CENTRO UNIVERSITÁRIO UNIDADE DE ENSINO SUPERIOR DOM BOSCO
CURSO SISTEMAS DE INFORMAÇÃO

WILLIAM RICHARD EVERTON PINHEIRO

TESTES DE SOFTWARE: estudo exploratório sobre testes de segurança, usabilidade, performance e acessibilidade

São Luís
2022

WILLIAM RICHARD EVERTON PINHEIRO

TESTES DE SOFTWARE: estudo exploratório sobre testes de segurança, usabilidade, performance e acessibilidade

Monografia apresentada ao Curso de Sistemas de Informação do Centro Universitário Unidade de Ensino Superior Dom Bosco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.
Orientador: Prof. Dr. Giovanni Lucca França da Silva.

São Luís

2022

Dados Internacionais de Catalogação na Publicação (CIP)

Centro Universitário – UNDB / Biblioteca

Pinheiro, William Richard Everton

Testes de software: estudo exploratório dos tipos de teste de software. / William Richard Everton Pinheiro. __ São Luís, 2022.

44 f.

Orientador: Prof. Dr. Giovanni Lucca França da Silva.

Monografia (Graduação em Sistemas de Informação) - Curso de Sistemas de Informação - Centro Universitário Unidade de Ensino Superior Dom Bosco - UNDB, 2022.

1. Qualidade de software. 2. Usabilidade. 3. Testes de software. I. Título.

CDU 004.05

WILLIAM RICHARD EVERTON PINHEIRO

TESTES DE SOFTWARE: estudo exploratório sobre testes de segurança, usabilidade, performance e acessibilidade

Monografia apresentada ao Curso de Sistemas de Informação do Centro Universitário Unidade de Ensino Superior Dom Bosco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovada em: _____/____/_____.

BANCA EXAMINADORA:

_____

Prof. Dr. Giovanni Lucca França da Silva
Doutor em Engenharia Elétrica
Centro Universitário Unidade de Ensino Superior Dom Bosco (UNDB)

_____

Prof. Me. Rodrigo Monteiro de Lima
Mestre em Ciência da Computação
Centro Universitário Unidade de Ensino Superior Dom Bosco (UNDB)

_____

Prof. Me. Igor Luciano Cavalcanti Lima
Mestre em Ciência da Computação
Centro Universitário Unidade de Ensino Superior Dom Bosco (UNDB)

RESUMO

Os sistemas de informação começaram a ser desenvolvidos na década de 70, onde passaram por uma crise devido a baixa qualidade com que eram entregues, prazos não respeitados, alto custo final e requisitos que não eram atendidos. Devido a isso, foi necessária a criação de uma metodologia de testes, que atestasse o bom funcionamento da ferramenta, testes esses que podem ser divididos em vários tipos, como usabilidade, performance, segurança, estabilidade, entre outros. Tem-se então como objetivo geral do trabalho o estudo exploratório dos principais tipos de teste de software, qual seu uso e formas de medição. Apresenta-se então uma pesquisa teórica acerca de vários tipos de testes de software, e depois ferramentas e como eles podem ser realizados, de maneira a garantir a qualidade da entrega.

Palavras-chave: Qualidade de software. Usabilidade. Testes de software. Estudo exploratório.

ABSTRACT

Information systems began to be developed in the 70s, where they went through a crisis due to the low quality with which they were delivered, deadlines not respected, high final cost and requirements that were not met. Because of this, it was necessary to create a testing methodology, which would attest to the proper functioning of the tool, tests that can be divided into several types, such as usability, performance, security, stability, among others. Therefore, the general objective of the work is the exploratory study of the main types of software testing, its use and forms of measurement. A theoretical research is then presented about various types of software tests, and then tools and how they can be performed, in order to guarantee the quality of the delivery.

Keywords: Software quality. Usability. Software tests. Exploratory study.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicações
PMBOK	*Project Management Body of Knowledge*
TCC	Trabalho de Conclusão de Curso

SUMÁRIO

1 INTRODUÇÃO

No início dos anos 70 vivia-se uma crise no desenvolvimento de softwares, conforme o Portal Ciência da Computação (2015). As estimativas de prazo não eram corretas, a demanda era maior que a produtividade, boa parte dos custos ou prazos eram ultrapassados, pelo menos 1/3 dos projetos eram cancelados, não existia uma preocupação com manutenções futuras e boa parte dos sistemas não atendiam aos requisitos dos usuários.

Isso pode ser observado por exemplo com o lançamento do foguete Ariane 5, que explodiu após 36 segundos de lançamento devido a uma falha conhecida como “estouro de inteiros”, ou seja, quando uma variável tenta armazenar um valor maior do que o definido e disponível em um determinado espaço de memória (ARIANE-5G, 2014).

A solução encontrada foi mudar o paradigma de como os sistemas estavam sendo desenvolvidos, tendo-se uma maior preocupação em como deveria ser feito, treinar os usuários, utilização de ferramentas, processos e testes, de acordo com Pressman (2005).

Dessa maneira, conforme Cunha (2010), é a Engenharia de Software, dentro da fase de testes, que fornece informações sobre a qualidade de um sistema com relação com contexto em que se espera que ele opere, e que planeja como será feito o desenvolvimento, para que todos os requisitos desejados pelo usuário sejam atendidos.

1.1 Justificativa

Os testes de software asseguram um equilíbrio na entrega no software, garantindo não somente o bom funcionamento da ferramenta, mas também o atendimento a princípios básicos de segurança e aos requisitos contratados, conforme Rocha (2011). Ainda podem ser feitos testes de usabilidade, performance, estabilidade, entre outros, de forma a atestar quais os limites de desempenho do sistema.

Dessa forma, o conhecimento dos tipos de testes, como são feitos e sua abordagem permite que o time de testadores ganhe tempo e, conforme complexidade e risco identificado no sistema, faça os testes mais apropriados, de acordo com Junckes e Morgado (2013).

1.2 Objetivos

Tem-se então como objetivo geral do trabalho o estudo exploratório dos tipos de teste de software, qual seu uso e formas de medição. Para isso, os objetivos específicos são:

a) Definição sobre os passos necessários para planejamento e construção de um sistema;
b) Pesquisar sobre os passos envolvendo um teste de software;
c) Entender os tipos de teste de software que podem ser executados;
d) E por fim, analisar o tipo de teste de software mais adequado a cada caso.

1.3 Organização do Trabalho

O trabalho está dividido da seguinte forma:

a) No capítulo 2 abordam-se os conceitos necessários ao entendimento do trabalho no referencial teórico;
b) No capítulo 3 abordam-se como as ferramentas podem ser utilizadas para fazer testes em sistemas;
c) No capítulo 4 apresenta-se como o uso das ferramentas podem indicar quais itens precisam ser melhorados de forma a entregar o máximo possível de qualidade;
d) E por fim, no capítulo 5 apresentam-se as conclusões finais e sugestão de trabalho futuro.

2 DESENVOLVIMENTO

Neste capítulo são explicados os temas referentes a construção do trabalho e que permitem que os objetivos específicos sejam atendidos, como a definição do processos necessários ao desenvolvimento de projetos, os tipos de testes de software existentes e algumas de suas ferramentas para análise.

2.1 Gerenciamento de Projetos

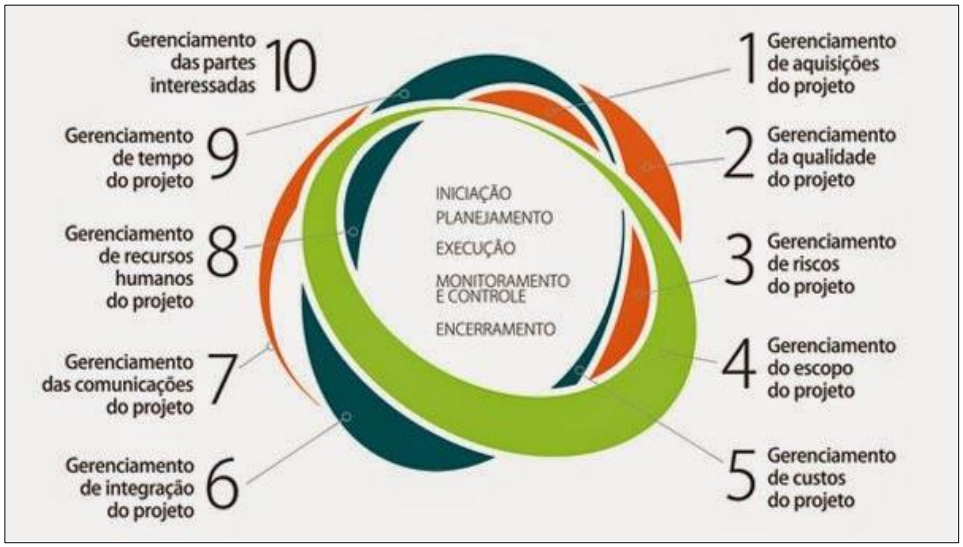
De acordo com a metodologia descrita no *Project Management Body of Knowledge* (PMBOK) o gerenciamento de projetos é um esforço temporário de uma ou mais pessoas para se alcançar um objetivo específico e em comum (PMBOK São Paulo, 2022). E conforme o PMI – SC (2022), mesmo os projetos que não são executados de forma contínua possuem um início, um meio e pontos de entrega, mesmo sendo em várias etapas, e que essas etapas se estendam por um período considerável.

