

**UNIVERSIDADE ESTADUAL DE GOIÁS
CAMPUS CERES
BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

ALAN VINICIUS FERREIRA DE SOUZA

**BOAS PRÁTICAS DE PROGRAMAÇÃO: TÉCNICAS E
PADRÕES PARA DESENVOLVIMENTO DE SOFTWARE**

ALAN VINICIUS FERREIRA DE SOUZA

**BOAS PRÁTICAS DE PROGRAMAÇÃO: TÉCNICAS E
PADRÕES PARA DESENVOLVIMENTO DE SOFTWARE**

O trabalho de conclusão de curso apresentado à disciplina de Trabalho de Conclusão de Curso II, como requisito parcial à conclusão do curso de Bacharelado em Sistemas de Informação, da Universidade Estadual de Goiás Campus Ceres.

Orientador: Elton Cesar Silva Moraes

ALAN VINICIUS FERREIRA DE SOUZA

**BOAS PRÁTICAS DE PROGRAMAÇÃO: TÉCNICAS E
PADRÕES PARA DESENVOLVIMENTO DE SOFTWARE**

O trabalho de conclusão de curso apresentada à disciplina de Trabalho de Conclusão de Curso II, como requisito parcial à conclusão do curso de Bacharelado em Sistemas de Informação, da Universidade Estadual de Goiás Campus Ceres.

Orientador: Elton Cesar Silva Moraes

Ceres de 2018

Banca Examinadora

Professor (a): Esp. Elton Cesar Silva Moraes/ Ceres / UEG
Orientador (a)

Professora:/ Ceres / UEG
(Graduação do professor) (orientadora)

Professora:/ Ceres / UEG
(Graduação do professor) (orientadora)

DEDICATÓRIA

Dedico esse trabalho primeiramente ao Deus que no princípio criou os céus a terra e o que nela há, por me dar inteligência e forças para ter chegado até aqui, mesmo com o universo conspirando contra mim. E em segundo lugar a todos que me apoiaram, me ajudaram e acreditaram em mim.

AGRADECIMENTOS

Agradeço ao Deus da bíblia, ao meu professor e orientador Elton Cesar Silva Morais, e a todos que me ajudaram nessa jornada.

EPÍGRAFE

“A imaginação é mais importante que a ciência,
porque a ciência é limitada, ao passo que a
imaginação abrange o mundo inteiro.”

Albert Einstein

RESUMO

Esse trabalho tem como objetivo tentar minimizar o desenvolvimento de softwares com códigos mal desenvolvidos. O foco será alertar os desenvolvedores de softwares, dos custos e consequências de um código mal programado, e apresentar boas práticas de programação com ênfase nos seus benefícios e vantagens. O trabalho foi feito utilizando como base pesquisas em livros, sites, notícias, e artigos acadêmicos. Com intuito de enfatizar a situação, foi elaborado um estudo de caso sobre testes unitários com o PHPUNIT, o que proporcionou resultados interessantes sobre códigos mal escritos. Todo o levantamento e os resultados estão dispostos no decorrer do trabalho. Essa pesquisa traz diversos métodos, para se conseguir um código limpo e de fácil entendimento com as melhores práticas de programação do mercado, que foram escolhidas, elaboradas e melhoradas pelo autor. Essa pesquisa irá auxiliar os desenvolvedores de software que estão começando no mercado, a desenvolver bons códigos e também poderá contribuir para outros artigos e trabalhos acadêmicos futuros.

Palavras-chave: Desenvolvimento, Softwares, Custos, Consequências.

ABSTRACT

This work aims to minimize the development of software with poorly developed codes. The focus will be to alert software developers of the costs and consequences of poorly programmed code, and to present good programming practices with an emphasis on its benefits and advantages. The work was done using as a base searches on books, websites, news, and academic articles. In the final part, a case study was elaborated on unit tests with PHPUNIT. At the end of the study the case presents the results of the unit tests. This research brings several methods to achieve a clean and easy to understand code with the best programming practices in the market, which were chosen, elaborated and improved by the author. This research will assist software developers who are starting out in the market, developing good codes and may also contribute to other articles and future academic work.

Keywords: Development, Software, Codes, Consequences.

LISTA DE FIGURAS

FIGURA 1CÓDIGO FONTE	49
FIGURA 2EXECUÇÃO.....	50
FIGURA 3EXEMPLO.....	50
FIGURA 4RESULTADO	51

LISTA DE ABREVIATURAS E SIGLAS

HTML - Hypertext Markup Language

CSS - Cascading Style Sheets

JS - JavaScript

W3C - World Wide Web Consortium

POO – Programação Orientada a Objetos

SEO - Search Engine Optimization

TIC – Tecnologia da Informação e Comunicação

PHP - Hypertext Preprocessor

SUMÁRIO

INTRODUÇÃO	12
1 MAIOR PROBLEMA NO DESENVOLVIMENTO DE SOFTWARE.....	14
1.1 CARACTERÍSTICAS DE UM BOM CÓDIGO	15
1.2 BOAS PRÁTICAS EM PROGRAMAÇÃO.....	17
1.3 BOAS PRÁTICAS DE PROGRAMAÇÃO COM A W3C	17
1.3.1 Benefícios de Utilizar os Padrões Oferecidos Pela W3C	18
2 DESENVOLVIMENTO WEB	19
2.1 HTML – HYPERTEXT MARKUP LANGUAGE.	19
2.1.1 Benefícios De Se Utilizar HTML5.....	20
2.2 CSS – CASCADING STYLE SHEETS	21
2.2.1 Boas Práticas de Programação com CSS.....	22
2.3 JAVASCRIPT	22
2.3.1 Boas Práticas de Programação com JavaScript.....	23
3 BOAS PRÁTICAS DE PROGRAMAÇÃO COM PADRÕES DE PROJETO	25
3.1 VANTAGENS DE SE UTILIZAR PADRÕES DE PROJETO	26
3.2 BOAS PRÁTICAS DE PROGRAMAÇÃO COM ORIENTAÇÃO A OBJETOS (POO) ...	26
3.2.1 Pilares da Programação Orientada a Objetos	27
4 CÓDIGOS MAL DESENVOLVIDOS	32
4.1 O QUE PODE CAUSAR O AUMENTO DE SOFTWARES MAL PROGRAMADOS	33
4.2 OS CUSTOS E CONSEQUÊNCIAS DE UM CÓDIGO MAL DESENVOLVIDO	34
4.3 PROBLEMAS CAUSADOS POR SCRIPTS MAL PROGRAMADOS	36
4.3.1 O Caso da Foxbit	36
4.3.2 O Caso do TCE Amazonas	36
4.3.3 O Caso da Honda.....	37
4.3.4 O Caso do Google.....	37
4.3.5 O Caso da Apple	37
4.4.O REPLANEJAMENTO DO CÓDIGO DE UM SISTEMAS	38
5 PARÂMETROS PARA MENSURAR QUALIDADE DE UM CÓDIGO	39
5.1 FERRAMENTAS PARA ANÁLISE DE CÓDIGO FONTE	39
5.1.1 A Ferramenta CheckStyle.....	40
5.1.2 Analisador de Código Fonte PMD.....	42
5.2 OUTRAS FERRAMENTAS PARA TESTAR A QUALIDADE DE UM CÓDIGO.....	43
5.3 TESTES UNITÁRIOS	45
5.3.1 A Importância dos Testes Unitários.....	46
5.3.2 Frameworks Para Automação de Testes em PHP	47
5.4 UM ESTUDO DE CASO SOBRE TESTES UNITÁRIOS COM PHPUNIT	48
5.4.1 Como Começar os Testes com PHPUnit.....	48
5.4.2 Escrevendo Testes Unitários para o PHPUnit.....	49
CONSIDERAÇÕES FINAIS	52
REFERÊNCIAS BIBLIOGRÁFICAS	53

INTRODUÇÃO

Boas práticas de programação consistem em uma série de bons padrões, que um programador deve seguir, para o desenvolvimento de algoritmos bem feitos.

A programação é uma área da tecnologia da informação e comunicação (TIC) responsável, pela escrita de um sistema em uma linguagem de programação ou de máquina, consiste em passos lógicos bem definidos com o objetivo de resolver determinado problema.

Uma linguagem de programação pode ser entendida como uma serie de métodos desenvolvidos para passar instruções a um computador, representada por um conjunto de regras sintáticas e semânticas que quando executados de forma correta geram uma ação.

Foi na década de 50 que as primeiras linguagens modernas começaram a surgir. FORTRAN (1955), LISP, a "List Processor" e COBOL, a COmmon Business Oriented Language. Também apareceu na mesma época a ALGOL 60.

O programador, analista de sistemas ou engenheiro de software é o profissional responsável pela realização da programação, ela permite que ele especifique de forma precisa a manipulação e armazenamento de dados, e também quais ações o sistema deve tomar.

Um sistema pode ser entendido como um programa de computador, que foi desenvolvido para automatizar ou resolver um problema, seu interior possui uma sequência de códigos que executam uma determinada instrução, de acordo com a interação do usuário. Ele pode ser desenvolvido tendo como base diversas linguagens de programação, pois a lógica sempre será a mesma as únicas mudanças serão na sintaxe.

