



QUALIDADE DE SOFTWARE

Nome completo

Gabriela Mattesco Gomes

Análise de Qualidade

Cidade

Rio de Janeiro

Ano

2022

1. RESUMO

O presente trabalho tem como objetivo apresentar a importância da qualidade nos processos e no software desenvolvido.

Para garantir esta qualidade é fundamental possuir uma equipe de testes. Esta equipe será responsável por validar todos os processos e passar por todas as etapas do software, verificando se todos os requisitos foram atendidos sem apresentar nenhum erro.

A consequência final deste investimento é obter produtos melhores e com maior confiabilidade.

Os softwares precisam atender as especificações dos clientes, sem ultrapassar prazos e custos previamente definidos.

Este trabalho trata dos conceitos relacionados à Qualidade de Software e Garantia da Qualidade de Software. E a importância de todas as fases do Gerenciamento da Qualidade.

Pois, ter um software criado com qualidade desde o seu início, garante a satisfação dos clientes.

2. SUMÁRIO

1. RESUMO	2
2. SUMÁRIO	3
3. INTRODUÇÃO	4
4. O PROJETO	6
4.1 Estratégia de teste	10
4.2 Critérios de aceitação	10
4.2.1 História de usuário 1: [US-0001] – Adicionar item ao carrinho	11
4.2.2 História de usuário 2: [US-0002] – Login na plataforma	11
4.2.3 História de usuário 3: [US-0003] – API de cupons	12
4.3 Casos de testes	12
4.3.1 História de usuário 1: Adicionar item ao carrinho	13
4.3.2 História de usuário 2: Login na plataforma	13
4.3.3 História de usuário 3: API de Cupom	14
4.4 Repositório no Github	15
4.5 Testes automatizados	15
4.6 Integração contínua	16
4.7 Testes de performance	16
5. CONCLUSÃO	18
6. REFERÊNCIAS BIBLIOGRÁFICAS	19

3. INTRODUÇÃO

O presente trabalho abordará sobre a importância da qualidade em testes de Software.

É possível afirmar que atualmente as organizações estão aumentando sua dependência tecnológica. Portanto, atingir um alto nível de qualidade de produto ou serviço passou a ser o principal objetivo das empresas. Nos dias de hoje, não é mais aceitável entregar produtos com baixa qualidade, pelo contrário, é necessário causar um impacto positivo.

Desta forma, passou a ser fundamental ter softwares produzidos com qualidade tendo como resultado final a satisfação dos clientes, a redução de custos com correções e retrabalho e maior confiança no produto produzido.

Qualidade de Software é a área de conhecimento da Engenharia de Software que objetiva garantir a qualidade do software através da definição e normatização de processos e desenvolvimento.

Exemplo de definição que pode ser atribuída a Qualidade:

“Qualidade de Software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos”. (BARTIÉ, 2002 p.16)

- Estar em conformidade com os requisitos dos clientes;
- Antecipar e satisfazer as necessidades dos clientes;
- Documentar tudo o que deve ser feito, e fazer tudo o que foi documentado.

Já a garantia de qualidade é o processo responsável por certificar que as atividades e os objetivos determinados serão atingidos, garantindo a correta execução do que foi determinado. Sendo necessário garantir a qualidade em todas as partes do processo, não iniciando uma nova etapa caso ainda haja erros na etapa atual.

“Qualidade não é a fase do ciclo de desenvolvimento de Software, é parte de todas as áreas” (BARTIÉ, 2002)

Após a execução de um software, se o mesmo apresentar problemas ou erros, isto ocasionará um custo maior do que o previsto, e os prejuízos serão tanto para o cliente quanto para a empresa.

O prejuízo para o cliente é de ter investido em um projeto achando que

seria entregue na data determinada em contrato.

Já o prejuízo para a empresa é de ter gastos com retrabalho, sendo necessário alocar uma equipe, pois após a data acordada a responsabilidade dos custos passam a ser da empresa.

Para garantir a qualidade de um software, empresas de Tecnologia passaram a investir em equipes de teste, demonstrando que o capital investido em pessoas qualificadas além de reduzir o custo investido, ainda aumenta a satisfação do cliente no resultado final do produto.

