que significa que o teste foi verificado e validado. Caso não estivessem de acordo com o esperado, na *issue* eram adicionadas *Labels* e descrição do comportamento atual do sistema e retornada para *"Ready for Development*", o local no quadro de tarefas, no qual, os desenvolvedores irão verificar para realizar a(s) correção/correções necessária(s).

Depois da possível correção realizada pela pessoa desenvolvedora, o fluxo de testes é retornado. Sendo a tarefa enviada para ambiente de testes, movida a *issue* para "*Ready for Test*" e quando possível realizar o teste pelo analista de QA, a *issue* é movida para "In Test" e testada novamente. Após validação das *issues* das tarefas da Sprint pelo Analista de Qualidade, o sistema é colocado em ambiente de homologação, para também, o cliente verificar e validar se o sistema está apto para ser publicado em ambiente de produção.

5 CONSIDERAÇÕES FINAIS

A longo do período acadêmico, pode-se notar o fato de que existiram, dentre todas as excelentes disciplinas disponibilizadas no curso, aquelas que, certamente, impactaram e possibilitaram melhor execução do estágio. A disciplina de Engenharia de Software foi extremamente importante para aprender na prática, o fluxo de desenvolvimento de um sistema, com as etapas de planejamentos, desenvolvimento, teste e manutenção do software. Ademais, a disciplina eletiva de Testes de Software, que foi fundamental para a autora no aprendizado dos variados tipos de testes existentes e como realizar estes testes. Outra disciplina primordial para melhor desempenho no estágio, foi a de Qualidade de Software, na qual o tema do estágio está ligado ao conteúdo aprendido em sala de aula, como o conteúdo de normas e diretrizes para a melhoria do processo de desenvolvimento de um software com maior qualidade.

Nesse sentido, no projeto do Observatório foram realizados diversos testes nos painéis desenvolvidos durante o *sprint* para o estágio. Cada um deles foi testado de acordo com os cenários criados para cada uma das telas existentes.

Depois da mesclagem no ambiente de teste, foram executados os testes para cada nova tarefa desenvolvida e com frequente comunicação com os desenvolvedores para informar e tirar dúvidas dos comportamentos do sistema. Por fim, foi possível analisar se a parte do sistema já codificada se encontrava em conformidade com as regras de negócios definidas e em relação ao protótipo que o cliente aprovou. Além disso foi validado se o sistema retorna a resposta correta e esperada.

Portanto durante o estágio, foi possível aprender na prática como o mercado de trabalho aborda os assuntos estudados durante o curso de graduação. Além disso, ocasiounou observar como a atuação profissional trouxe além do desenvolvimento técnico, um desenvolvimento comportamental da autora, pois aprimorou aspectos de comunicação, liderança e planejamentos. O aprendizado do dia a dia de organizações com o uso da Metodologia Ágeis como SRUM, foi imensamente enriquecedor para a carreira da autora.

Sendo assim, como trabalhos futuros pode ser esperada a automação dos testes realizados na organização YOUXGROUP, visto que a automação simplifica tarefas manuais repetitivas economizando tempo e garantindo maior segurança.

REFERÊNCIAS

CHAUBEY, R.; SURESH, J.K. Integration vs. development: an engineering approach to building web applications. **IEEE Software**, feb. 2001, p. 171-181.

CONALLEM, J. **Building Web Applications with UML.** Addison Wesley, 2000. Curso W3C Escritório Brasil. CSS. Disponível em: <http://www.w3c.br/pub/Cursos/CursoCSS3/css-web.pdf>. Acesso em: 29 de Julho de 2019.

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

CASTRO, Ronney Moreira; DE CLASSE, Tadeu Moreira; SIQUEIRA, Sean Wolfgand Matsui. Técnicas e Tecnologias Diversas no Ensino Remoto Emergencial de Engenharia de Software. In: **Anais do II Simpósio Brasileiro de Educação em Computação.** SBC, 2022. p. 163-170.

DELAMARO, Marcio; JINO, Mario; MALDONADO, Jose. **Introdução ao teste de software.** Elsevier Brasil, 2013.

FERNANDES, Matheus; FONSECA, Samuel Tomkelski. **Automação de testes de software:** estudo de caso da empresa Softplan. Sistemas de Informação-Florianópolis, 2020.

FUKUSAWA, João; JUNIOR, José Orlando Balastrero. A Importância Dos Testes De Software Na Melhoria Da Qualidade Dos Sistemas Computacionais. R**ETEC-Revista de Tecnologias**, v. 8, n. 1, 2015.

HOST, King. **DBeaver:** Como acessar um banco SQL Server. 2022. Disponível em https://king.host/wiki/artigo/acessar-sql-server-pelo-dbeaver/ acesso em 15 de agosto de 2022.