Os sistemas evoluirão tanto que hoje se encontram completamente inseridos na vida dos seres humanos, e acabou se tornando parte do cotidiano das pessoas, a maioria dos indivíduos não ficam 1 hora sem estar conectados.

Com os sistemas tão presentes na vida dos seres humanos, e na grande utilização por indústrias e empresas, surgiu a necessidade de mantê-los ao longo do tempo. Por exemplo o desenvolvedor lança um sistema no mercado, quanto mais ele permanecer em utilização, mais tempo será realizado manutenção e atualizações, se o código estiver mal programado é o software não for fácil de ser mantido logo os desenvolvedores acabaram se desanimando e deixando de lado.

Antigamente era mais difícil para desenvolvedores, desenvolver e manter os softwares, pela falta de recursos tecnológicos, com isso surgiu a necessidade de se automatizar o processo de desenvolvimento e manutenção, então começou a aparecer os chamados frameworks, ferramentas para teste de código fonte, metodologias ágeis para desenvolvimento de software, e teorias para se aplicar na programação de computadores, para se conseguir códigos bons.

No decorrer desse trabalho será abordado vários tópicos que se trataram da qualidade de um código, o objetivo principal desse trabalho é passar um pouco de conhecimento aos desenvolvedores pois o mercado precisa de profissionais melhores.

1 MAIOR PROBLEMA NO DESENVOLVIMENTO DE SOFTWARE

Atualmente um dos maiores problemas no desenvolvimento de software, são os códigos mal desenvolvidos, trazendo complicações durante o processo de desenvolvimento, manutenção e utilização.

“O processo era lento, imprevisível e, em geral, nunca resultava em um produto que as pessoas queriam ou estavam dispostas a pagar para obter.” (SUTHERLAND, 2016, p. 5).

Os resultados de um sistema mal programado são: erros, falhas e defeitos. Esses três são popularmente chamados de bug, é trazem consequências como perdas financeiras, atrasos, demissões etc.

Um erro pode ser entendido como o resultado de um defeito ou falha, apresentando um retorno diferente do esperado, a falha pode ser causada no sistema se o retorno não for como esperado, e o defeito é uma imperfeição que está implementada no código fonte de maneira errada.

Esses problemas nem sempre são simples de serem resolvidos, o que demanda recursos como tempo, e mão de obra qualificada gerando altos custos, que muitas das vezes ultrapassam o orçamento, principalmente quando não houve uma preocupação, em seguir boas práticas de programação, é o código está desorganizado.

“Qualquer idiota pode escrever código que um computador entenda. Bons programadores escrevem código que humanos podem entender. ” (FOWLER, 2002, p.22).

Existem vários autores com suas obras conhecidas na área de engenharia de software, que tem como objetivo principal tentar minimizar esses problemas, como código limpo, ou codificador limpo. Esses títulos trazem boas práticas de programação, e soluções altamente eficientes para se conseguir um código de fácil desenvolvimento, e manutenção.

Mas o objetivo desses livros, não são só apresentar boas práticas de programação, mas também meios que levantam a preocupação dos desenvolvedores de software, com a qualidade dos seus códigos.

As boas práticas em programação de computadores, tem como objetivo entregar o código da melhor forma possível, é pode ser definida como um conjunto de práticas e

padrões, que foram estudados e elaborados afim de conseguir um código direto, organizado e eficiente.

Habilidade e profissionalismo é a garantia de um código entregue da melhor forma possível, organizado e eficiente, no tempo pedido.

Um programador pode adquirir habilidade e profissionalismo, de várias formas, sendo as principais praticando e estudando. A maioria das pessoas que não são da área de tecnologia geralmente se assustam, quando olham um programador escrevendo aquele tanto de linhas em inglês, com comandos que para elas não fazem sentido nenhum é até se perguntam como vocês escrevem isso?

Assim como muitas coisas na vida, como jogos de vídeo game e matemática, programação é prática e lógica, quanto mais um programador pratica, com acertos e erros mais ele aprende, na verdade ele aprende mais errando do que acertando. Assim ao longo do tempo o programador conquistará muita experiência, como raciocínio, lógica, habilidade e profissionalismo para lidar com diferentes tipos de tecnologias.

Aprender a criar códigos limpos é uma tarefa árdua e requer mais do que o simples conhecimento dos princípios e padrões. Você deve suar a camisa: praticar sozinho e ver que cometeu erros, assistir a outros praticarem e errarem, vê-los tropeçar e refazer seus passos. (MARTIN, 2011, p.8).

Além de suar camisa, conforme ressalta Martin (2011), existem algumas características importantes que devem ser seguidas com a finalidade da construção de bons códigos. Essas características serão abordadas no próximo tópico.

1.1 CARACTERÍSTICAS DE UM BOM CÓDIGO

A organização é uma das etapas que deve ser mantida, não se pode deixar de utilizar durante o processo de programação, pois ela é a base para manter toda a ordem do código, quanto maior o sistema ficar, mais organização será necessária.

A nomenclatura diz respeito as palavras que são usadas durante a programação, pois será preciso definir nomes para variáveis, classes, objetos, ou frases para comentários, por isso a nomenclatura do código deve ser bem definida, seguindo um padrão de acordo com o sistema que está sendo desenvolvido.

Supondo que um programador está trabalhando em um sistema voltado para hospitais, ele até pode nomear os nomes de variáveis, relacionados a uma empresa de engenharia por exemplo, não teria problema nenhuma, mas ficaria muito confuso no momento de realizar qualquer manutenção, afinal o sistema é para um hospital é não para uma empresa de engenharia.

A forma como o desenvolvedor estruturar e dividir o código, influencia é muito em sua organização e execução, essa técnica consiste em dividir o programa, em várias sub-rotinas¹, sendo cada uma delas responsável por executar uma sequência de ações.

Na programação de computadores, antes de escrever qualquer linha de código, um bom programador precisa parar e pensar na lógica que será utilizada, para se chegar no resultado da melhor forma possível, ao invés de sair escrevendo qualquer coisa para se chegar no resultado o mais rápido possível.

Sendo assim, um padrão de projeto é uma solução geral, para um problema que ocorre dentro do projeto de software, sendo um modelo de como resolver um problema que pode ser utilizado em situações diferentes. Garante desempenho, robustez, compreensão, reutilização, modificação e uso de códigos. Esse assunto será melhor abordado no tópico sobre padrões de projeto.

Ainda, afim de proporcionar uma melhora no desenvolvimento e na qualidade dos códigos desenvolvidos, utilizar a orientação a objetos é uma maneira rápida, organizada e de fácil entendimento, o que melhora sua utilização, o desenvolvimento e a manutenção.

Um bom código é responsável por um programa com rápidas respostas na realização de processos, executando as suas funcionalidades de forma eficiente, proporcionando uma boa interação com o usuário.

Para elaborar códigos que atendam esses princípios, as boas práticas na programação devem ser implementadas por um profissional com experiência que entenda, é saiba aplica-la. A seguir será abordado alguns pontos sobre boas práticas em programação.

¹ Sub-rotina é uma função secundária, que executa uma sequência de comandos específicos. Fonte: próprio autor 2018.

1.2 BOAS PRÁTICAS EM PROGRAMAÇÃO

Quanto mais as tecnologias evoluírem, mais é necessário sistemas bem trabalhados, rápidos, eficientes e com bom funcionamento, pois as exigências tendem a crescer cada vez mais, ou seja, os programadores que estão começando, e até os considerados sêniores, precisaram continuar estudando e evoluindo junto com as tecnologias e suas mudanças.

As vezes pode ser assustador para quem é da área de tecnologia, principalmente de programação as diversas tecnologias que um bom desenvolvedor precisa dominar, pois quanto mais tecnologias ele precisar saber mais difícil será dele se adaptar, devido as mudanças frequentes como por exemplo as atualizações.

Nas empresas de desenvolvimento web de pequeno porte, por exemplo, alguns profissionais precisam saber HTML, CSS, JAVASCRIPT, PHP, etc., no mínimo para criar um sistema web completo. Ao surgir melhorias nessas linguagens ou atualizações muitos ficam perdidos devido ao excesso de informações. Essas linguagens são melhores explanadas no capítulo 2, que trata sobre desenvolvimento web.

Para ser um bom desenvolvedor são necessárias várias habilidades, mas as duas principais são o domínio das linguagens e a consciência de programar códigos bem feitos. Estes podem ser entendidos como boas práticas na hora de programar um sistema, o que consiste em preocupar-se com a qualidade da escrita.

No desenvolvimento web, a W3C² vem ao longo dos anos desenvolvendo padrões e técnicas afim de proporcionar melhorias significativas no desenvolvimento de softwares, assim como na qualidade dos códigos implementados.