Portanto, podemos afirmar que todo o gasto realizado com a qualidade deve ser considerado investimento.

4. O PROJETO

Este projeto irá abordar os processos necessários que um QA precisa seguir para que testes sejam entregues com sucesso e como resultado apresentar/entregar um software de qualidade.

Primeiramente a empresa precisa identificar qual metodologia será usada, pois através dela, a equipe conseguirá administrar as atividades. Temos duas metodologias ágeis mais usadas, a Kanban e o Scrum.

Na Kanban, as atividades possuem um fluxo contínuo. Nada mais é que um método de gestão à vista. Onde em um quadro é apresentado o número de tarefas que um time possui atualmente, onde uma equipe consegue fazer o acompanhamento destas tarefas, mostra a capacidade de fluxo de trabalho de um time. Esta apresentação pode ser feita através de colunas, cartões... Sendo possível fazer mudanças a qualquer momento.

Tem como vantagem acelerar e encurtar o tempo das atividades, facilitar a redução de desperdícios e custos, gerar o aumento da comunicação da equipe, gerar o aumento da produtividade, eliminar atividades que não agregam valor, além de ser um processo simplificado.

Já o Scrum, as atividades possuem uma duração fixa. Esta metodologia se resume em trabalho em equipe, desde que exista um grupo trabalhando juntos para um objetivo. É uma estrutura que ajuda as equipes a trabalharem juntas. Tem como valores: coragem, foco, comprometimento e respeito. Porém nesta metodologia as equipes não devem fazer mudanças durante a Sprint.

Seus eventos são o planejamento da Sprint, reunião diária e revisão da Sprint.

Sprint tem como objetivo otimizar entregas, ela corresponde aos períodos, nos quais as atividades pré-definidas precisam ser cumpridas. É cada uma das fases de um projeto, definidas por tempo, que normalmente tem duração de uma a quatro semanas.

Na revisão da Sprint o objetivo principal é o time apresentar o trabalho que foi desenvolvido durante as fases do projeto, a Sprint. Precisam apresentar que todos os critérios de aceite de uma história foram concluídos com sucesso.

Já a Sprint retrospectiva ocorre ao final de uma Sprint. Esta reunião serve para identificar o que funcionou bem, o que pode ser melhorado e que

ações serão tomadas para melhorar. A equipe discute sobre os pontos positivos e negativos da Sprint. Pensando sempre em um plano para melhor atingir seu objetivo.

Muitas empresas estão adotando esta infraestrutura ágil pois ela ajuda a otimizar processos, faz com que a equipe tenha mais eficiência no trabalho, ajuda no planejamento, identifica o nível de colaboração necessária que deve ser aplicada além da construção do pacote de entrega, ou seja, produto dividido em Sprints.

Em resumo, o método ágil nada mais é que:

- Indivíduos e interações são mais que processos e ferramentas.
- Softwares em funcionamento são mais que documentação.
- Colaboração com o cliente mais que negociação de contrato.
- Responder a mudanças mais que seguir o plano.

É uma forma de conduzir projetos com maior rapidez aos processos e à conclusão de tarefas. Não significa entregar mais rápido, mas sim entregar valor mais rápido ao cliente.

É fundamental saber distinguir Quality Analyst e Quality Assurance.

O papel do Quality Analyst (QA) vai muito além de só testar, ele gera impacto positivo para o projeto. Ele tem como responsabilidades: Planejar os testes e elaborar um roteiro de testes, otimizar processos, minimizar erros, prevenir bugs além de propor melhorias, desenvolver scripts de testes automatizados, validar a configuração, adiantar os testes e participar ativamente de reuniões junto ao time. Garantindo assim que o software a ser implantado atenda todas as especificações exigidas pelo cliente. Em um resumo final, o QA precisa pensar o que ele fará hoje para ajudar a equipe a atingir a meta da Sprint.

Quality Assurance é um conjunto de atividades que garantem que o produto/serviço esteja de acordo com o padrão de qualidade exigido.

Em um projeto, é fundamental passar por todas as fases do processo para efetuar um teste de qualidade.

Efetuar um Plano de Teste, nele é documentado todo o planejamento para a realização dos testes, apontando os itens que serão testados, quais atividades serão executadas e os riscos dos testes. Identifica também quais serão os ambientes a serem testados. O Plano de Teste é o documento que

dará origem aos demais documentos.