Com base nisso, tem-se o conceito de que os projetos são executados por pessoas e que na maior parte dos casos existe uma limitação não apenas de capital humano, como dos demais recursos, como de tempo e financeiro, por isso é importante planejar a boa utilização desses, de acordo com o PMI – SC (2022).

Procura-se então evitar que o projeto tenha uma duração muito grande, aumentando os custos para a empresa que presta o serviço e para a que está fazendo a aquisição do sistema, e também para garantir que o que foi solicitado seja entregue no final.

Dessa maneira, pode-se fazer uso do PMBOK, que é um manual de boas práticas que pode ser aplicado em qualquer tipo e tamanho de projeto, independente do tamanho da organização, auxiliando no controle e planejamento do projeto, e pode ser dividido em 10 áreas, sendo uma delas a Gestão da Qualidade, conforme Figura 1.

Figura 1 – Áreas de gerenciamento de projetos conforme o PMBOK



Fonte: Nascimento (2014).

1. Gerenciamento de aquisições do projeto: de acordo com Justo (2016), a empresa pode optar por adquirir algum produto ou serviço externo que esteja fora da sua área de desenvolvimento, passando a oferecer esse serviço para seus clientes.
2. Gerenciamento de qualidade do projeto: o gerenciamento da qualidade, segundo Justo (2015), é a medida de quanto um projeto atende aos requisitos especificados no escopo do projeto, e em geral utiliza-se a sigla SMART para medir a qualidade de cada requisito, que é um acrônimo das iniciais de cada palavra, onde:
   a) Específico (*Specific)*: requisitos descritos de forma clara;
   b) Mensurável (*Menrurable)*: cada requisito deve ter uma forma de poder ser medido de forma numérica, para se verificar se está dentro dos padrões de qualidade. Um exemplo seria quantidade total de execuções dividido pela quantidade de acertos, com isso teríamos uma média de acerto do sistema e quanto mais alta a medida, maior a qualidade do mesmo.
   c) Atingível (*Attainable*/): os requisitos definidos devem ser possíveis de serem alcançados, por exemplo, não pode ser definido um requisito considerando comunicação com outro sistema se o mesmo não oferece suporte ou API para que isso seja executado, ou se não há *hardware* adequado para execução dos algoritmos.
   d) Relevante (*Relevant)*: os requisitos precisam ser relevantes e contribuir para que o escopo do projeto seja alcançado.

e) Temporal (*Time-bound*): deve haver um prazo delimitado para que o requisito seja atendido.

Algumas das ferramentas utilizadas para medição da qualidade podem ser: coleta de dados, questionários, análises de desempenho, testes e avaliações, amostragem estatística, reuniões, entre outras (Justo, 2015).

3. Gerenciamento de riscos do projeto: cada projeto apresenta riscos diferentes, como por exemplo o risco de invasão e alteração dos dados, riscos de cancelamento do projeto, risco de legislações serem alteradas, entre outros. Alguns riscos podem e devem ser mitigados, enquanto que com outros é necessário ter outra abordagem.
4. Gerenciamento de escopo do projeto: essa é a delimitação do que será feito no projeto, abordando os requisitos que devem ser completados e também o que o projeto não irá executar.
5. Gerenciamento de custos do projeto: conforme Montes (2020), há uma série de custos envolvidos na execução de um projeto, sendo: capital humano, máquinas e equipamentos, compra de softwares externos, consultoria, entre outros.
6. Gerenciamento de integração do projeto: muitas vezes o projeto será integrado com outros módulos já existentes, exigindo que seja utilizado tecnologias similares ou que o sistema que está sendo desenvolvido considere os parâmetros ou banco de dados já existentes.
7. Gerenciamento de comunicações do projeto: em um projeto, principalmente os que exigem um time maior, existem informações que são coletadas com o cliente e analistas e repassadas ao setor técnico e gerente de projeto, e é necessário que essas sejam distribuídas de forma clara, sem ruído e que retratem o que foi acertado com o cliente, pois muitas vezes a área de execução não tem contato com o cliente, e precisa executar o que foi combinado com ele através da documentação desenvolvida.
8. Gerenciamento de recursos humanos do projeto: o gerenciamento de recursos aborda a boa utilização dos equipamentos envolvidos e também do capital humano, através, por exemplo, de passar cada tarefa para o profissional mais capacitado para a mesma.
9. Gerenciamento de tempo do projeto: em geral o tempo de entrega de um projeto é definido no seu escopo, e através do planejamento do mesmo é definido um

cronograma de entregas, conforme essas forem acontecendo é essencial que o cronograma seja ajustado devido a algumas tarefas serem desenvolvidas em menor tempo e outras com atraso, sempre visando que o projeto seja entregue no prazo.

10. Gerenciamento das partes interessadas: entre as partes interessadas tem-se o cliente, a empresa que foi contratada para desenvolver o projeto e todos os colaboradores que farão parte disso. Dessa maneira é importante o desenvolvimento de estratégias para que todos estejam engajados durante a execução do projeto, conforme Montes (2021).

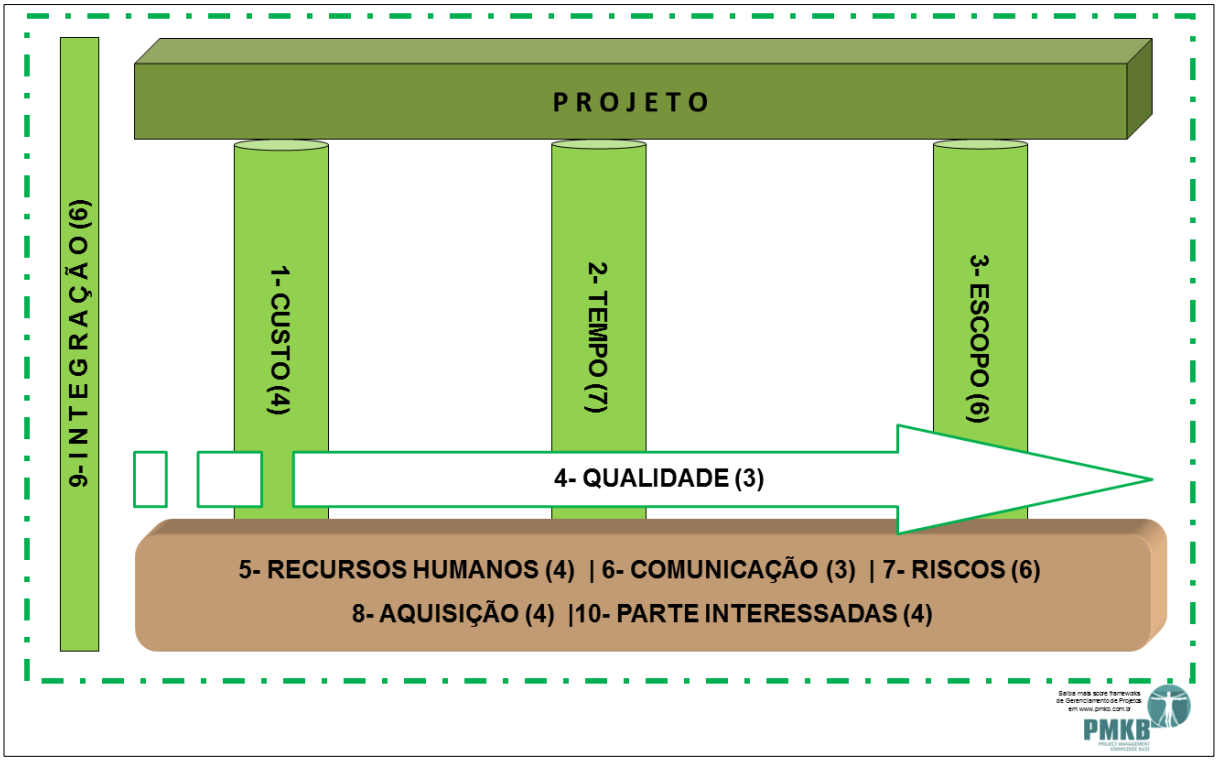
E todas essas formas de gerenciamento estão em torno do núcleo do projeto, que seria seu início, planejamento, execução, monitoramento e controle e encerramento, sendo:

- Início do projeto: será definido o escopo do projeto, ou seja, o que será feito, seus requisitos, qual o valor e prazo final.
- Planejamento: define o cronograma de entregas e o que deverá ser entregue em cada etapa, e por quem.
- Execução: onde o projeto será de fato desenvolvido.
- Monitoramento e controle: acompanhamento da execução do projeto com base em seu cronograma, sendo muitas vezes necessário fazer ajustes no mesmo de forma que atrasos sejam compensados e o projeto seja entregue no tempo previsto, e também são realizados testes de qualidade, retornando atividades para execução caso não apresentem a qualidade mínima definida.
- Encerramento: a formalização de entrega do projeto, e novas funcionalidades devem seguir novamente esse ciclo.