1.3 BOAS PRÁTICAS DE PROGRAMAÇÃO COM A W3C

A W3C é uma organização de padronização da Word Wide Web, é consiste em um grupo mundial com aproximadamente 400 membros, fundada com o intuito de elevar as aplicações ao seu potencial máximo. Os sistemas que seguem seus padrões e práticas

² Maiores informações sobre W3C podem ser encontradas no seu site <http://www.w3c.br/>.

podem ser melhores acessados e visualizados, pelos usuários. A W3C trata desde segurança, a boas práticas para o desenvolvimento de sistemas.

Seus padrões tendem a melhorar a usabilidade, eficiência e segurança dos sistemas, assim como também o seu o desenvolvimento.

Dentre as várias boas práticas que a W3C oferece, as principais são para: Web Designer e Aplicações, Arquitetura Web, Web Semântica, Tecnologia XML, Web de Serviços, Web de Dispositivos e Navegadores.

A W3C fornece boas práticas de programação, para as linguagens: HTML, CSS, PHP, JS, entre outras.

A organização tem por objetivo, ser uma fonte de bons padrões de desenvolvimento para as aplicações, contribuindo para melhorar cada vez mais as suas qualidades e obter melhores desempenhos na sua utilização.

1.3.1 Benefícios de Utilizar os Padrões Oferecidos Pela W3C

Um dos primeiros benefícios ao utilizar-se de padrões é a segurança, que se trata da capacidade do sistema de proteger suas informações e seus dados, impossibilitando o acesso de pessoas não autorizadas.

O segundo benefício é a performance, um sistema pode ganhá-la ao utilizar os padrões oferecidos pela W3C, pois o sistema fica mais ágil, ou seja, tem um aumento de performance na execução dos seus processos e respostas.

O terceiro benefício é a otimização para mecanismos de busca. Trata-se de um conjunto de padrões e práticas aplicados na programação, que possibilita a otimização dos sistemas web, possibilitando ganhos de performance e bons rankings orgânicos.

O quarto benefício é a acessibilidade, ela permite ao sistema ser utilizado por diversos tipos de pessoas como por exemplo: deficientes auditivos e visuais.

2 DESENVOLVIMENTO WEB

O desenvolvimento de softwares para web requer, em sua maioria equipes multiprofissionais, pois há o envolvimento de atores como os desenvolvedores, analistas, webdesingers, especialistas na área ou software que será desenvolvido, etc. (OLIVEIRA; SEABRA, 2015).

Essencialmente, quando se trata de softwares web, o HTML, CSS e JAVASCRIPT estão presentes, por serem pela ordem, os responsáveis pela estruturação das páginas web, assim como o estilo e interação.

2.1 HTML – HYPERTEXT MARKUP LANGUAGE.

Atualmente os clientes estão muito exigentes, é não querem mais uma simples página que fique online eles esperam resultados, e benefícios para os seus negócios como por exemplo aumento de visitas e vendas, é não simplesmente um sistema que está online sem gerar nenhum resultado.

Para um sistema proporcionar esses benefícios, não é necessário somente marketing e divulgação. Se a mesma foi bem desenvolvida irá atrair as visitas quase que por conta própria, bastando apenas algumas divulgações o que diminui gastos para o cliente, pois ele não precisará investir tanto em marketing.

Para que esses benefícios sejam alcançados sem gastar tanto com marketing, tem o HTML. Mas para isso o desenvolvedor precisa saber utilizar as suas TAGS, abaixo será falado mais um pouco sobre isso.

HTML é uma linguagem de marcação de hipertexto utilizada para a criação do conteúdo de páginas web de forma hierárquica, ela pode ser combinada com a linguagem de folha de estilo CSS para juntas obterem melhores resultados. Os navegadores recebem seus códigos interpretam e geram um conteúdo off-line ou online que é exibido pelo navegador atualmente a linguagem se encontra na sua quinta versão.

Para o desenvolvimento de páginas web, dinâmicas e interativas não se deve adotar o HTML mas sim o HTML5 com suas tags específicas, para determinados objetivos como padrão, para o desenvolvedor conseguir obter o máximo dos benefícios

que o Google é a própria página podem oferecer. Por exemplo rankings orgânicos, eficiência, rápidas respostas, facilidade de desenvolvimento e manutenção, esses são alguns dos vários benefícios que a linguagem oferece.

Ao utilizar HTML 5, não adianta apenas o programador declarar no topo da página `<!DOCTYPE HTML>`, especificando que a página utiliza HTML5 como padrão e não utilizar as tags específicas da linguagem de HiperTexto, ao trabalhar com ela o programador deverá saber seguir e respeitar suas regras conhecer ao máximo sobre a mesma para então programar da forma correta. O desenvolvedor deverá ter em mente que cada tag no HTML5, tem sua finalidade.

Para a criação de menus tem a tag `<menu></menu>`, para artigos `<article></article>`, para o rodapé `<footer></footer>`, parece simples, mas alguns programadores utilizam essas tags de forma errada fazendo o rodapé por exemplo com as tags `<article></article>`, o que não é o correto afinal **article** é para artigo. O uso dessas tags de forma incorreta irá prejudicar não só no processo de desenvolvimento, mas também no momento que o Sistema Web for indexado nos buscadores do Google, se as tags estiverem sendo mal utilizadas, o site irá perder inúmeros benefícios.

2.1.1 Benefícios De Se Utilizar HTML5

O HTML5 se bem utilizado pode tornar a navegação nas páginas web mais rápidas e simples, ganhando melhoras significantes na sua performance. Os programadores web podem utilizar o HTML5 para reduzir o tamanho dos arquivos que se encontram no servidor, como por exemplo, as imagens de um site, melhorando a experiência do usuário.

O HTML5 foi criado para o desenvolvimento de aplicações universais, ou seja, capazes de serem bem visualizadas em diferentes tipos de dispositivos como smartphones, tablets ou televisões.

Mais um de seus benefícios, são as técnicas de SEO (Otimização Para Sistemas de Busca) que podem ser aplicadas na hora da programação. Ela estrutura os sistemas da web, facilitando os buscadores a encontrarem um determinado site buscado, tornando a experiência de busca dos usuários na internet cada vez mais precisa.

HTML5 permite utilizar elementos de mídia para tratar imagens ou vídeos, podendo criar sistemas de chamadas de vídeo, ligações e jogos. Essas aplicações ficam mais parecidas com sistemas desenvolvidos para funcionar no sistema operacional e não no navegador, o que torna a experiência mais fácil e prazerosa.

O HTML5 permite o armazenamento de até 4Gbytes de dados estruturados pelo banco de dados presentes no navegador, no lado do cliente-side. Trabalhando de forma parecida com o uso dos cookies, mas eliminando as limitações impostas como o tamanho de 4Kb que é uma das funcionalidades mais usadas do HTML5.

Trabalhando em conjunto com HTML5, no entanto totalmente desvinculado, o CSS torna-se responsável pela estilização das páginas HTML, o que permite uma melhor interação entre elas e seus usuários.

2.2 CSS – CASCADING STYLE SHEETS

CSS é o acrônimo para o termo em inglês Cascading Style Sheets, que traduzido para o português significa Folha de Estilo em Cascata. É uma linguagem desenvolvida e mantida pela W3C, utilizada por desenvolvedores de software para criar toda a parte gráfica de um sistema web ou de qualquer aplicação que permita a sua utilização, essa linguagem ganhou muita fama ao longo dos anos sendo utilizada em milhares de aplicações, ela permite o desenvolvimento de jogos, desenhos, gráficos, e assim por diante, atualmente se encontra na sua terceira versão.

Em CSS se utiliza o caractere octothorpe (#) para se referenciar a um id, e um ponto final (.) para se referenciar uma classe.

Uma classe é uma forma de se identificar e referenciar um elemento que contém um conjunto de atributos, através dela poderá realizar a formatação dos dados, informações e imagens HTML em CSS.

Um id é uma forma de se identificar e referenciar um elemento que pode conter um conjunto de atributos ou seja, através dele também podemos realizar a formatação dos dados. Ele se diferencia da classe pois possui, a característica de um valor de HASH³ para

³ Hash é uma propriedade que define ou retorna uma string para urls. Uma url se refere ao endereço de rede no qual se encontra algum recurso informático como um site. Fonte Próprio autor.

urls único como uma impressão digital, essa característica o torna muito utilizado para o desenvolvimento de ancoras.

2.2.1 Boas Práticas de Programação com CSS

A formatação CSS deve estar na mesma ordem da referência no código HTML para facilitar no processo de manutenção. A simplicidade é a palavra-chave aqui. É de extrema importância evitar uso de seletores desnecessários. Funcionalidades parecidas devem ser agrupadas.

Evite o uso de CSS Inline, que é o uso de estilos dentro do HTML, ou seja, os dois tipos de códigos misturados, isso não é uma boa prática de programação pois a junção das duas linguagens proporciona no final um código confuso e difícil de entender o que dificulta muito o processo de manutenção.

Utilize-se de herança, pois seu principal objetivo é proporcionar ao desenvolvedor mais produtividade e gastar menos tempo com a escrita de códigos, a herança permite que os elementos filhos herdem as propriedades do elemento pai, mas não funciona com todos os elementos apenas com alguns. Não confundir com herança da orientação a objetos.