Criar uma história de usuário, isto significa, uma descrição curta, informal e em linguagem simples do que um usuário quer fazer dentro de um produto de software para obter algo que ele considere valioso.

As histórias de usuários normalmente seguem um padrão:

- Como [tipo de usuário]
- Quero [uma ação]
- Para [resultado/valor]

O benefício da história de usuário, é dar à equipe de desenvolvimento o contexto e o porquê do que eles estão desenvolvendo. Fazer isso os ajuda a entender como eles estão fornecendo valor para o negócio e para manter o usuário/cliente no foco das atenções.

Os casos de teste - A melhor definição para casos de teste é o maior detalhamento possível de um teste. São identificadas as informações que serão utilizadas durante o teste e quais serão os resultados retornados.

Caso durante o teste ocorra um resultado negativo/inesperado, isso nada mais é do que um erro ou falha do sistema, fazendo com que ele produza um resultado incorreto. Esta falha é conhecida como bug.

Um bug pode custar caro para um projeto, pois perde tempo e dinheiro.

Portanto, é necessário que ele seja reportado durante todo o ciclo de desenvolvimento e quanto antes melhor, pois diminui o risco de um problema na produção.

É primordial que o QA siga as boas práticas na hora de reportar um bug, ele precisa ser objetivo, precisa tornar evidente as situações de falha, deixar claro o resultado esperado, apresentar um título claro e explicativo, fornecer informações sobre o evento inesperado, facilitando desta forma o trabalho dos desenvolvedores da equipe.

Exemplo de como reportar um bug:

- ID/Título
- Descrição
- Ambiente
- Evidências (prova visual)
- Resultado e comportamento esperado

- Gravidade (crítica, principal, trivial)
- Prioridade (alta, média, baixa)
- Criado por:
- Atribuído para:

Exemplo de definição para defeitos e erros:

“Defeitos de software é quando uma aplicação não se comporta da forma esperada, ou não retorna os resultados esperados. Estes são os principais motivos que geram retrabalho, custos, não cumprimento de prazos, redução da produtividade e insatisfação do cliente.” (BARTIÉ, 2002)

“Os erros estão em todas as etapas do processo, porém, estudos reforçam que a maior parte está presente nas fases iniciais do projeto. Com isto, se a qualidade fosse aplicada desde a definição do projeto, muitos erros seriam identificados prematuramente evitando que fossem migrados para as outras fases.” (BARTIÉ, 2002)

É necessário que as empresas adotem o shift left testing, ou seja, fazer os testes desde o início, na fase de requisitos, no desenvolvimento. Os testes executados no final do desenvolvimento são os realizados pelo cliente, para apenas validar que o produto atende as suas necessidades.