Com base nisso, tem-se maior facilidade de gerenciar o projeto como um todo.

E finalmente, todo o projeto tem relação com sua qualidade, sendo esse um fator atrelado com os custos, o tempo de desenvolvimento e o escopo, e vários fatores auxiliam isso como a comunicação entre os envolvidos, os riscos, os recursos humanos, aquisição e parte interessada, conforme Figura 2.

Figura 2 – Relação da qualidade com outros fatores do projeto



Fonte: PMBOOK (2022).

2.2 Testes de Software

Os testes de software podem ser aplicados a várias áreas do desenvolvimento, como acessibilidade, segurança, usabilidade, entre outros. O ideal é que os projetos façam pelo menos alguns testes dentro de cada área de forma a assegurar um funcionamento adequado do produto que será entregue ao cliente.

2.1.1 Teste de Acessibilidade

Conforme Manual W3C (2021), a acessibilidade em sistemas se refere ao acesso do mesmo de maneira igualitária tanto por pessoas com deficiência quanto pelas que não as tem. Para isso, é necessário que os sistemas sejam adaptados conforme algumas normas e padrões.

Fenner (2018) menciona que a acessibilidade não beneficia apenas pessoas com deficiência, mas também idosos que vão perdendo a visão ou tendo limitações de

movimentos conforme envelhecem, assim como pessoas que preferem fazer uso da tecnologia de outra forma, como por exemplo ver um vídeo com legenda enquanto está em um ambiente com mais pessoas e sem fone de ouvido.

Assim, nem sempre a forma padrão é a maneira que a maior parte das pessoas prefere acessar o conteúdo, por exemplo, o uso do mouse por muitas horas causa dores em alguns usuários, a tela branca e com brilho pode causar dor de cabeça e dor nos olhos, entre outros.

Com base nisso, Godinho (2010) define os termos de acessibilidade a serem abordados dentro de 5 categorias: visual, motora e mobilidade, auditiva, convulsão e cognitiva e intelectual.

2.1.1.1 Dificuldades visuais

As dificuldades visuais incluem desde a baixa visão, tipos de daltonismo até cegueira, e Godinho (2010) menciona que para esses usuários uma opção pode ser os *softwares* conhecidos como leitores de tela, que enviam as informações sobre a página de forma auditiva e ainda pode existir um terminal braille externo, tal como demonstrado na Figura 3.

Figura 3 – Computador portátil com terminal em braile

Fonte: Godinho (2020).

Porém, os sistemas leitores de tela não resolvem todos os problemas, já que não é possível fazer a leitura de um gráfico ou texto em formato de imagem. Nesse caso uma outra opção seria a possibilidade de aumentar o tamanho das letras e conteúdo em geral da página.

2.1.1.2 Dificuldades motoras

Segundo Godinho (2010) as dificuldades motoras podem ser causadas por tendinites, paralisias cerebrais, perda de membros ou dedos, artrites, tremores musculares, doença de Parkison, distrofia muscular, entre outros. Essa fraqueza ou redução muscular pode tornar uma tarefa difícil o uso de teclados e mouses, bem como de ações que exijam maior agilidade.

Assim, alguns sistemas eliminam o uso do mouse através de telas sensíveis ao toque, ou ao reconhecimento de voz, que permite a execução de comandos.

2.1.1.3 Dificuldades auditivas

Os problemas auditivos vão desde a surdez até pouca audição, dessa maneira conteúdos que envolvam áudio devem ter legenda em texto ou libras, de forma a tornar esse conteúdo disponível para todos. Além disso, muitas vezes o usuário quer acessar um conteúdo em que não pode ouvir o áudio no momento, e formas alternativas facilitam o seu entendimento.

2.1.1.4 Convulsão ou dificuldades cognitivas e intelectuais

Godinho (2010) menciona que as convulsões podem ser causadas por efeitos estroboscópicos ou pisca-pisca, sendo mais comuns em jogos e filmes. Já a dificuldade cognitiva e intelectual trata das dificuldades de desenvolvimento na questão de adquirir ou manter o conhecimento, o que pode ser causado por problemas de memória, dificuldade lógica e na resolução de problemas e o desenvolvimento da maturidade.

Como forma de contornar essas limitações, pode-se tentar diferentes métodos de aprender, pois cada indivíduo tem maior facilidade em adquirir conhecimento de uma maneira, como por exemplo, através de vídeo aulas, através da leitura de um texto ou exercícios de fixação.

Com base nisso, quanto mais formas de acessar o mesmo conteúdo estiverem disponíveis, maior a chance do usuário encontrar a forma mais adequada para ele.

Com base nisso, muitos usuários podem usar ferramentas conhecidas como tecnologias assistentes para lhes ajudar a navegar pelos sites e sistemas, tal como leitores

de tela, terminais braille, softwares de ampliação de telas e reconhecimento de fala e teclados com superposição, que torna mais fácil a digitação para pessoas com mobilidade reduzida, conforme W3C (2021).

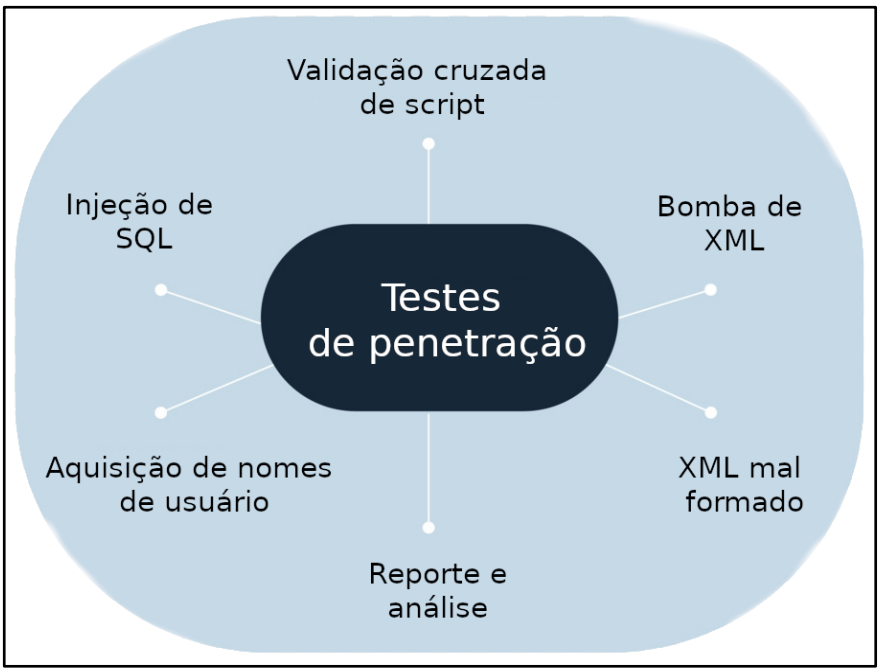
Porém, para que essas ferramentas tenham um bom funcionamento, é necessário que o site ou sistema atenda alguns padrões básicos, tal como as imagens terem legendas no atributo "*alt*", entre outros.

2.1.2 Teste de Segurança

De acordo com o Portal BayMetrics (2022), os testes de segurança se referem ao processo de verificar se existem vulnerabilidades nos sistemas desenvolvidos, ou alguma brecha que possa servir para que um atacante tenha acesso ao sistema como um usuário registrado, ou a base de dados ou servidor.

Como garantir a segurança total de um sistema não é algo viável, devido a constante evolução das tecnologias, deve-se ao menos procurar garantir que o sistema não esteja vulnerável aos problemas de segurança mais conhecidos, como *sql injection* ou XSS, conforme representado na Figura 4.

Figura 4 – Principais vulnerabilidades de segurança



Fonte: Adaptado, BayMetrics (2022).

De acordo com o Portal BayMetrics (2022), existem dois principais componentes que podem ser avaliados com relação aos testes de segurança, sendo:

- Avaliação de vulnerabilidades: também chamados de “*scanners* de vulnerabilidade”, consistem em testes de segurança automatizados com o objetivo de descobrir falhas nas aplicações ou infraestrutura, cobrindo as vulnerabilidades mais conhecidas.
- Teste de penetração: essa é uma avaliação manual executada geralmente por profissionais de segurança da informação que determinam até que ponto um possível atacando conseguiria ir, sendo utilizado para encontrar pontos fracos que não foram identificados pelas ferramentas automatizadas, o que pode incluir métodos de engenharia social também.