Em CSS tem os frameworks também conhecidos em CSS como pré-processadores, para utilizar durante a aplicação dos estilos. Eles aumentam a produtividade do desenvolvedor, alguns deles são conhecidos como SASS, LESS, STYLUS.

2.3 JAVASCRIPT

JavaScript é o nome pelo qual é conhecida uma linguagem de script interpretada. Ela foi desenvolvida com o intuito de facilitar e aumentar a interação do cliente com o sistema sem a necessidade dos scripts passarem pelo servidor. É considerada atualmente a principal linguagem de programação cliente-side, é considerada por muitos uma

linguagem altamente poderosa, muitos desenvolvedores acreditam que ela vai dominar a web no futuro se tornando a linguagem da sua categoria mais utilizada.

2.3.1 Boas Práticas de Programação com JavaScript

A não utilização da palavra reservada `var`, não é uma boa prática de programação pois com a não utilização da mesma a variável irá ficar definida como global, mesmo estando dentro de uma função, ou seja, será alocado um espaço de memória desnecessário, caso o programador não necessite.

Nem todas as linguagens de programação fornecem operadores que comparam se uma variável é idêntica a outra, mas a JavaScript fornece esse recurso por exemplo, vamos supor que uma condição para retornar um valor precise realizar uma comparação entre duas variáveis, e o resultado dessa comparação retornar verdadeiro, mas para retornar verdadeiro o valor das duas variáveis precisam ser idênticos, então ambas as variáveis precisariam ser do mesmo tipo primitivo e conter o mesmo valor para serem consideradas idênticas.

Essa função JavaScript e considera muito perigosa, pois fornece acesso ao compilador do JavaScript, ela diminui a performance do código JS, e risco a segurança do sistema, pois fornece muito poder a quem a utiliza.

A maioria dos scripts em JS são interpretados de forma correta pelo navegador com algumas abreviações como a ausência de ponto e vírgula ou o fechamento de um escopo com as chaves, não é indicado abreviar de mais o código pois, o mesmo pode ficar de difícil entendimento, então deverá realizar as abreviações com moderação.

JS Lint é um depurador desenvolvido por Douglas Crockford, seu script e responsável por scanear problemas e erros nos códigos desenvolvidos em JavaScript, ele retorna o problema e a sua localização.

Colocar os scripts na parte final do código HTML é muito utilizado pois, ela proporciona um carregamento mais rápido do conteúdo da página deixando a execução dos scripts JS em segundo plano para isso, basta colocar os scripts e referências a arquivos JS antes do fechamento da tag `body`.

A linguagem JavaScript permite a criação de um objeto global que recebem uma estrutura com vários tipos de dados, isso proporciona ao programador declarar várias

variáveis globais sem precisar repetir a palavra reservada `var` pois, ela é declarada como o tipo da estrutura.

3 BOAS PRÁTICAS DE PROGRAMAÇÃO COM PADRÕES DE PROJETO

Na engenharia de software usa-se o termo padrões de projeto ou design pattern para se referir a um modelo de como solucionar um determinado problema que já foi resolvido em problemas anteriores. Os padrões de projeto são orientados a objeto e trazem consigo uma ajuda aos desenvolvedores, ele possibilita a reutilização de código fonte, evitando que o programador perca tempo escrevendo códigos para recriar uma solução já existente.

Gamma et al. (2000, p. 20) definem que padrões de projeto como “descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular”, de forma semelhante.

Alur, Crupi e Malks (2004) afirmam que qualquer definição dada a padrões de projeto relaciona o padrão como uma solução para um determinado problema em um contexto específico.

Na Engenharia de Software, segundo Filho (2003), um processo terá subdivisões que possibilitem avaliar o andamento de um projeto e corrigir seus rumos quando ocorrem problemas se ele for bem definido. Diante disso, uma das medidas a serem adotadas para alcançar tal expectativa é a utilização dos padrões de projeto.

Gamma et al. (2000) descrevem os padrões de projeto com as seguintes características:

- Nome: é usado para identificar o padrão.
- Intenção: representa o propósito do padrão de projeto.
- Motivação: descreve um cenário que contém um problema que o padrão irá resolver.
- Aplicabilidade: descreve as situações nas quais os padrões podem ser aplicados.
- Estrutura: representação gráfica através de diagramas de classe e sequência para demonstrar as classes do padrão e a relação entre os objetos.
- Participantes: entidades que participam do padrão.
- Colaborações: descreve como as classes envolvidas colaboram para realizar suas tarefas.
- Consequências: avaliação dos resultados que serão obtidos com a aplicação do padrão.

- Implementação: como o padrão deve ser implementado.
- Exemplo de código: códigos que ilustram a implementação do padrão.
- Usos conhecidos: demonstração de casos de sistemas reais em que os padrões foram utilizados;
- Padrões relacionados: descreve o relacionamento existente entre os padrões.

Conforme propõem Gamma et al. (2000), os padrões de projeto são ainda classificados por dois critérios: propósito e escopo. O primeiro divide os padrões de projeto em três categorias: criacional, estrutural e comportamental. Enquanto o segundo especifica se o padrão se aplica a objetos ou classes.

3.1 VANTAGENS DE SE UTILIZAR PADRÕES DE PROJETO

Ao utilizar padrões de projeto podemos encontrar soluções já prontas para diversas tipos de obstáculos e o programador também pode reutilizar seus próprios códigos futuramente.

Os padrões de projeto proporcionam códigos limpos e eficientes, sem bugs, bem expressados e organizados. Os códigos desenvolvidos em cima de padrões de projeto são diretos e objetivos. Muitos problemas encontrados no desenvolvimento de sistemas já possuem soluções baseadas em padrões de projeto.

Os padrões de projeto proporcionam agilidade e produtividade devido a qualidade do código. Ao realizar manutenção em um sistema que utiliza padrões de projeto a qualidade do mesmo nos proporciona um ambiente de fácil manutenção.

3.2 BOAS PRÁTICAS DE PROGRAMAÇÃO COM ORIENTAÇÃO A OBJETOS (POO)

Orientação a objetos é um paradigma de programação é uma forma de se desenvolver e pensar, baseado no conceito de objetos, que possuem características e ações.

O paradigma da orientação a objetos é um processo conceitual independente da linguagem de programação [...]. O desenvolvimento baseado em objetos é fundamentalmente uma forma de pensar e não uma técnica de programação. (RUMBAUGH, 1994, p. 5).

A orientação a objetos foi originada nos anos 60 na Noruega, com Kristen Nygaard e Ole-Johan Dahl, no Centro Norueguês de Computação. Através da linguagem simula 67, foram então introduzidos os seus primeiros conceitos como objetos, métodos e herança.

Na programação orientada a objetos, os programas de computador são construídos por meio da declaração de objetos que são representados como classes, a classe pode possuir uma instancia que será um objeto que o desenvolvedor poderá utilizar por exemplo para executar um método.

As principais linguagens de orientadas a objetos, conforme disponível na Catho⁴, são C++, Java, Python, PHP e C#.

Um dos quatro pilares da orientação a objetos se chama herança, a herança é um relacionamento entre classes que permite as classes filhas que possuem algo em comum herdar os atributos e os métodos da classe pai.

A orientação a objetos traz para os desenvolvedores, outra forma de se programar totalmente diferente da estruturada, por exemplo ela permite mais organização, produtividade, facilidade e entendimento do código. Ou seja, melhora muito a vida do desenvolvedor de software.

Um software quando é bem projetado e programado, com orientação a objetos se torna um software mais confiável, pois as linguagens orientadas a objetos possuem melhores recursos. Esses recursos oferecidos pelas linguagens orientadas a objetos, se utilizados da forma correta proporcionam mais segurança.

3.2.1 Pilares da Programação Orientada a Objetos

Nessa parte será falado um pouco sobre uma das principais características da orientação a objetos chamada pilares, a palavra pilar pode ser entendida como uma coluna que sustenta algo de forma estruturada, sem essa coluna o que necessitar dela pode vir a

⁴ Catho é um site brasileiro de classificados de empregos, mais informações podem ser encontradas no seu site <https://www.catho.com.br/>.

se desestruturar, ou seja o pilar é a base para se manter uma determinada estrutura de forma estruturada.

A orientação a objetos não é diferente e possui sua base desenvolvida em cima de quatro pilares, são eles abstração, encapsulamento, herança e polimorfismo, que a sustenta para o desenvolvimento de sistemas computacionais altamente robustos e eficientes se utilizada da forma correta.

Abstração é um dos pilares mais importantes da orientação a objetos, ao programar com esse paradigma o desenvolvedor irá lidar com a representação de um objeto do mundo real no mundo virtual, então temos que pensar o que esse objeto irá realizar dentro do sistema que será desenvolvido, para abstrair as suas características e ações.

De acordo com Campos (2001, p. 103), [...] “a abstração consiste na seleção que um analista faz de alguns aspectos de um assunto em restrição a outros considerados”.