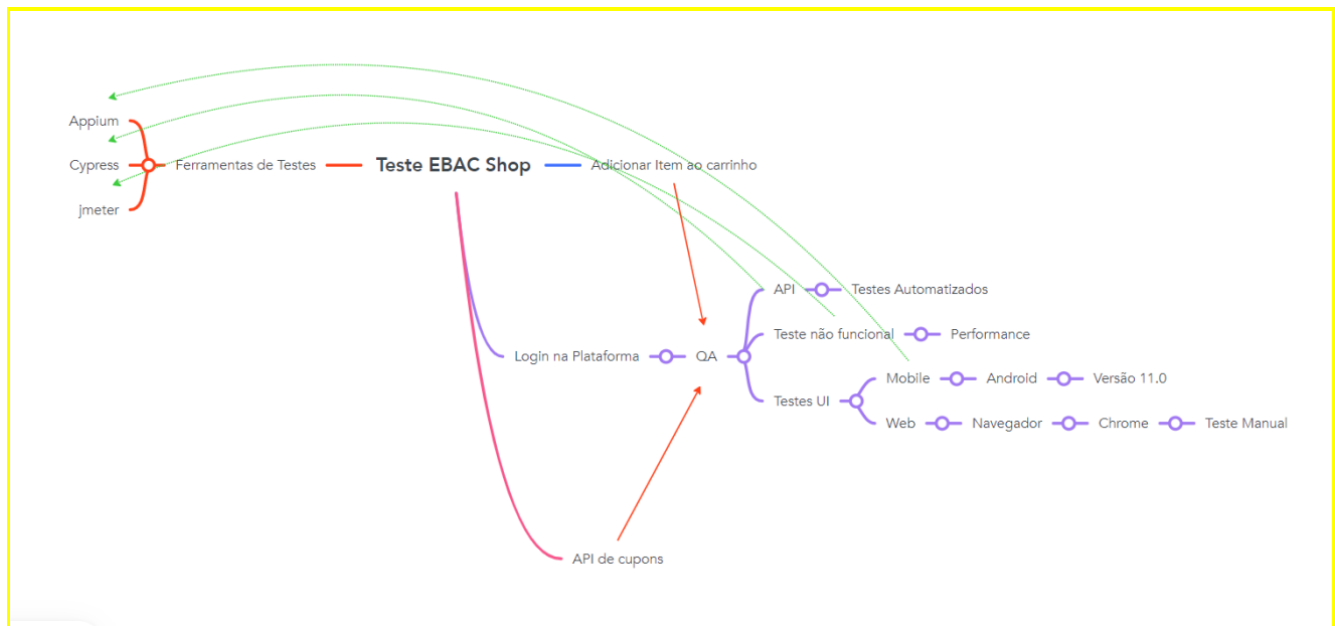
Existem diferentes tipos de testes, eles servem para avaliar a funcionalidade, usabilidade, performance, confiabilidade e estrutura. E diferentes formas de realizar: testes manuais e testes automatizados, este tem o objetivo de simular a ação de um usuário de forma automática.

Neste projeto será realizado um teste automático para verificar a funcionalidade do software, um teste de performance para verificar o desempenho, a eficiência, o tempo de resposta do site e um teste de API, este nos ajuda a garantir o funcionamento e a confiabilidade do aplicativo e sistema baseado em dados.

4.1 Estratégia de teste

- Faça uma estratégia de testes em um mapa mental, seguindo algumas diretrizes como objetivos, papéis e responsabilidades, fases de testes, padrões, tipos de testes, técnicas de testes, ambientes, ferramentas, abordagem (manual ou automatizado), framework ou ferramenta usados, plataformas (web, api, mobile), etc.;
- Referência: Módulo 5
- Após fazer sua estratégia de teste, tire um print e cole aqui:

Mapa mental – Estratégia de teste



4.2 Critérios de aceitação

- Considere as histórias de usuário: [US-0001] – Adicionar item ao carrinho, [US-0002] – Login na plataforma e [US-0003] – API de cupons
- Para cada uma delas crie pelo menos 2 critérios de aceitação usando a linguagem Gherkin;
- Em pelo menos um dos critérios, usar tabela de exemplos (Esquema do Cenário / Scenario Outline);
- Referência: Módulo 8

4.2.1 História de usuário 1: [US-0001] – Adicionar item ao carrinho

Critério de aceitação: Não deve permitir produtos sem estoque

Cenário 1: Compra de produto sem estoque

Dado que eu acesse a página de produtos da EBAC-SHOP

Quando eu selecionar o produto “camisa Brasil”

E o tamanho M

Então deve exibir uma mensagem de “produto sem estoque”

Critério de aceitação: Deve Informar sucesso da compra, após aprovação do pagamento

Cenário 2: Compra realizada com sucesso

Dado que eu acesse a página meu carrinho da EBAC-SHOP

Quando eu inserir os dados para pagamento

E clicar em concluir

Então deve exibir uma mensagem de “Pagamento aprovado - Compra efetuada com sucesso”

4.2.2 História de usuário 2: [US-0002] – Login na plataforma

Critérios de aceitação: Deve permitir login através de e-mail ou cpf

Cenário 1: Login com campo e-mail válido

Dado que eu acesse a página de login da EBAC-SHOP

Quando eu digitar um usuário válido “gabriela@ebac.com.br”

E a senha “gabi2022”

Então deve exibir uma mensagem de boas vindas “Olá Gabriela”

Esquema do Cenário 2: Autenticar múltiplos usuários

Dado que eu acesse a página de login da EBAC-SHOP

Quando eu digitar o <usuario>

E a <senha>

Então deve exibir a <mensagem> de sucesso

Exemplos:

usuario	senha	mensagem
"gabriela@ebac.com.br"	"gabiebac2022"	"Olá Gabriela!"
"maria@ebac.com.br"	"mariaebac2022"	"Olá Maria!"
"gabriel@ebac.com.br"	"bielebac2022"	"Olá Gabriel!"