Tem-se então que esse teste exige um certo grau de conhecimento especializado, e quanto mais tempo dedicado maiores as vulnerabilidades que podem ser encontradas, o que também do tempo disponível para execução dos mesmos.

2.1.3 Teste de Performance

Conforme Iguera (2022), esse teste também pode ser chamado de “carga” ou “desempenho”, e seu objetivo é medir a performance de uma aplicação, cada uma delas avaliando um item. Dessa maneira o teste de performance vai avaliar como a aplicação se comporta na realização de uma determinada tarefa, como por exemplo o tempo que leva para se comunicar com a administradora de cartão para concluir (ou não) um pagamento, o tempo que leva para carregar as informações em uma página, entre outros.

O teste de carga verifica qual o limite de usuários simultâneos ou transações antes de ficar fora do ar, lento ou com problemas de carregamento dos dados. Com isso pode-se avaliar diversos indicadores de desempenho como uso da memória, processamento, armazenamento, acesso simultâneo ao banco de dados, entre outros.

O teste de stress é de certa forma um complemento ao de capacidade, pois verifica como a aplicação irá se comportar caso sejam estourados os limites da aplicação, proporcionando um limite de capacidade de qualidade definido.

Assim, esses testes são executados dentro de um ambiente controlado e monitorado, com diferentes volumes de transações ou usuários simultâneos de forma a entendermos qual seu limite, para que, ao criar campanhas de marketing tenhamos em

mente se será necessário a contratação de servidores ou recursos adicionais, conforme Iguera (2022).

Complementares a esses tem-se ainda testes de escalabilidade, resistência e concorrência, de acordo com Iguera (2022). O teste de escalabilidade permite verificar qual a capacidade de escalabilidade da aplicação, o de resistência analisa o comportamento da aplicação em um longo período de tempo e o de concorrência verifica como fica a qualidade da aplicação no uso concorrente de recursos.

Por fim, apresentamos na Tabela 1 uma comparação entre os testes de performance, que englobam os itens mencionados até o momento, e os testes funcionais, que também testam o comportamento da aplicação, mas voltado a um usuário.

Tabela 1 – Comparativo entre os testes de performance e funcionais

Teste de Performance	**Teste Funcional**
Não testa o front-end da aplicação, ou seja, as funcionalidades.	Testa o front-end da aplicação, bem como sua usabilidade e funcionalidades.
Testa a escalabilidade da aplicação e monitora o uso dos recursos de hardware.	Não testa a escalabilidade da aplicação e monitora o uso dos recursos de hardware.
Projetado para determinar como uma aplicação/ sistema irá realizar ao longo do tempo.	Não pode determinar como uma aplicação/ sistema irá realizar ao longo do tempo.
Requer uma aplicação totalmente funcional para que os cenários sejam executados adequadamente.	Não requer uma aplicação totalmente funcional para que os cenários sejam executados adequadamente.
Vários usuários	Um usuário

Fonte: Campos (2015).

As principais vantagens desses testes são, conforme Campos (2015): redução do custo de mudanças e de sistema, aumento dos lucros, aumento da satisfação do usuário, controle total do desempenho que o sistema pode proporcionar, entre outros.

Já como restrições, tem-se também que existe uma grande complexidade na elaboração desses testes, requer a simulação de uma grande variedade de cenários e que as ferramentas de automação muitas vezes tem um custo elevado, o ambiente real de

produção nunca pode ser completamente simulado, e é necessário um ambiente adequado para simulação, segundo Campos (2015).

2.1.4 Teste de Usabilidade

Os testes de usabilidade avaliam o uso real do sistema pelos usuários, e o objetivo é observar como eles se saem utilizando a ferramenta, com isso pode-se descobrir problemas e pontos de melhoria, conforme o Portal Caelum (2022). Nesse ponto pode-se utilizar protótipos para validar a interface que está sendo criada.

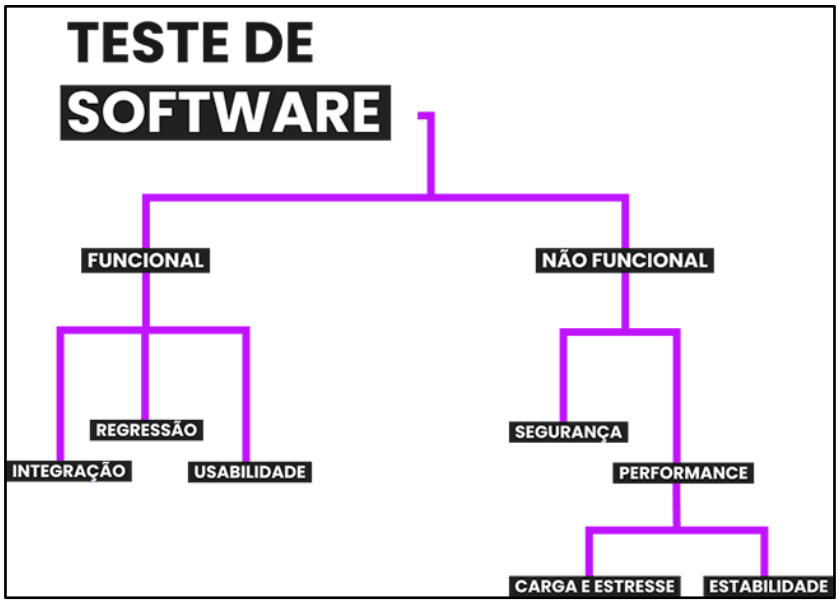
Com isso, pode-se fazer as seguintes medições:

- Desempenho: quantidade de cliques necessários para se executar uma tarefa;
- Precisão: quantidade de erros que o usuário cometeu, como por exemplo clicar nos itens errados ou dificuldade em recuperar informações do sistema;
- Lembrança: se em uma próxima utilização o usuário se recorda como executar determinada ação;
- Emocional: como o usuário se sentiu após realizar uma ação, foi difícil, fácil, estressante, gratificante, entre outros.

Dessa forma, os avaliadores costumam dar uma lista de tarefas para os usuários, e observam ou filmam como eles se saem, verificando o que pode ser melhorado, telas que podem ser reduzidas ou suprimidas, layouts que podem ser melhorados, botões que podem ser modificados, entre outros, conforme o Portal Caelum (2022).

Finalmente, a Figura 5 resume os principais tipos de testes, e cabe a equipe de projeto verificar em quais testes deve-se focar com maior ênfase, conforme os riscos identificados no projeto, pois a execução aprofundada de todos demandaria um custo e tempo que poucos clientes estariam dispostos a pagar, além de um maior tempo de desenvolvimento.

Figura 5 – Testes de software



Fonte: Portal Ivory IT (2021).

2.2 Ferramentas de Análise

As ferramentas reúnem uma série de diretrizes que auxiliam no desenvolvimento de páginas que estejam aptas a ter um bom desempenho em alguns testes, como por exemplo os de acessibilidade e usabilidade, sendo a principal delas a W3C.

2.2.1 W3C

Segundo o W3C (2021), ele é um consórcio internacional cujo objetivo é o desenvolvimento de padrões para a web, sendo todos livres e gratuitos, tendo iniciado suas atividades em 2008. Atualmente existem cerca de 100 padrões que podem ser utilizados como base para o desenvolvimento de interfaces que se adaptem em diversos dispositivos.

Seus princípios são baseados na equiparação tanto quanto possível das possibilidades de uso, flexibilidade de uso, uso simples e intuitivo dos sistemas, tolerância ao erro e mínimo esforço para uso. Com base nisso visa-se que todos os usuários tenham uma experiência agradável na utilização dos sistemas, sem dificuldades ou limitações, independente de sua mobilidade, por exemplo.

Uma ferramenta que pode ser usada em conjunto com esses padrões é o Google Lighthouse[1], que mede a qualidade das páginas web, audita seu desempenho e oferece uma nota a cada categoria avaliada, sugerindo inclusive mudanças e alterações que podem ser feitas para que a pontuação seja aumentada.

1 https://developers.google.com/web/tools/lighthouse?hl=pt-br

3 FERRAMENTAS PARA TESTES DE SOFTWARE

As ferramentas para testes de software são separadas conforme tipo de teste que será feito, e a profundidade que ele será realizado depende do foco da organização e do risco que o sistema apresenta para a empresa.

3.1 Teste de Acessibilidade

Conforme CTA (2019), para saber se um site ou sistema é acessível pode-se utilizar metodologias para procurar problemas de acessibilidade conforme o manual da W3C, e assim, fazer as correções necessárias. A Figura 6 sugere alguns passos que podem ser feitos, porém não há uma metodologia universal, sendo necessário verificar um conjunto de itens através de mais de um procedimento.