A definição apresentada por Rumbaugh et al. (1994, p. 9) diz que a abstração consiste na concentração dos aspectos essenciais próprios de uma entidade e em ignorar suas propriedades acidentais. No desenvolvimento de sistemas, isso significa concentrar-se no que um objeto é e faz antes de decidir como ele será implementado. O uso da abstração preserva a liberdade de se tomar decisões evitando tanto quanto possíveis comprometimentos prematuros com detalhes.

Em outra definição, Oliveira e Amaral (2001) afirmam que abstração é [...] um conceito no qual não se leva em conta um valor específico determinado e sim qualquer entre todos os valores possíveis daquilo com que estamos lidando ou ao que estamos nos referindo. Por exemplo, em álgebra, quando dizemos que x é uma variável, desconsideramos o seu valor atual, mas consideramos todos os possíveis valores de x como sendo números, os quais não são objetos físicos e sim objetos linguísticos, formados pela abstração durante o ato de contar.

Para deixar mais claro, quando for realizado o desenvolvimento de um sistema como, por exemplo, um sistema escolar, teriam vários objetos como: aluno, professor, diretor, funcionários e assim por diante.

Alguns aspectos que devem ser levados em consideração no momento de realizar a abstração. O primeiro é dar um nome ao objeto que será instanciado, esse nome será a sua identidade e precisa ser obrigatoriamente única dentro do sistema para que não haja problema.

Na maioria das linguagens de programação, há o conceito de namespaces, nas linguagens que possuem esse conceito, a identidade do objeto não pode ser repetida dentro do pacote, e não no projeto inteiro.

O segundo diz respeito aos atributos do objeto no mundo real, um objeto possui características que o define, no desenvolvimento orientada a objetos, essas características são chamadas de atributos por exemplo, os atributos do objeto “Pessoa” podem ser “Altura”, “Cor”, “Idade” e assim por diante.

A terceira parte é definição das ações que um objeto pode executar, essas ações, são chamadas de métodos ou funções. Esses métodos podem variar de diversas formas e irá depender do objeto pois cada objeto, possui uma ação diferente do outro, desde correr a pular que são ações de uma pessoa.

O segundo pilar é o encapsulamento. É responsável por acrescentar segurança ao sistema, pelo fato de esconder as suas propriedades.

As linguagens de programação orientadas a objetos implementam o encapsulamento baseado em propriedades privadas, ligadas a métodos especiais chamados *getters* e *setters* que irão retornar e validar o valor da propriedade. O encapsulamento evita o acesso direto aos atributos do objeto adicionando uma camada de segurança ao sistema.

O encapsulamento “consiste em evidenciar ou não as funcionalidades do objeto ao ambiente externo”. Um exemplo do motor de um carro. O motor de um carro é um objeto acionado através do método da ignição e que este método interage com outros métodos internos, como acender vela de ignição para prover a funcionalidade de ligar o carro. O método responsável por acionar a vela de ignição não está visível ao usuário do carro. Não é ele quem decide quando a vela irá acender ou não. Esse método é interno, pois a entidade usuário do carro, que interage com o carro, não interage com o motor (OLIVEIRA, 2011, p. 69).

Outro exemplo de encapsulamento seria um computador o usuário pode ver o seu funcionamento, mas algumas de suas partes físicas ficam dentro do computador, ou seja, ao utilizar o computador não sabemos o que está acontecendo internamente, mas sabemos externamente pois os métodos das funcionalidades do computador estão encapsulados.

O próximo pilar a ser abordado é a herança. Conforme definição de Rumbaugh et al. (1994, p. 4), Herança é [...] o compartilhamento de atributos e operações entre classes, com base em um relacionamento hierárquico. Uma classe pode ser definida de

forma abrangente e depois refinada em sucessivas subclasses mais definidas. Cada subclasse incorpora, ou herda, todas as propriedades de sua superclasse e acrescenta suas próprias e exclusivas características. As propriedades da superclasse não precisam ser repetidas em cada subclasse.

A reutilização de código fonte é uma das maiores vantagens do desenvolvimento orientado a objetos. Isso se dá pelo pilar que é conhecido como herança. Esse pilar é responsável por otimizar o desenvolvimento do sistema em tempo e linhas de código.

Para que seja melhor entendido sobre esse pilar, imagine uma tradicional família brasileira com pai, mãe e filhos. Os filhos herdam características de seus pais os pais herdam algo dos avós dos seus filhos, é isso faz com que as crianças também herdem essas características, e assim sucessivamente passando de geração para geração.

Na orientação a objetos, é exatamente assim, objeto na hierarquia abaixo irá herdar as características de todos os objetos acima dele, ou seja, dos seus ancestrais.

A herança dos atributos do objeto acima dele é considerada herança direta, enquanto as outras são chamadas de heranças indiretas. Por exemplo, em uma família, os filhos herdam diretamente dos pais e indiretamente dos avós e dos bisavós.

A herança irá variar de linguagem para linguagem, pois cada uma possui suas características. Na linguagem de programação C++, ele fornece a funcionalidade da herança múltipla. Que funciona da seguinte maneira o objeto herda as características de todos os seus ancestrais ao mesmo tempo, cada objeto poderá possuir quantos pais for definido.

Devido a problemas causados por conflitos entre dois ou mais membros, essa funcionalidade não é mais implementada em linguagens modernas.

E por fim, o polimorfismo, que segundo [MOTA, 2003], em Orientação a Objetos polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse.

O último pilar que será falado se chama polimorfismo, que significa “qualidade ou estado de ser capaz de assumir diferentes formas. ”, alguns animais como os camaleões podem mudar sua forma e continuar com o mesmo nome científico o polimorfismo trabalha de forma parecida, ao ler sobre herança vocês se informaram a respeito dos filhos, que herdam por herança as características dos seus pais. Um método polimórfico

é um método que se comporta de uma forma na classe pai e de outra forma na classe filha sem alterar a sua identificação.

Um exemplo de polimorfismo: Imagine um animal, todos os tipos de animais têm uma ação em comum que é emitir um som não tem? Mas cada animal emite som de uma forma diferente dos outros uns latem ou miam, outros cacarejam e assim por diante, ou seja, a classe animal pode passar o método emitir som para os seus filhos como herança que seria para as diferentes raças de animais, mas a estrutura interna desse método passado como herança para os ancestrais teria que ser alterada para executar de formas diferentes.

Se você veio direto nessa parte do trabalho é necessário que você volte um pouco mais atrás e leia um pouco sobre herança, se não vai fazer confusão e necessário entender um pouco sobre esse pilar para entender melhor o polimorfismo, cada linguagem de programação fornece funcionalidades para implementar o polimorfismo mais de forma diferente.

Enfim esses são os pilares da programação orientada a objetos que a tornaram tão poderosa e utilizada a cada dia mais, deixando para trás a programação estruturada.

4 CÓDIGOS MAL DESENVOLVIDOS

Os códigos mal desenvolvidos, podem ser entendidos como algoritmos confusos, com a lógica mal expressada ou muito grande, sem comentários e falta de organização, bugs, falhas e inúmeras consequências ruins se tornam algo frequente. Com o passar do tempo traz grandes complicações para os desenvolvedores.

Na citação abaixo, Robert C. Martin fala a respeito de um código considerado ruim.

Aliás, é como se caminhássemos penosamente por um lamaçal de arbustos emaranhados com armadilhas ocultas, isso é o que fazemos num código ruim. Peleamos para encontrar nosso caminho esperando avistar alguma dica, alguma indicação do que está acontecendo, mas tudo o que vemos é um código cada vez mais sem sentido. (MARTIN, 2011, p. 13).

Atualmente no mercado de sistemas, se encontra um grande número de softwares com problemas, é a grande maioria desses problemas são relacionados ao código fonte, que muitas das vezes estão completamente mal escritos sem seguir o mínimo de boas práticas para desenvolvimento de software.

A grande base para se ter uma noção disso e o número de reportagens que podemos encontrar, falando de erros de scripts mal inseridos, assim como códigos inseguros com brechas, para crackers ou que podem acabar causando bug mais tarde. Isso não tem acontecido somente com empresas pequenas, mas empresas grandes como Google, Microsoft, e o Facebook, já foram alertadas que seus sistemas possuem brechas por causa de alguns códigos.

Se você que está lendo isso é programador, ou analista de sistemas sabe muito bem na bomba relógio que um sistema pode se transformar e explodir a qual quer momento, se programado de qual quer maneira, os códigos mal desenvolvidos trazem consigo uma serie de vulnerabilidades.

O grande aumento dos softwares com códigos mal desenvolvidos pode ter vários motivos como por exemplo: profissionais mal capacitados, falta de tempo, baixo investimento em desenvolvimento e falta de ética profissional.

Cada um desses fatores pode influenciar de forma negativa, no desenvolvimento de um sistema, mas também existem inúmeras outras coisas que podem prejudicar a programação do sistema. Ultimamente muitos desenvolvedores não tem consciência, com

os clientes pelo fato deles muitas das vezes, não entenderem nada de programação, e acabam entregando um software sem qualidade, se importando apenas com o lucro.