4.2.3 História de usuário 3: [US-0003] – API de cupons

Critério de aceitação: Deve apenas mostrar cupons de 10% da loja selecionada

Cenário 1: Cupom BEMVNDO10 loja EBAC

Dado que eu acesse a API de cupons

Quando eu efetuar uma pesquisa sobre a loja da EBAC

E procurar por cupons de 10%

Então deve exibir uma mensagem de "Cupom BEMVINDO10 para primeira compra no site"

Critério de aceitação: Deve expor somente cupons válidos

Cenário 2: Cupons válidos loja EBAC-SHOP

Dado que eu acesse a API de cupons

Quando eu selecionar a loja EBAC-SHOP

E clicar em visualizar cupons

Então deve exibir uma mensagem "Cupons válidos para a data de hoje da loja EBAC-SHOP"

4.3 Casos de testes

- Crie pelo menos 3 casos de testes para cada história de usuário, sempre que possível, usando as técnicas de testes (partição de equivalência, valor limite, tabela de decisão etc.).
- Considere sempre o caminho feliz (fluxo principal) e o caminho alternativo e negativo (fluxo alternativo). Exemplo de cenário negativo: "Ao preencher com usuário e senha inválidos deve exibir uma mensagem de alerta..."
- Referência: Módulo 4 e 5

4.3.1 História de usuário 1: Adicionar item ao carrinho

CT01: Os valores dos itens devem estar entre R\$19,00 e R\$99,00

CT02: Permitir compra dos itens apenas das 9 às 18 horas

CT03: Permitir adicionar no máximo de 30 itens por carrinho

CT01 - Inserir item ao carrinho no valor de R\$19,01 (Válida)

CT02 - Permitir compra às 8:59 horas (Inválida)

CT03 - Permitir adicionar 29 itens (Válida)

Condições	Regra 1	Regra 2	Regra 3	Regra 4
Adicionar no máximo de 30 itens?	sim	sim	não	não
Compra das 9 às 18 horas?	sim	não	sim	não
Ações				
Deve ser adicionado?	sim	sim	não	não
Permitir compra?	sim	não	sim	não

4.3.2 História de usuário 2: Login na plataforma

CT01: Login efetuado das 6 às 18 horas devem ser executados com sucesso

CT02: Login realizado há mais de 30 minutos devem ser deslogado

CT03: Permitir login na plataforma de no máximo de 30 usuários por vez

CT01 - Login efetuado às 18:01 (Inválida)

CT02 - Validar login com 27 minutos (Válida)

CT03 - Permitir login de 29 usuários (Válida)

Condições	Regra 1	Regra 2	Regra 3	Regra 4
Login efetuado às 9 horas?	sim	sim	não	não
Máximo de 30 usuários?	sim	não	sim	não
Ações				
Efetuada com sucesso?	sim	sim	não	não
Permitir cadastrado?	sim	não	sim	não

4.3.3 História de usuário 3: API de Cupom

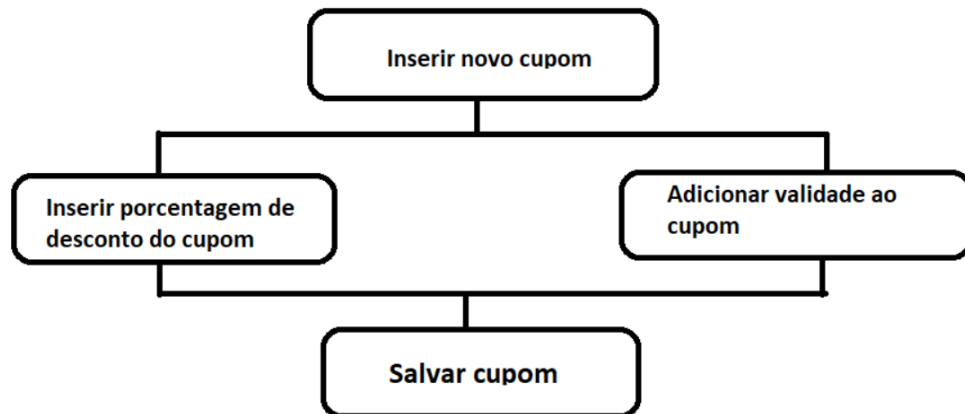
Caminho feliz:

CT01: Inserir novo cupom

CT02: Inserir porcentagem de desconto do cupom

CT03: Adicionar validade ao cupom

CT04: Salvar cupom



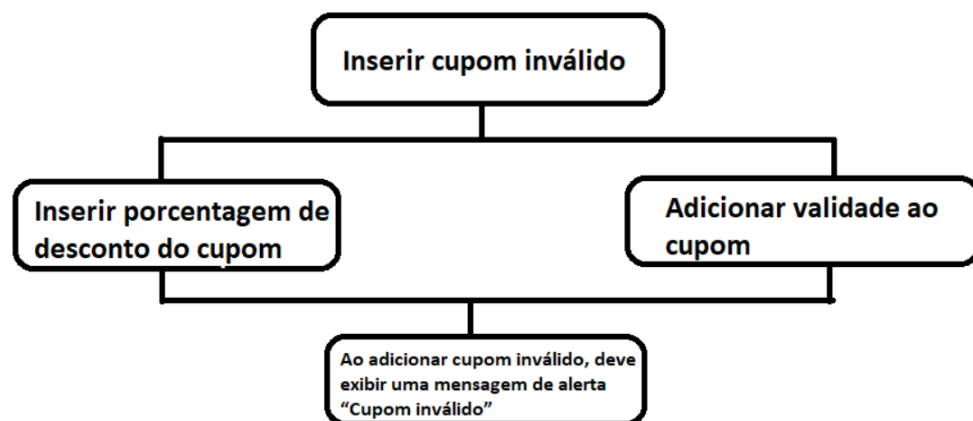
Caminho negativo:

CT01: Inserir cupom inválido

CT02: Inserir porcentagem de desconto do cupom

CT03: Adicionar validade do cupom

CT04 : Ao adicionar cupom inválido, deve exibir uma mensagem de alerta "Cupom inválido"



4.4 Repositório no Github

- Crie um repositório no github com o nome TCC-EBAC;
- Deixe o repositório público até a análise dos tutores;
- Neste repositório você deve subir este arquivo e todos os código fontes da automação WEB, API, Mobile, Performance e CI.
- Referência: Módulo 10
- Link do repositório: <https://github.com/GabiMattesco/TCC-EBAC.git>

4.5 Testes automatizados

4.5.1 - Automação de UI

- Crie um projeto de automação no Cypress;
- Crie uma pasta chamada UI para os testes WEB da História de Usuário [US-0001] – Adicionar item ao carrinho;
- Na automação deve adicionar pelo menos 3 produtos diferentes e validar se os itens foram adicionados com sucesso.

4.5.2 - Automação de API

- Crie uma pasta chamada API para os testes de API da História de usuário “**Api de cupons**”.
- Faça a automação de **listar** os cupons e **cadastrar** cupom, seguindo as regras da História de usuário.
- Exemplo da automação de Api – GET

```
it('Deve listar todos os cupons cadastrados', () => {  
  cy.request({  
    method: 'GET',  
    url: 'coupons',  
    headers: {  
      authorization: 'código_da_autorização_aqui'  
    }  
  }).should((response) => {  
    cy.log(response)  
    expect(response.status).to.equal(200)  
  })  
});
```

- Obs.: Considere todas as boas práticas de otimização de cenários (Page Objects, Massa de dados, Custom Commands, elementos etc.).
- Referência: Módulo 11, 12 e 14

4.6 Integração contínua

- Coloque os testes automatizados na integração contínua com jenkins, criando um job para execução da sua automação;
- Compartilhe o *jenkinsfile* no repositório, junto ao seu projeto.
- Referência: Módulo 15