Figura 6 – Avaliação de acessibilidade em sites

Fonte: CTA (2019).

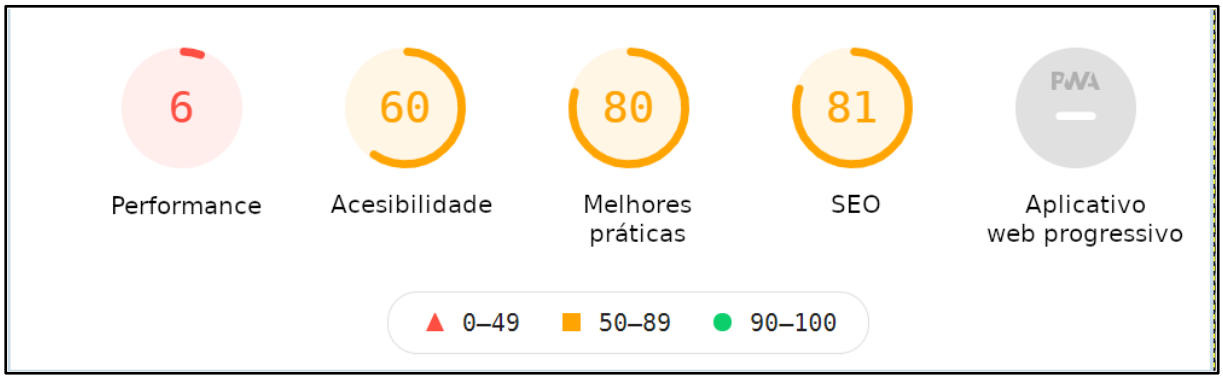
CTA (2019) ainda sugere que seja combinado procedimento manuais com automáticos, pois os automáticos fazem as validações no código fonte considerando as diretrizes do W3C para HTML e CSS, enquanto que as avaliações manuais serão feitas por

usuários que de fato irão interagir com o sistema, e o resultado será conforme seus perfis e habilidades de uso.

Como ferramentas automáticas pode-se mencionar o AccessMonitor e o ASES, que após a inserção do link da página emitem um relatório com os problemas encontrados. Tem-se ainda alguns plugins a serem vinculados com os navegadores como o WAVE AWAVE Accessibility Extension para Firefox[2], Accessibility Developer Tools para Chrome[3], Google Lighthouse[4] e o eScanner para Chrome[5].

O Google Lighthouse é uma das ferramentas mais conhecidas nos testes de acessibilidade, possui código aberto e mede a qualidade das páginas web, emitindo um resultado conforme o demonstrado na Figura 7. Seu uso é feito através de uma extensão para Google Chrome ou por linha de comando, que permite a simulação de outros testes.

Figura 7 – Resultado de acessibilidade pela execução na ferramenta Google Lighthouse



Fonte: Autoria Própria (2022).

Nessa ferramenta, a análise de performance verifica quão rápido a ferramenta carrega os dados e quais são os primeiros elementos apresentados ao usuário, muito útil caso ele esteja em uma conexão com baixo sinal (no celular, por exemplo). Para a acessibilidade é apresentado um relatório com diagnóstico, tal como para as suas

2 https://addons.mozilla.org/en-US/firefox/addon/wave-accessibility-tool/

3 https://chrome.google.com/webstore/detail/accessibility-developer-t/fpkknkljclfencbdbgkenhalefipecmb?hl=en

4 https://developers.google.com/web/tools/lighthouse?hl=pt-br

5 https://chrome.google.com/webstore/detail/escanner/mpiipiobgejghkocofogeonfkapjgfmk?hl=pt-BR

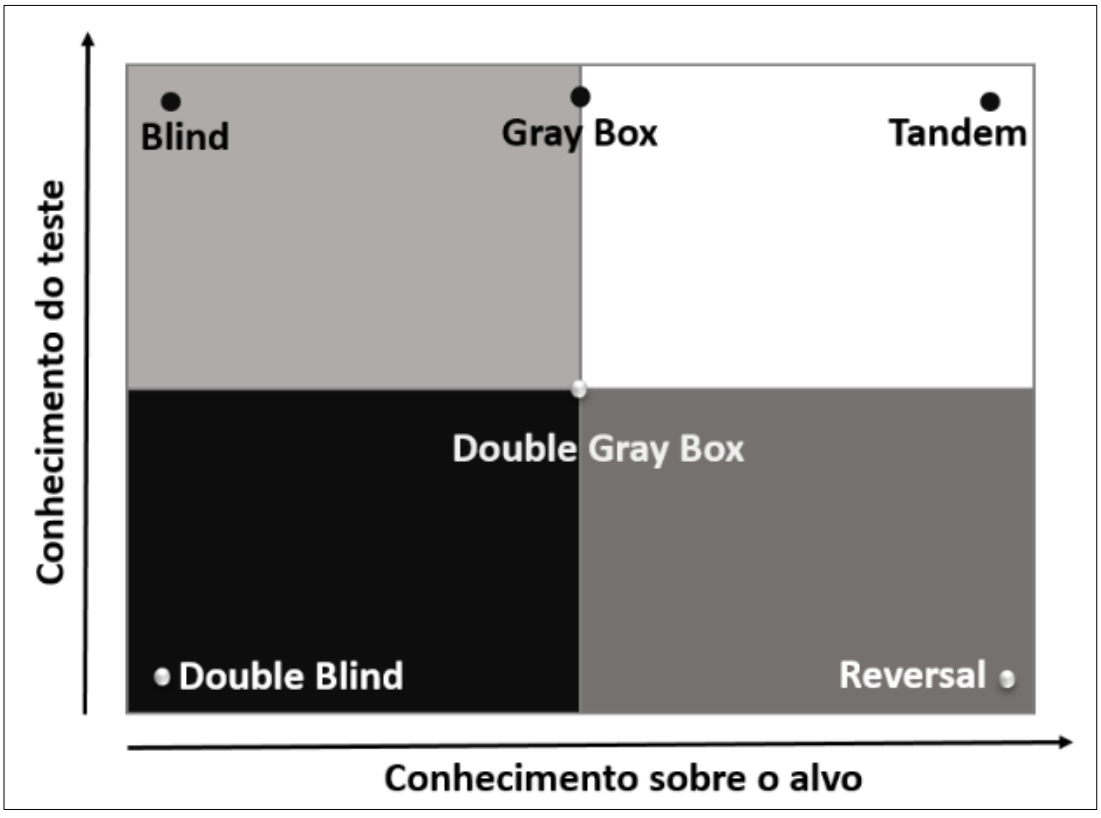
concorrentes. A análise de boas práticas considera o uso do HTTP/2 e outras boas práticas, e por fim, a análise de SEO válida como a página se comporta com os mecanismos de busca, através dos princípios de otimização.

3.2 Teste de Segurança

Através dos testes de segurança consegue-se encontrar falhas nas aplicações web antes que algum usuário mal intencionado faça isso, e cause danos a organização. Tem-se então uma grande quantidade de ferramentas que podem ser utilizadas para fazer alguns testes, como: SQL injection, força bruta, identificação de portas abertas, entre outros.

Porém, antes, precisamos definir qual será o tipo de teste feito, que pode ser caixa branca, caixa preta, ou um combinado dos dois conhecido como caixa cinza, conforme Pereira (2017) e Figura 8.

Figura 8 – Tipos de teste de segurança



Fonte: Pereira (2017).

A diferença entre eles é quanto o possível atacante, ou o colaborador que irá se passar como atacante, sabe sobre o software, os funcionários e a organização. Na modalidade caixa branca ele tem conhecimento sobre a estrutura interna da organização e do código fonte, enquanto que na caixa preta ele é alguém “de fora”, ou seja, que não sabe nada sobre o sistema em si, e o caixa cinza é um intermédio dos dois, podendo ser representado, por exemplo, por ex-funcionários que detém algum conhecimento do funcionamento da empresa.

Quanto tratamos sobre a modalidade caixa preta o ideal é contratar uma auditoria externa, ou um pentester, para realmente simular uma invasão vinda de fora e até onde esse atacante conseguiria chegar.

De acordo com Brunner et al. (2006), o procedimento é dividido em 3 etapas, sendo:

1. Planejamento e preparação: é realizado o planejamento de quais sistemas e módulos serão testados, seu escopo, os aspectos legais, tempo de duração, entre outros;
2. Avaliação: fase em que os testes de penetração são de fato realizados, onde tem-se as seguintes sub fases:
    1. Coleta de informações;
    2. Mapeamento de rede;
    3. Identificação de vulnerabilidades;
    4. Penetração e intrusão dos sistemas;
    5. Acesso aos níveis de privilégio;
    6. Comprometimento de usuários remotos;
    7. Manutenção do acesso (para que em um próximo acesso/ ataque os passos não precisem ser executados novamente);
    8. Esconder rastros, apagar logs, entre outros.
3. Relatório: resultado final do que foi encontrado durante a análise com sugestões de correção.