4.1 O QUE PODE CAUSAR O AUMENTO DE SOFTWARES MAL PROGRAMADOS

Cada parágrafo abaixo falará um pouco sobre as causas, do que pode contribuir para o aumento de softwares mal programados. A ideia é passar um pouquinho de cada uma, começando com os profissionais que fazem tudo, mas não são especialistas em nada.

Uma das primeiras causas é o grande aumento de desenvolvedores, na área de tecnologia, pois isso contribuiu para o crescimento do número de profissionais especialistas, assim como de **full stack**, esses muita das vezes são profissionais, que fazem tudo em uma empresa de tecnologia e não são especialistas em nada, e muitas das vezes acabam fazendo tudo mais de qualquer jeito.

A segunda causa é o tempo, pois ele é uma das coisas mais preciosas que uma empresa tem, é por isso saber controla-lo muito bem é uma vantagem. Pegando como exemplo uma empresa de pequeno porte, existem menos profissionais, então cada tarefa deve ser bem organizada e executada para ser terminada em um bom tempo nem muito rápido e nem atrasada. E aí que mora o problema com um menor número de profissionais, se a demanda por desenvolvimento for grande os profissionais responsáveis pela programação, acabam tendo que programar mais rápido para ir para o próximo projeto, e acabam programando sem seguir boas práticas.

O tempo também vai ser um grande inimigo de acordo com a complexidade dos sistemas, quanto mais complexos mais difícil vai ser para implementar as funcionalidades e mais tempo vai gastar, o que acaba fazendo com que o programador resolva aquele problema complexo, mas o código não fique bom.

A terceira causa é o baixo investimento ele é mais um grande inimigo do código limpo, quanto menos investimento para o desenvolvimento do sistema, mais problemas aquele código vai ter, por que o custo da mão de obra qualificada é alta.

A quarta causa é falta de preocupação com um bom código principalmente nas empresas de desenvolvimento de software mais pequenas, muitas não se preocupam em

prezar o seu nome, bastando o código funcionar afinal o usuário só enxerga a parte gráfica.

A quinta causa é a consciência e uma das virtudes, que todo profissional precisa ter não somente na área de tecnologia, mas também em todas as áreas afinal quem produz um produto ou serviço, na maioria das vezes produz para pessoas leigas que não entendem nada do mesmo, vamos supor que você é especialista em alguma área, como você detém muito conhecimento sobre aquilo conseguiria tirar vantagens em cima de pessoas que não entendem nada, para benefícios próprios.

4.2 OS CUSTOS E CONSEQUÊNCIAS DE UM CÓDIGO MAL DESENVOLVIDO

Um software feito de forma rápida e malfeita, pode até funcionar, e satisfazer a necessidade dos clientes e dos stakeholders⁵, mas cedo ou tarde isso terá uma consequência ruim, o software poderá vir a apresentar inúmeros problemas sendo de responsabilidade de quem desenvolveu.

As consequências de um sistema mal feito podem vir de muitas formas, na citação abaixo Robert C Martin, fala a respeito das consequências de um código confuso.

Se você é programador a mais de dois ou três anos, provavelmente o código confuso de outra pessoa já fez com que você trabalhasse mais lentamente e provavelmente seu próprio código já lhe trouxe problemas. (MARTIN, 2011, p 14).

Se você trabalha em uma empresa cheia de programadores, ou já pegou um sistema para dar manutenção, com certeza já pegou diversos tipos de códigos como os considerados bons e os ruins, e com certeza o código ruim fez você perder muito tempo na hora de realizar a manutenção ou o desenvolvimento.

Ao pegar um sistema com o código todo mal feito, pode vir a acontecer de ter inúmeras confusões com os programadores envolvidos no projeto, e acabar não chegando a lugar nenhum.

⁵ O termo stakeholder, é utilizado para se referir a todos os envolvidos em um projeto de software diretamente ou indiretamente, por exemplo: clientes, programadores etc. Fonte: próprio autor.

As perdas financeiras são um dos maiores problemas, de qual quer empresário, ao desenvolver um sistema e lançá-lo no mercado ele irá se comporta de uma forma diferente, pois muitas coisas mudam como por exemplo o número de acessos por usuários pode crescer muito e se o software não estiver bem programado irá começar a ter bugs, travamentos e lag então os usuários poderão se frustrar e nem abrir mais aplicação, se a empresa depender dos acessos para se manter financeiramente logo começa a ter sérios problemas.

Os responsáveis por causar problemas, com os scripts mal programados acabaram sendo demitidos, por justa causa e substituídos por programadores mais qualificados.

Um código mal programado pode ser sim, um dos principais motivos para a falência de uma empresa, dependendo do que estiver acontecendo dentro da empresa o caos será instaurado e logo, logo a mesma ira fechar as portas.

Quanto mais problemas um software vier a ter, mais clientes a empresa irá perder, a final os clientes não iram pagar por um sistema cheio de problemas e eventos inesperados.

A contratação desnecessária de funcionários, pode vir a ocorrer em uma empresa que está com atraso em seus projetos, mas nem sempre e o melhor caminho pois os novos profissionais não estarão adaptados, para trabalhar nos sistemas que já estão sendo desenvolvidos, em vez da empresa ir logo contratando novos profissionais ela deve rever o que está causando o atraso dos projetos e melhorar isso.

A desmotivação, e outro fator que pode ser causado por códigos confusos, imagina só pegar um código de um sistema enorme, sem nenhuma linha de comentário sem nenhuma organização, para dar manutenção qual quer funcionário se sentiria desmotivado, pois saberia da dor de cabeça que viria a ter.

Um código desorganizado, sem o mínimo de comentários traz uma dificuldade muito grande em qual quer processo, de desenvolvimento ou manutenção, exigindo muito mais esforço para se entender o que está acontecendo em todas aquelas linhas de código.

O que mais pode frustrar um programador, e mexer em um código confuso que uma parte de nada, acaba afetando uma parte que está funcionando do sistema, e isso acabar se tornando uma bola de neve.

4.3 PROBLEMAS CAUSADOS POR SCRIPTS MAL PROGRAMADOS

Abaixo será abordado vários tópicos, que se tratam de passar aos leitores alguns problemas, que empresas reconhecidas nacionalmente e internacionalmente tiveram devido a scripts mal programados, esses tópicos tem como objetivo tentar alertar não só os desenvolvedores de softwares, mas também os usuários, das consequências que podem ser geradas.

4.3.1 O Caso da Foxbit

Uma das maiores corretoras de Bitcoin do Brasil, Foxbit, enfrentou problemas no seu código. Scripts (sequência de comandos), mal inseridos provocaram um bug, o que permitiu que usuários da plataforma realizassem saques duplicados, ou seja se um usuário realizasse o saque de 1 bitcoin, o valor final que ele receberia seria de 2 bitcoins, independente do usuário ter o saldo em conta. Essa falha acarretou em um prejuízo de aproximadamente 1 milhão de reais para a empresa.

O próximo tópico irá despertar a preocupação do leitor, toda vez que ele for executar um script no banco de dados.

4.3.2 O Caso do TCE Amazonas

Os processos eletrônicos do tribunal de Contas do Estado do Amazonas (TCE-AM), foram afetados pela pane nos seus sistemas de e-Contas e Spede (Sistema de Processos e Documentos Eletrônicos). Scripts, mal inseridos pelos próprios funcionários, no banco de dados postgresSQL, causou a exclusão de 16,5 mil processos do seu banco de dados.

No próximo tópico o leitor se informa a respeito do quanto é importante um sistema embarcado bem programado, é que as boas práticas de programação englobam tudo que se utiliza códigos desde um simples web site a um automóvel.

4.3.3 O Caso da Honda

Por causa de um defeito de programação no sistema dos carros, a Honda teve que realizar um recall de mais de 2 milhões de automóveis. O fato, ocorrido em 2011, custou alguns milhões de dólares à empresa japonesa. O airbag era ativado com muita força e pelo componente errado.

Nos dois próximos tópicos o leitor fica ciente, que nem as maiores empresas de tecnologia do mundo escapam das consequências de um código mal feito.

4.3.4 O Caso do Google

Em 2009, uma barra invertida adicionada por um programador às URLs que eram direcionadas para o buscador do Google provocou a identificação do site como sendo um malware no mundo inteiro por cerca de 1 hora. Prejuízo de quase US\$ 3 milhões.

4.3.5 O Caso da Apple

Após a atualização do sistema operacional High Sierra, um desenvolvedor turco descobriu uma falha grave que afetava a área de autenticação de usuários dos computadores da marca: ao digitar a palavra *root* no campo de login e pressionar Enter por algumas vezes, o computador era desbloqueado sem a necessidade de senha, ficando vulnerável a qualquer pessoa.

4.4.O REPLANEJAMENTO DO CÓDIGO DE UM SISTEMAS

“No final, a equipe se rebela. Todos informam à gerência que não conseguem mais trabalhar neste irritante código-fonte e exigem um replanejamento do projeto. ” (MARTIN, 2011, p.15).