4.7 Testes de performance

- Usando o Apache Jmeter, faça um teste de performance com o fluxo de login da História de usuário: [US-0002] – Login na plataforma
- Crie um template de gravação no jmeter (recording);
- Use massa de dados dinâmica em arquivo CSV;
- Referência: Módulo 18
- Configurações do teste de performance:

-Usuários virtuais: 20

-Tempo de execução: 2 minutos

-RampUp: 20 segundos

-Massa de dados: Usuário / senha:






user1_ebac / psw!ebac@test

user2_ebac / psw!ebac@test

user3_ebac / psw!ebac@test

user4_ebac / psw!ebac@test

user5_ebac / psw!ebac@test

<input type="checkbox"/> Nome de usuário	Nome	E-mail	Função
<input type="checkbox"/>  user1_ebac	—	user1_ebac@ebac.com	Assinante
<input type="checkbox"/>  user2_ebac	—	user2_ebac@ebac.com	Assinante
<input type="checkbox"/>  user3_ebac	—	user3_ebac@ebac.com	Assinante
<input type="checkbox"/>  user4_ebac	—	user4_ebac@ebac.com	Assinante
<input type="checkbox"/>  user5_ebac	—	user5_ebac@ebac.com	Assinante

- DICA: Em uma das requisições, após a gravação, vai aparecer os parâmetros usados. Substitua esses parâmetros pela sua massa de dados, conforme aprendido em aula:

HTTP Request

Name: -31

Comments: Detected the start of a redirect chain

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP:

HTTP Request

POST Path: /minha-conta/

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data

☐ Browser-compatible headers

Parameters

Body Data

Files Upload

Send Parameters With the Request:

Name:	Value	UF
username	\$(usuario)	
password	\$(senha)	
woocommerce-login-nonce	00000000000000000000000000000000	
_wp_http_referer	/minha-conta/	
login	Login	

5. CONCLUSÃO

Este trabalho possibilitou entender o quanto é importante implantar um software de qualidade nos dias de hoje.

Pois pôde-se perceber que o cliente e o usuário estão cada vez mais exigentes, buscando projetos que atendam tanto os custos, como prazos e qualidade.

A qualidade precisa ser vista pelas empresas como um investimento, uma forma de produzir softwares melhores. Uma vez que os gastos realizados com softwares “problemáticos” são muito maiores.

Todos os esforços realizados para se obter um software com qualidade são muito mais rentáveis do que gerar trabalhos, manutenções, perder a confiabilidade dos clientes e com grande possibilidade de ao passar do tempo não serem mais úteis para o objetivo determinado.

As metodologias ágeis vieram suprir estas exigências garantindo uma formalização e uma produção de alto desempenho, sem deixar de lado o atendimento aos requisitos de segurança e controle de todo o processo de desenvolvimento de software.

A qualidade contribui para a satisfação do cliente. Pois é melhor encontrar bugs no processo de desenvolvimento do que nas mãos do cliente.

Sendo possível afirmar que o teste inicial economiza tempo e dinheiro. Com isto a necessidade de possuir um profissional qualificado na equipe para a realização dos testes. Pois, o Quality Analyst determina os meios para garantir a qualidade do software a ser entregue, monitorando os processos e métodos. Evitando assim custos desnecessários.

Ao final deste trabalho conclui-se que os objetivos propostos foram plenamente alcançados e que os resultados foram gerados com sucesso.

Em resumo, foi entender que um software pronto não é apenas um software entregue e sim um software com garantia de qualidade.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Meios Impressos

BARTIE, Alexandre. Garantia da Qualidade de Software. Rio de Janeiro: Elsevier. 2002.

PRESSMAN, Roger. Engenharia de Software, São Paulo: Makron Books, 2002.

Meios Digitais

Disponível em:

<http://lojaebac.ebaonline.art.br>

Acessado em 01 Junho de 2022 às 19h10min

<https://pt.slideshare.net/DnySchmidt/tcc-qualidade-e-testes-de-sofwar-69829493>

Acessado em 17 de Junho de 2022 às 18h59min

<https://blog.mandic.com.br/artigos/10-dicas-para-testes-de-api-bem-sucedidos/>

Acessado em 20 de Junho de 2022 às 20h03min

<https://www.digite.com/pt-br/agile/historias-de-usuarios/>

Acessado em 20 de Junho de 2022 às 20h15min