Quanto a coleta de informações sobre o alvo, existem algumas ferramentas que podem auxiliar, e com base nas informações coletadas, e possíveis falhas descobertas, o profissional já inicia seu trabalho. Na Tabela 2 demonstramos algumas dessas ferramentas.

Tabela 2 – Ferramentas para coleta de dados

Ferramenta	Descrição
Netcat [29]	O netcat é uma ferramenta de rede *open source*. É usada para ler e escrever dados em conexões de rede usando o protocolo TCP/IP. É considerado o canivete suíço do TCP/IP, pois pode ser usado para fazer desde análise de porta até ataque por força bruta.
Httprecon [30]	Httprecon é uma ferramenta para impressões digitais de servidor *Web*. O objetivo é a identificação de determinadas informações do httpd.
SET [31]	O SET (*Toolkit de Social-Engineer*) Trata-se de uma ferramenta *open-source Python* para testes de penetração. É utilizado para a coleta de informações por meio da engenharia social.
Whois [32]	Ferramenta para a coleta de informações sobre o proprietário de um domínio e outras informações.
DIG [33]	O DIG (*Domain Information Groper*) é uma ferramenta para coletar os nomes de DNS de determinado domínio ou *host*.
Maltego [34]	O Maltego é uma ferramenta que coleta informações de varias fontes publicas e relaciona os dados coletados para análise. É uma plataforma única que mostra todo o ambiente de uma organização ou rede.

Fonte: Pereira (2017).

3.2.1 *SQL Injection*

Uma falha bem comum em sistemas web é deixar os campos dos formulários HTML livres para receber qualquer tipo de dado, e enviá-los diretamente para o banco de dados, sem nenhum pré-processamento, com isso, caso seja enviado trechos de código ou caracteres inválidos, o atacante pode ter acesso a todo o banco de dados.

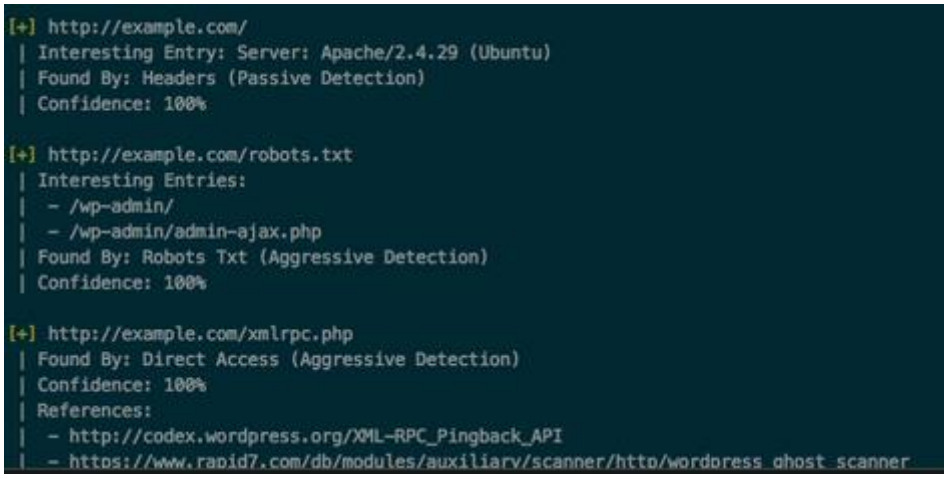
Algumas das ferramentas que podem ser utilizadas para fazer esses testes nos campos dos formulários são: SQLMap[6], jSQLInjection[7], NoSQLMap[8], entre outros.

3.2.2 *Wordpress*

Muitas empresas fazem uso de sites desenvolvidos em Wordpress para sites institucionais ou algo rápido, porém, como é um sistema de código aberto, permite que qualquer pessoa com tempo disponível consiga buscar e encontrar vulnerabilidades, principalmente considerando a quantidade de módulos externos e templates existentes.

Inclusive existe uma ferramenta voltada exclusivamente a ataques envolvendo sites em Wordpress, que pode mostrar as vulnerabilidades encontradas em todos os módulos (com links sobre como corrigir ou entender mais sobre o problema), mostra de páginas de login e também ataques de força bruta em formulários de acesso com base em dicionário de dados, a ferramenta é chamada de WPScan[9] e seu scan pelo site apresenta um resultado similar ao da Figura 9.

Figura 9 – Exemplo de resultados apresentado pela ferramenta WPScan



Fonte: Autoria Própria (2022).

6 https://sqlmap.org/

7 https://github.com/ron190/jsql-injection

8 https://github.com/codingo/NoSQLMap

9 https://wpscan.com/wordpress-security-scanner

3.2.3 Outras Vulnerabilidades

Algumas outras vulnerabilidades que podem ser identificadas, e que podem ter grande impacto no sistema são apresentadas na Tabela 3.

Tabela 3– Ferramentas para exploração de vulnerabilidades

Ferramenta	Descrição
Nmap [35]	O Nmap (*Network Mapper*) é uma ferramenta para varredura de portas e serviços em uma rede ou em um *host*. Ele lista uma tabela de portas interessantes com o número das portas, o estado e os serviços disponíveis.
Nessus [36]	É um *scanner* de vulnerabilidades para aplicações *Web*. Ele realiza a varredura de portas, detectando servidores ativos e simulando invasões para detectar vulnerabilidades. Ele é composto por um cliente e um servidor, sendo que o *scan* propriamente dito é feito pelo servidor.
Nikto [37]	É uma ferramenta desenvolvida em *Perl*, cujo objetivo é fazer varredura de vulnerabilidades em servidores *Web*. Foi desenvolvida para encontrar diversos tipos de arquivos, configurações e programas padrões ou inseguros, em servidores *Web*.
Acunetix WVS [38]	É um scanner de vulnerabilidades para aplicações *Web*. Ele verifica automaticamente vulnerabilidades como injeções SQL, *cross site scriptinge* senha fraca em páginas de autenticação.
AppScan [39]	É um *scanner* de vulnerabilidades para aplicações *Web* e *mobile*. Ele permite identificar vulnerabilidades de segurança e gerar relatórios e recomendações de correção.
NeXpose [40]	Um *scanner* de vulnerabilidades que acompanha todo o ciclo de vida de gerenciamento de vulnerabilidades, incluindo descoberta, detecção, verificação, classificação de risco, análise de impacto, relatórios e mitigação.

Fonte: Adaptado, Pereira (2017).

Nesse teste, é importante realizar alguns testes de segurança com pelo menos algumas dessas ferramentas, pois como são as mais conhecidas, espera-se que um atacante real tenha conhecimento delas, e tente as utilizar contra os sistemas da empresa.

3.3 Teste de Performance

Através dos testes de performance pode-se avaliar como a aplicação vai se comportar, principalmente em casos extremos. Alguns dos itens que podem ser medidos são, conforme Correia e Couto (2015):

1. Tempo que o sistema leva para “pensar”, ou seja, fazer a requisição ao banco de dados e montar a página. Nesse processamento podem ser injetados atrasos e paradas nas variáveis do sistema para simular uma carga mais realista;
2. Cache do navegador: o que permite que requisições estáticas como imagens, folhas de estilo e áudio sejam carregadas apenas uma vez, pois são itens que não são alterados com frequência.
3. Concorrência e número de iterações: vários usuários utilizando a aplicação de uma única vez;
4. Teste de capacidade: mede até que o ponto o sistema fica sobrecarregado e começa a falhar;
5. Testes de tolerância: avaliam a performance do sistema sob condições normais de carga através de um longo período de tempo;
6. Testes de volume: qual o número de transações por minuto que um sistema suporta;
7. Teste de stress: descobrir até que o ponto o sistema funcionará sem sair do ar.

Como ferramenta de testes, uma das mais conhecidas e poderosa é a JMeter[10], segundo Correia e Couto (2015), criada em Java e que consegue automatizar testes em sistemas web e medir o desempenho das aplicações, através da simulação de diversos cenários, tendo como resultado final um relatório similar ao apresentado na Figura 10, e através dele pode-se verificar melhorias, implementações ou trocas de servidor que suportem a performance esperada, dentro das condições esperadas.

10 https://jmeter.apache.org/

Tais testes também são essenciais para dias de muito pico do sistema, como campanhas promocionais, últimos dias de inscrição para eventos, entre outros. Dessa maneira a aplicação não ficará fora do ar nem lenta para os usuários.

Figura 10 – Exemplo de relatório apresentado pelo JMeter

Fonte: Correia e Couto (2015).

O relatório da Figura 10 pode mostrar para a equipe técnica o desempenho do sistema, e dessa maneira, o que precisam adequar para que o software tenha um bom funcionamento, porém para os gestores e área de tomada de decisão, os resultados costumam ser demonstrados como no Quadro 1, de maneira também a justificar gastos com aquisição de novos equipamentos.