O replanejamento de parte ou de todo o código de um sistema, pode ser um dos maiores tormentos de quem produz software, por que quanto maior o sistema mais dificuldades ele poderá gerar. Ao resolver um problema pode ser que apareça outros dez, se aqueles programadores que trabalharam no projeto não estão mais na empresa será que os novos programadores iram entender o código do sistema, os desenvolvedores então iram se fazer várias perguntas como por exemplo, devemos refazer todos os códigos do início? Ou devemos refazer apenas parte dos códigos? Quanto isso irá custar? Será mesmo viável?

Talvez os códigos estejam tão confusos que seja melhor desenvolver outro do início o replanejamento irá demandar mão de obra, custos financeiros e tempo. E se a empresa não estiver preparada para arcar com isso, se isso aconteceu com um sistema pode acontecer com vários.

Então a equipe irá se reunir em uma sala, e começar a discutir e se fazer várias perguntas sobre o replanejamento do código do sistema, nessa discussão serão decididas as medidas que a equipe de desenvolvimento irá tomar para conseguir lidar com esses códigos problemáticos.

Ao começar o replanejamento, a empresa poderá por exemplo se dividir em duas equipes diferentes de trabalho, uma para realizar a manutenção nos códigos problemáticos, e outra para iniciá-lo do zero, com novos profissionais mais capacitados, então se iniciara uma corrida para concluir o software.

Enquanto uma das equipes, decide o que fazer com o código do sistema problemático a outra equipe irá desenvolver um software do zero, com as mesmas funcionalidades do antigo, seguindo à risca boas práticas de programação para não ocorrer os mesmos problemas do software antigo, o que poderá levar bastante tempo.

Se você ainda não viu uma cena assim, quando chegar a ver terá a certeza do quanto é importante limpar bem seu código, mesmo levando mais tempo e o custo aumentar um pouco, essa cena pode durar meses ou até anos.

5 PARÂMETROS PARA MENSURAR QUALIDADE DE UM CÓDIGO

O que é qualidade? A qualidade pode ser definida como um atributo que designa uma característica boa a algo, como um produto, software ou código.

Entre os aspectos avaliados para medir a qualidade de código incluem-se (DARWIN, 2007): formatação consistente e facilidade de entendimento (indentação e espaçamento); regras de nomeação consistentes; ausência de erros de compilação; capacidade adequada e consistente de tratamento de erros de execução; aderência a boas práticas de programação e de projeto; documentação abrangente e de fácil entendimento em todo o código.

Algumas das características apresentadas podem ser consideradas fáceis de implementar, mas não todas. Muitas empresas e organizações possuem padrões documentados que os desenvolvedores devem seguir. Na prática, provou-se ser muito difícil para essas organizações obter sucesso nessa padronização (MYATT, 2008).

Para definir se a qualidade de um código está boa ou ruim o desenvolvedor deve se inteirar sobre as boas práticas de programação e então analisar frequentemente e rever se o código está seguindo os bons padrões para desenvolvimento de software. De acordo com sua revisão o programador irá chegar a uma conclusão a respeito da qualidade do código.

O processo de analisar o código irá gastar tempo e será uma tarefa árdua e difícil de se conduzir, é a partir daí que surge a necessidade de automatizar essa tarefa caso ela esteja sendo executada de forma manual, através de sistemas capazes de realizar o processo de avaliação da qualidade do código, o que tornará isso ainda mais produtivo e integrar essas ferramentas a uma IDE (interface de desenvolvimento integrado).

5.1 FERRAMENTAS PARA ANÁLISE DE CÓDIGO FONTE

Com o avanço da tecnologia e dos sistemas computacionais, surgiu a necessidade de se desenvolver programas, para analisar o código de outros programas e gerar relatórios com dados e informações a respeito desses códigos. Assim o

desenvolvedor pode ter conhecimento de como o código está se comportando quando executado, se está com falhas, bugs ou gerando consequências ruins para o software, cada sistema que trabalha com análise de código fonte possui um objetivo específico.

Dentre as possibilidades há a ferramenta CheckStyle, o PMD, dentre outras. A seguir será abordado um pouco mais sobre essas duas ferramentas.

5.1.1 A Ferramenta CheckStyle

A ferramenta CheckStyle é gravada em java, projetada para auxiliar os desenvolvedores de software a analisar a qualidade dos códigos desenvolvidos. Ela, atualmente, se encontra disponível na sua versão 8.10 e para usá-lo, basta acoplá-lo como parte de um script ou como plug-in para IDEs como NetBeans e Eclipse.

Checkstyle disponibiliza diversos controles (“checks”), um para cada padrão de codificação ou estilo capaz de identificar. É totalmente configurável e seus controles podem ser habilitados para avaliar o código como desejado.

A partir do momento que a avaliação é feita, é gerado relatório, a respeito do código, onde podem incluir avisos e erros.

O Checkstyle ajuda a identificar por exemplo linhas duplicadas, o que inclui partes em que o desenvolvedor empregou o atalho de copiar e colar para reproduzir blocos de código semelhantes. Esses blocos podem gerar duplicação, redundância, falhas, bugs e ainda dificultar o entendimento do que está escrito.

Um dos controles contém uma funcionalidade que especifica o número de linhas para que um bloco de código seja considerado repetido. No arquivo de configuração, se dois blocos de código com três linhas iguais forem detectados em qualquer parte do código fonte, serão considerados duplicados, mesmo que a indentação esteja diferente.

A duplicidade só não será detectada se os blocos de códigos forem funcionalmente idênticos, mas usarem nomes de variáveis diferentes. Segundo (BURN, 2008), CheckStyle procura manter uma relação de compromisso entre rapidez, baixo uso de memória, baixa ocorrência de alarmes falsos e implementação de recursos avançados de pesquisa.

Atualmente no mercado de sistemas se encontram disponíveis ferramentas comerciais para detecção de códigos duplicado capazes de obter melhores resultados.

Uma vez identificado uma duplicação de código, o desenvolvedor deve examinar e tomar as medidas de acordo com sua consciência.

O CheckStyle também disponibiliza uma funcionalidade considerada a mais simples e importante disponibilizada, ela permite a verificação de todo o código e identifica comandos de importação que não estão sendo necessários no programa ou estão sendo duplicados.

Imagine que várias classes de uma biblioteca são importadas para o projeto, tornando-se necessário manter as referências ao seu chamado, mesmo não sendo mais necessária à sua utilização, então esses imports continuaram sendo referenciados, sem uma utilização.

O NetBeans, pode detectar imports não utilizados em um código fonte aberto a partir da funcionalidade de corrigir importações. Mas esse processo é difícil de ser colocado em prática em grandes projetos de software, compostos de muitos arquivos de código fonte.

Então o CheckStyle pode identificar onde existem imports não utilizados em todo o projeto. O desenvolvedor terá apenas que abrir e corrigir os códigos fontes detectados. Deve se destacar também que imports não utilizados devem ser removidos antes de gerar o executável da aplicação. Se o chamado a biblioteca estiver incorreto ou ausente ocorrerá erro na build, o que pode demorar para ser resolvido.

O Checkstyle, também tem a capacidade de identificar a ocorrência de múltiplas variáveis declaradas e inicializadas em uma única linha de código. Códigos escritos dessa forma são de difícil entendimento e documentação. O arquivo de configuração do Checkstyle inclui um controle para tratar esse tipo de ocorrência.

Todos os controles do Checkstyle são declarados em um arquivo XML em que uma tag module é definida para cada exame habilitado. O arquivo de configuração pode ser executado a partir de uma linha de comando, como parte de um script ou a partir de um plug-in.

O CheckStyle proporciona que o processo verificação de qualidade de código torna-se mais produtivo quando incorporado a um ambiente de desenvolvimento de software. Existem diversos plug-ins para integrar os recursos do Checkstyle a IDEs utilizados pelos engenheiros de software, como o Eclipse.

5.1.2 Analisador de Código Fonte PMD

PMD (COPELAND et al., 2008) é uma ferramenta de análise estática projetada para a identificação de violações de código e bugs. Enquanto Checkstyle é mais focado em estilos e padrões de codificação, PMD está mais relacionado a bugs e à otimização do código fonte, além de uma série de outros problemas detectáveis em programação.

O desenvolvimento da ferramenta PMD teve início a partir dos trabalhos de Dixon-Peugh e Copeland e se encontra atualmente na versão 6.8.0, disponível para uso sob licença BSD. Segundo os autores, PMD não se trata de nenhum acrônimo e, portanto, não há um nome por extenso aplicável ao software (COPELAND, 2005).

PMD pode ser usado a partir de linha de comando do sistema operacional, como um build ou integrado as principais IDEs disponíveis, como Eclipse e Netbeans e também em ambientes voltados ao meio acadêmico, como o BlueJ.

Assim como Checkstyle, PMD também possui uma série de controles, cada um relacionado a um tópico específico que a ferramenta é capaz de avaliar. É possível configurar o PMD usando um ou mais controles e avaliar a partir daí o código fonte. O PMD irá gerar um relatório contendo as violações detectadas, o desenvolvedor Java poderá então examinar o código e fazer as alterações que achar necessária.