KALMUKOV, Yordan; MARINOV, Milko. Hadoop as a Service: Integration of a Company's Heterogeneous Data to a Remote Hadoop Infrastructure. **International Journal of Advanced Computer Science and Applications**, v. 13, n. 4, 2022.

LATTINE, Group. **O que é Microsoft SharePoint e quais são seus benefícios?** 2021. Disponível em https://lattinegroup.com/microsoft-sharepoint/o-que-e-microsoft-sharepoint/ acesso em 15 de agosto de 2021.

LESSA, Rafael Orivaldo; LESSA JUNIOR, Edson Orivaldo. **Modelos de processos de engenharia de software.** Monografia (Graduação em Sistema de Informação), Universidade do Sil de Santa Catarina (UNISUL), Palhoça – SC, 2009. Disponível em: http://xps-project. googlecode. com/svn-history/r43/trunk/outros/02_Artigo. pdf, 2009. Acesso em: 15 ago. 2022.

OBSERVATÓRIO. Observatório da Agropecuária Brasileira. Página inicial. Disponível em: <https://observatorio.agropecuaria.inmet.gov.br/>. Acesso em: 11 de jan. de 2023

ORACLE. **O Que É um Banco de Dados?** 2022. Disponível em https://www.oracle.com/br/database/what-is-database/ acesso em 15 de agosto de 2022.

PALESTINO, Caroline Munhoz Corrêa. **Estudo de tecnologias de controle de versões de software.** 2015.

PIMENTEL, Carlos Alberto; SILVA, Danila; BEZERRA, Igor Salume. **Principais Objetivos da área de Garantia da Qualidade na Engenharia de Software e na Organização.** 2005.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software-9.** McGraw Hill Brasil, 2021. Disponível em https://books.google.com.br/books?hl=pt-BR&lr=&id=FSE3EAAAQBAJ&oi=fnd&pg=PT33&dq=PRESSMAN,+Roger+S.%3B+MAXIM,+Bruce+R.+Engenharia+de+software-9+.+McGraw+Hill+Brasil,+2021.&ots=kzMLgSxHvH&sig=9j2wV5Y3k_LhB18yxwAgMpzD2Eo#v=onepage&q=PRESSMAN%2C%20Roger%20S.%3B%20MAXIM%2C%20Bruce%20R.%20Engenharia%20de%20software-9%20.%20McGraw%20Hill%20Brasil%2C%202021.&f=false acesso em 31 de agosto de 2022.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação.** Brasport, 2006.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de software.** Alta Books Editora, 2006.

RIZZUTTO, Gabriel Almeida. **Criação de uma aplicação para automação e otimização de processos internos de uma empresa em linguagem JAVA.** 2022.

ROSA, Luis Henrique Carvalho et al. Ensino de testes de software por meio de digital Storytelling e Chatterbots. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE).** 2017. p. 797.

SANTANA, L. Implantação de Testes Não Funcionais em uma Fábrica de Software. **Revista Ada Lovelace**, [S. l.], v. 4, 2022. Disponível em: http://anais.unievangelica.edu.br/index.php/adalovelace/article/view/8292. Acesso em: 6 fev. 2023.

SANTOS, Júlia de Avila et al. Pensamento Computacional e Engenharia de Software: primeiros passos em direção a uma proposta de sistematização de resolução de problemas. In: **Anais do XXX Workshop sobre Educação em Computação.** SBC, 2022. p. 451-462.

SCRUMGUIDES. **Scrum Guides,** c2020. Scrum Guides. Disponível em: https://scrumguides.org/index.html. Acesso em: 16 agosto de 2022.

SCHWABER, Ken; SUTHERLAND Jeff. **O Guia do Scrum.** Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-2.0.pdf. Acesso em: 16 de agosto de 2022.

SOMMERVILLE, Ian. **Engineering software products**. London: Pearson, 2020

SOUZA, Rafael Santana de. **Testes de software para garantir a qualidade.** 2016.

TRIGO, Cláudia Filipa Ferreira. **Impacto na Realização de Testes de Software.** 2017. Tese de Doutorado. Instituto Politecnico de Viseu (Portugal).

VICTORINO, Marcio; BRÄSCHER, Marisa. Organização da informação e do conhecimento, engenharia de software e arquitetura orientada a serviços: uma abordagem holística para o desenvolvimento de sistemas de informação computadorizados. **Revista de Ciência da Informação**, v. 10, n. 3, 2009.

VALENTE, Marco Tulio. **Engenharia de Software Moderna.** Disponível em: https://engsoftmoderna.info/. Acesso em: 16 agosto de 2022.