Com base nisso, um gestor pode visualizar que se um usuário está esperando que uma página carregue para poder concluir uma compra, ou está dando erro, ele pode desistir da compra e ir fazê-la em outro site, o que representaria um prejuízo financeiro.

Quadro 1– Exemplo de relatório a ser apresentado para a área de tomada de decisão

Tempo de Carregamento	Percentual de Usuários Esperando
10 segundos	84%
15 segundos	51%
20 segundos	26%
30 segundos	5%

Velocidade do Modem	Tempo Esperado de Carregamento
14.4 Kilobytes/ Modem	11.5 segundos
33.6 Kilobytes/ Modem	7.5 segundos
56 Kilobytes/ Modem	5.2 segundos
Modem Cabo/ DSL	2.2 segundos
T1 e superior	0.8 segundos

Velocidade	Vendas Perdidas (em milhões de dólares)
14.4 Kilobytes	73
28.8 Kilobytes	97
56 Kilobytes	100
ISDN	14
T1	38

Tempo de Resposta	Visualização do Usuário
< 0.1 segundo	O sentimento dos usuários é de que o sistema está respondendo instantaneamente.
< 1.0 segundo	Experiência do usuário está um pouco comprometida, mas ele ainda mantêm o foco no site
< 10 segundos	Esse é o máximo de tempo que os usuários mantêm foco no site, porém sua atenção já está na **zona de distração**.
> 10 segundos	Usuários estão mais propensos a serem distraídos com outros sites ou perderem o interesse.

Fonte: Adaptado, Campos (2015).

Outro ponto apresentado é referente ao tempo de resposta do sistema, que caso demore muito o usuário pode se distrair, ir navegar em outras páginas web e esquecer o que estava fazendo nessa, ou simplesmente perder o interesse.

3.4 Teste de Usabilidade

Através dos testes de usabilidade é possível entender o comportamento do usuário no sistema, e ver se ele consegue utilizá-lo de forma simples, sendo que o objetivo geral é testar a aplicação. Esse teste pode ser feito de diversas formas, de acordo com Pinheiro (2021) e Estrella (2019).

1. Teste de navegação: verifica se todas as páginas são compreensíveis e simples de usar, considerando-se a navegação, campos, botões, acesso para voltar ao menu principal, entre outros.
2. Teste de conteúdo: verificação de erros de gramática e ortografia nos conteúdos publicados, inclusive em outros idiomas, caso exista esse suporte. Imagens e seus tamanhos também são analisados, paleta de cores para que todas as informações estejam claras e o conteúdo compreensível e estruturado e links implementados de forma lógica.

Com base nesses testes, podem ser aplicados alguns métodos, como o *In-Person* ou *In-House* e os testes remotos. No caso do teste *In-Person* ou *In-House*, ele é conduzido no local e moderado pelos pesquisadores enquanto os participantes testam itens item específico do site, como por exemplo tentar realizar uma ação. Já no teste remoto o usuário faz o teste onde se sentir mais confortável, sem ser monitorado diretamente, para que não haja pressão e o teste possa ser mais realista, ele consiste em coletar os dados do que as pessoas vêem através de webcans, verificando onde os usuários olham primeiro e suas expressões faciais, conforme Estrella (2021).

Dentro desses testes, podem ser aplicadas algumas técnicas populares como:

- Rastreamento do olhar: verificando onde os participantes olham primeiro em uma página;
- Verificação de cartões: usada para construção de navegação ou criação de rótulos;

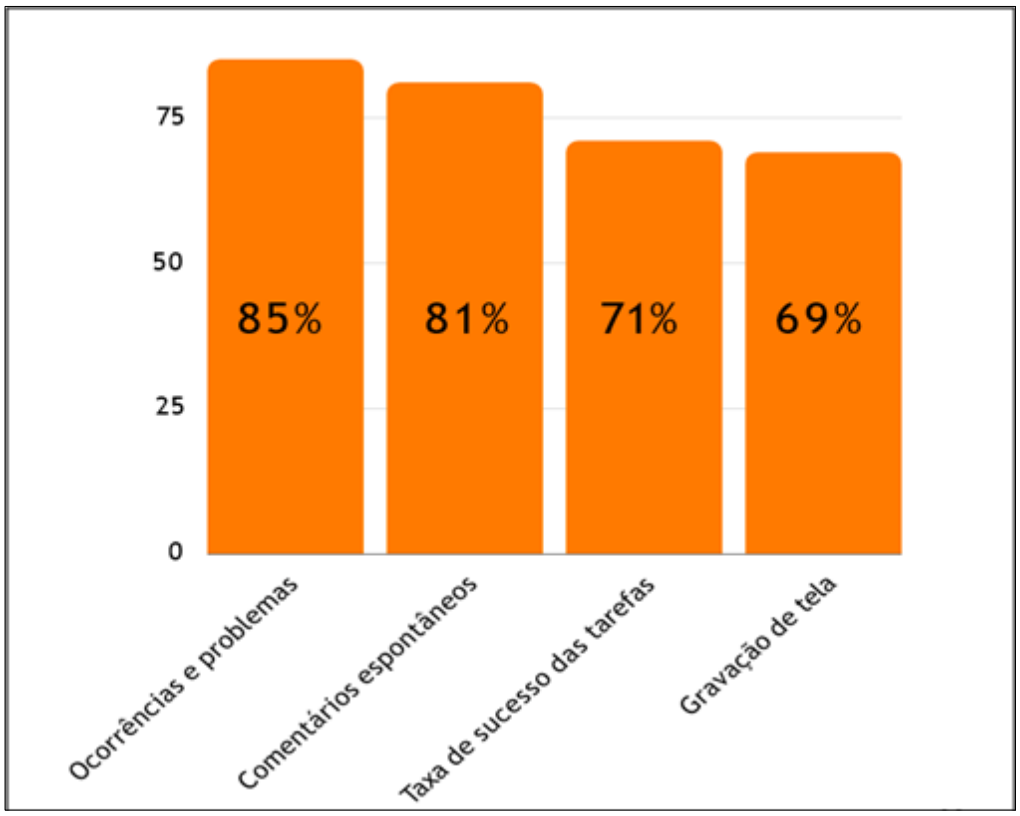
- Teste A/B: faz a comparação de tipos de layouts.

Como passo a passo para aplicação de um teste de usabilidade, tem-se os seguintes passos, de acordo com Estrella (2021):

1. Definir um escopo, ou seja, definir o que será avaliado, como por exemplo, como os usuários se comportam na compra de uma passagem aérea ou para fazer cadastro em um site;
2. Escolha do método de aplicação: por exemplo criar uma lista de atividades para que os participantes executem e observar ou gravar sua tela durante o teste, o que irá permitir sanar dúvidas sobre o comportamento e dificuldade que eles terão ao executar determinada ação.
3. Criar um cenário: cenário que será proposto;
4. Taxa de sucesso: métrica para medir se a ação foi concluída com sucesso, como por exemplo concluir um cadastro ou emitir uma passagem;
5. Encontrar os participantes: podem ser pessoas aleatórias ou personas ao qual o serviço se destina, interessante procurar pessoas de diferentes faixas de idade e conhecimentos em informática para ver como todos se sairiam no uso da ferramenta, e encontrar alguma dificuldade que possa ser melhorada;
6. Condução do teste: ele pode ser monitorado através de observação, ou o usuário pode ser filmando, ou ter sua tela filmada para análise posterior;
7. Analisar o relatório: observar tudo o que foi identificado, principalmente os pontos onde os usuários ficaram perdidos e propor correções;

A pesquisa de Volpato (2019) e demonstrada na Figura 11 apresenta algumas métricas que podem ser coletadas durante um teste de usabilidade, permitindo posterior avaliação da satisfação (ou não) do usuário em realizar determinadas tarefas em um site.

Figura 11– Métricas e dados que podem ser coletados em testes de usabilidade para posterior análise



Fonte: Autoria Própria (2022).

Dessas, as métricas mais utilizadas são:

- Ocorrências de problemas: quando o usuário se sente perdido e não sabe como sair daquela tela, ou voltar a tela inicial, ou não sabe como resolver algum problema da página;
- Comentários espontâneos como exclamações ou xingamentos;
- Gravação da tela: conforme mencionado para análise posterior;
- Taxa de sucesso nas tarefas, ou seja, o usuário conseguir executar o que estava tentando fazer.

Com base nisso, a equipe pode identificar os principais problemas encontrados na utilização do sistema e fazer os ajustes para que se uso se torne mais fácil e intuitivo.

4 CONCLUSÃO

O teste de software é uma fase muito importante do ciclo de vida de desenvolvimento de software. O teste de software é um elemento vital e pode fornecer excelentes resultados se o teste for feito de forma adequada e eficaz. Infelizmente, o teste de software geralmente é menos formal e rigoroso do que deveria e a principal razão para isso é porque tem-se lutado para definir as melhores práticas, metodologias, princípios e padrões para o teste de software ideal. Para realizar testes de software de forma eficaz e eficiente, todos os envolvidos com testes devem estar familiarizados com os objetivos básicos de teste de software, importância, tipos, limitações e conceitos. Já muito trabalho foi feito neste campo, e ainda continua até hoje.

A realização de testes para identificar falhas no código é uma tarefa importante no desenvolvimento de software que tem recebido pouca atenção. Apresenta-se então uma pesquisa teórica acerca de vários tipos de testes de software, e depois ferramentas e como eles podem ser realizados, de maneira a garantir a qualidade da entrega. Importante citar que quanto mais a fundo cada teste for feito, mais é possível garantir o bom funcionamento do software.

No teste de software existem diferentes tipos de testes que são feitos para diferentes propósitos para que um produto de qualidade seja desenvolvido. O principal objetivo por trás disso é desenvolver software livre de erros. A verificação adequada levará a menos problemas presentes durante a fase de validação, pois a maioria dos problemas já foi descoberta e corrigida durante a verificação e a validação adequada garantirá que o aplicativo de software seja desenvolvido de acordo com os artefatos de software finalizados na verificação. Portanto, é importante conhecer todas as atividades de teste, pois o processo de Validação e Verificação anda de mãos dadas e reduz o retrabalho e custo futuros.

4.1 Trabalhos Futuros

Por fim, como trabalho futuro tem-se o objetivo de aplicar os métodos de qualidade propostos em um site real, de forma a verificar seu nível de qualidade a respeito de cada tipo de teste, como por exemplo o atual site da instituição de ensino, onde pode-se aplicar os testes com as seguintes propostas:

- Teste de segurança: de forma a identificar se a área interna de professores e notas não pode ser acessado indevidamente por alunos mal intencionados;
- Teste de acessibilidade: como a instituição possui conhecimento sobre os alunos com deficiência, pode-se tomar isso como base estatística para ajustes no site que facilitem seu uso por esses indivíduos;
- Testes de usabilidade: funcionamento do site em dispositivos mobile;
- Testes de performance: em momentos de grande pico, como inscrição para o vestibular ou consulta de notas pelos alunos.

REFERÊNCIAS

ARIANE-5G. **Gunter's Space Page**. Disponível em <http://space.skyrocket.de/doc_lau_det/ariane-5g.htm>. Acessado em 07 de março de 2022. 2014.

BRUNNER, M. & DILAJ, M. & HERRERA, O. & BRUNATI, P. & SUBRAMANIAM, R. & RAMAN, CHAVAN, S. U. & RATHORE, B. **Information systems security assessment framework (ISSAF) draft 0.2.1.** ISSAF. Disponivel em <http://www.oissg.org/downloads/issaf-0.2/information-systems-security-assessment-framework-issaf-draft-0.2.1/view.html>. Acessado em 25 de maio de 2022. 2006.

CAMPOS, Fábio Martinho. **Teste de desempenho: Conceitos, Objetivos e Aplicação - Parte 1.** Disponível em <http://www.linhadecodigo.com.br/artigo/3256/teste-de-desempenho-conceitos-objetivos-e-aplicacao-parte-1.aspx>. Acessado em 23 de maio de 2022. 2015.

CTA. **Avaliação de acessibilidade em sites**. Disponível em <https://cta.ifrs.edu.br/avaliacao-de-acessibilidade-em-sites/>. Acessado em 21 de abril de 2022. 2019.

CORREIA, Thiago de Almeida & COUTO, Thiciane. **Teste de Performance em Aplicações Web: Garantindo o Desempenho de uma Aplicação.** Interfaces Científicas - Exatas e Tecnológicas, Aracaju, V.1, N.3. 2015.

CUNHA, Simone Moreira. **Engenharia de Software: uma abordagem à fase de testes**. Monografia de Especialização. Universidade Federal de Minas Gerais, MG. 2010.

ESTRELLA, Carlos. **Teste de Usabilidade de Site: Tudo o que Você Precisa Saber.** Disponível em <https://www.hostinger.com.br/tutoriais/teste-de-usabilidade-site>. Acessado em 22 de maio de 2022. 2019.

FENNER, Priscila. A**cessibilidade na Web: tudo o que você precisa saber sobre o assunto!** Disponível em <https://blog.handtalk.me/acessibilidade-na-web/>. Acessado em 15 de março de 2022. 2018.

GODINHO, Francisco. **Acessibilidade web**. Disponível em <http://www.acessibilidade.net/>. Acessado em 20 de março de 2022. 2018.

IGUERA, André. **Testes de performance: tipos, importância e benefícios.** Disponível em <https://blog.onedaytesting.com.br/testes-de-performance/>. Acessado em 26 de março de 2022. 2022.

JUNCKES, Gabriel Dias e MORGADO, Paulo. **Gerência de riscos em desenvolvimento de software**. Universidade do Sul de Santa Catarina (UNISUL). 2013.

JUSTO, Andreia Silva. **Gerenciamento da qualidade: o que é e como fazer em 3 passos.** Disponível em <https://www.euax.com.br/2015/12/gestao-da-qualidade-em-gestao-de-projetos/>. Acessado em 24 de março de 2022. 2015.

JUSTO, Andreia Silva. **Gerenciamento de Aquisições: o que é e como fazer em 3 passos.** Disponível em <https://www.euax.com.br/2016/01/controlando-desgastes-no-gerenciamento-de-aquisicoes-do-projeto/>. Acessado em 24 de março de 2022. 2016.

MONTES, Eduardo. **Gerenciamento dos custos: O que é, objetivo e processos.** Disponível em <https://escritoriodeprojetos.com.br/gerenciamento-dos-custos-do-projeto>. Acessado em 20 de março de 2022. 2020.

MONTES, Eduardo. **Gerenciamento das partes interessadas: O que é e como fazer.** Disponível em <https://escritoriodeprojetos.com.br/gerenciamento-das-partes-interessadas-do-projeto>. Acessado em 20 de março de 2022. 2021.

NASCIMENTO, Ulisses. **PMBOOK**. Disponível em <http://bradofac.blogspot.com/2014/06/pmbok.html?spref=pi>. Acessado em 24 de março de 2022. 2014.

PEREIRA, Paulo Henrique Amorin. **Testes de Segurança em Aplicacões Web.** Disponível em <https://bdm.ufpa.br:8443/jspui/bitstream/prefix/2925/1/TCC_TestesSegurancaAplicacao.pdf>. Acessado em 24 de maio de 2022. 2017.

PINHEIRO, Rafael. **9 melhores práticas para testar websites.** Disponível em <https://rockcontent.com/br/talent-blog/testar-websites/>. Acessado em 29 de março de 2022. 2021.

PMBOOK São Paulo. **7ª edição do PMBOK® Guide.** Disponível em <https://pmisp.org.br/pmbok-guide/>. Acessado em 15 de março de 2022. 2022.

PORTAL BAYMETRICS. **Introdução aos testes de segurança: um guia prático para startups.** Disponível em <https://www.baymetrics.com.br/introducao-aos-testes-de-seguranca-um-guia-pratico-para-startups/>. Acessado em 27 de março de 2022. 2022.

PORTAL CAELUM. **Apêndice - Testes de Usabilidade.** Disponível em <https://www.caelum.com.br/apostila-ux-usabilidade-mobile-web/usabilidade#gamestorming-teste-de-usabilidade>. Acessado em 25 de março de 2022. 2022.

PORTAL CIÊNCIA DA COMPUTAÇÃO. **O que foi a Crise do Software e o início da Engenharia de Software.** Disponível em <https://cienciacomputacao.com.br/tecnologia/o-que-foi-a-crise-do-software-e-o-inicio-da-engenharia-de-software/>. Acessado em 07 de março de 2022. 2015.

PRESSMAN, Roger. **Software Engineering: A Practitioner's Approach**, 6ª edição, Mc Graw Hill, 2005.

ROCHA, Camila. **Estudo da qualidade de software na Metodologia V-model e sua interação com metodologias ágeis (SCRUM).** Monografia apresentada a Faculdade de Tecnologia de São Paulo. 2011.

VOLPATO, Elisa. **Pesquisa: métricas em teste de usabilidade.** Disponível em <https://medium.com/testr/pesquisa-m%C3%A9tricas-em-teste-de-usabilidade-d64067cfe36d>. Acessado em 21 de maio de 2022. 2019.

W3C. **W3C Brasil.** Disponível em <https://www.w3c.br/pub/Materiais/PublicacoesW3C/cartilha-w3cbr-acessibilidade-web-fasciculo-I.html>. Acessado em 22 de agosto de 2021. 2021.