Entre os principais problemas de codificação avaliados pelo PMD destacam-se (COPELAND et al., 2008): – possíveis bugs, por exemplo, try/catch vazios ou switch sem breaks; – dead code, por exemplo, variáveis locais ou métodos não referenciados; – suboptimal code, por exemplo, no uso inadequado de classes String e StringBuffer; – código ou expressões complexas, por exemplo, em if's desnecessários, laços for em lugar de laços while mais simples; – código duplicado.

A versão 4.2.3 do PMD possui centenas de controles (checks) para verificação da qualidade do código fonte, classificados em diversas categorias, como básicos, design, strings e Junit tests. Todos os controles do PMD devem ser declarados em um arquivo xml, em que uma tag rule ref é definida para cada check habilitado, esses controles foram melhorados cada vez mais com o passar do tempo.

O arquivo de configuração pode ser executado a partir de uma linha de comando, como parte de um script ou a partir de um plug-in para as principais IDEs disponíveis, como Eclipse e Netbeans.

O PMD pode identificar comandos switch sem cláusulas break. O arquivo de configuração apresentado na ilustração 8 inclui um controle MissingBreakInSwitch. O exemplo de código fonte da ilustração 9 contém um comando switch sem as cláusulas break correspondentes a cada case e o controle específico incluído no arquivo cesjf.xml permitirá ao PMD detectar a violação de código.

Esse controle do PMD é capaz de identificar inserções inadequadas em objetos das classes como String. Muitas vezes os programadores escrevem trechos de programas com concatenações repetidas em strings, principalmente em comandos de laço, o que gera um código ineficiente, sujeito à degradação na performance e inadequação no uso da memória.

A concatenação a partir de StringBuffer é mais eficiente porque os objetos dessa classe não são imutáveis, ao contrário dos objetos da classe String.

O problema reside no fato de que, a cada nova concatenação com strings, a JVM gera um novo objeto na memória para executar a operação (HUTCHERSON, 2000).

Após a determinação pela equipe de desenvolvimento de todos os controles PMD a serem utilizados no projeto, será criado um arquivo XML de configuração.

O objetivo do trabalho de verificação do código fonte com a ferramenta PMD a partir do arquivo de configuração produzido é compelir a equipe de desenvolvimento a seguir boas práticas de codificação em Java definidas para o projeto (DOEDERLEIN, 2006).

O arquivo de configuração pode ser executado via linha de comando, através de um script ou a partir de um plug-in para algum IDE.

5.2 OUTRAS FERRAMENTAS PARA TESTAR A QUALIDADE DE UM CÓDIGO

Existem várias outras ferramentas disponíveis no mercado para a análise da qualidade de código-fonte como por exemplo: FindBugs, projetado e desenvolvido pela Universidade de Maryland, EUA, e distribuído sob licença LGPL, atualmente ele se encontra na versão 3.0.1. Também tem o UCDetector, lint, sonar, JustCode, FxCop, NDepend.

O FindBugs é um analisador de código estático de código aberto criado por Bill Pugh e David Hovemeyer, que detecta possíveis erros em programas Java. O FindBugs pode ser usado como uma aplicação Java, seja a partir da linha de comando do sistema operacional ou de uma interface gráfica baseada em componentes Swing, como parte de um script ou integrado a ambientes de desenvolvimento como Netbeans e Eclipse (PUGH et al., 2008).

Os erros examinados pelo programa podem ser classificados em quatro categorias, são elas: Muito grave I; Grave II; Muito preocupante III; e Preocupante IV. Esse programa está disponível para as seguintes IDEs: Eclipse; NetBeans; IntelliJ IDEA; Gradle; Hudson; Maven; Bamboo; e Jenkins.

Outra ferramenta é o Lint, que é uma ferramenta para análise de código fonte ela pode ser integrada ao Android Studio, ela foi introduzida na versão 16 do ADT, ela trabalha em busca de bugs e possibilidades de otimização no projeto, ela tem como principal objetivo proporcionar um código conciso, seguro e com alta performance.

Para testar o Lint basta executar um check e olhar o relatório que irá gerar sobre o projeto.

O PageSpeed⁶ Insights conhecido também pelo acrônimo PSI é uma ferramenta web disponibilizada e mantida pelo Google e pode ser acessada por meio de um navegador com acesso à internet.

A ferramenta recebe como parâmetro o endereço de um sistema web qualquer que será examinado, então o sistema gera um relatório de como o sistema está se comportando na web, como por exemplo se está lento, no final da análise se a página estiver lenta o sistema disponibiliza uma série de boas práticas para melhorar o desempenho do sistema.

Para gerar o relatório o sistema utiliza como parâmetro diversas condições como, por exemplo, o CrUX.

O CrUX é um relatório de experiência do usuário do Chrome, para exibir dados e informações do desempenho real de uma página, então o PSI informa para ele duas métricas a FCP relacionada a exibição de conteúdo, e a DCL relacionada ao carregamento do conteúdo.

O valor da mediana de cada métrica (FCP ou DCL) é comparado a todas as páginas monitoradas pelo relatório CrUX. Cada métrica recebe uma categoria,

⁶ Para acessar o PageSpeed utilize o link:
<https://developers.google.com/speed/pagespeed/insights/?hl=pt-br>

dependendo de onde ela se encaixa na distribuição. Rápido – caso o valor da mediana da métrica está no terço mais rápido de todos os carregamentos de página; lento – quando o valor da mediana da métrica está no terço mais lento de todos os carregamentos de página; e por fim, médio, quando o valor da mediana da métrica está no terço médio de todos os carregamentos de página. A pontuação geral de velocidade é calculada com base nas categorias de cada métrica.

5.3 TESTES UNITÁRIOS

Como os testes unitários são criados e executados? Eles são criados e executados a partir de classes de teste e métodos, por programadores uma de suas principais características e que são automatizados.

Testar um sistema significa verificar se a sua execução, está de acordo com o que foi especificado, para determinar se ele irá se comportar como esperado no ambiente para o qual foi desenvolvido. O profissional que é responsável por essa tarefa é chamado testador. Ao testar ele tem a difícil missão de garantir a qualidade do software antes que o mesmo vá para o usuário final.

Para executar essa tarefa com êxito, o testador sênior precisa assumir um papel totalmente contrário à de um programador. Enquanto o programador tenta fazer seu software em desenvolvimento funcionar perfeitamente, o testador possui um papel contrário, tentando, de todas as formas, provar para o programador que o sistema possui falhas. Por esse motivo, não se recomendam que os testes sejam feitos por programadores.

Segundo Barry Boehm[5], um defeito encontrado no desenvolvimento do design e requisitos da solução custa 10 vezes menos que um encontrado na codificação em si e 100 vezes menos do que um encontrado na entrega do produto. Portanto, o quanto antes os desenvolvedores e os testadores começarem a testar o produto, menos custoso o projeto será e terá, definitivamente, muito mais qualidade.

Testes unitários, são os testes das menores partes de um código que são funções. Na orientação a objetos podemos realizar o teste do método de um objeto. O teste unitário é realizado por partes e não tem como objetivo realizar o teste de um sistema como um todo.

Imagine uma função que multiplica dois parâmetros e retorna o seu resultado, podemos escrever um teste unitário para testar essa função.

O que de fato são testes unitários? São métodos que testam partes dos pequenos blocos de código do sistema. Esses métodos de testes podem ser criados com o auxílio de um framework de testes e então utilizados via linha de comando ou por um serviço de build automatizado.

Após os testes serem realizados, o programador recebe um feedback de quais passaram e quais falharam, as falhas geralmente são erros no código.

Os testes unitários são baseados em dois pilares Extreme Programming e Test Driven Development.

Extreme Programming: Metodologia ágil de desenvolvimento de software com foco na agilidade da equipe e da qualidade dos projetos suas características são simplicidade, comunicação e feedback.

Test Driven Development: Desenvolvimento dirigido por testes, define que antes de ser criado um código deve se escrever um teste.

5.3.1 A Importância dos Testes Unitários

Os testes unitários são muito importantes na programação, de computadores pois eles automatizam, os testes do sistema, auxiliando o profissional responsável na hora de se encontrar falhas.

Logs de erros frequentes no funcionamento, este índice faz com que, muitas vezes, o software deixe de ser utilizado.

Quando executado, o produto pode até ter um bom desempenho, entretanto, ao longo do tempo, ele pode apresentar complicações que exigem a substituição de todo o sistema.

Dentre as principais vantagens dos Testes Unitários, pode-se citar: Incentivo a refatoração; incentivo a evolução da arquitetura do sistema; Simplificação e agilidade na entrega do produto aos usuários; fazer a documentação do código; previne regressão de bugs; e Substituir testes manuais.

A utilização de Testes Unitários previne bugs, tornando os códigos mais confiáveis, uma vez que os testes promovem situações de sucesso e falhas.

5.3.2 Frameworks Para Automação de Testes em PHP

A definição mais simples para framework é a união de códigos em comuns entre projetos de softwares promovendo funcionalidades genéricas. Assim, os frameworks para testes reúnem funcionalidades que permitem realizar diversos tipos de testes, conforme suas especificações.

Essencialmente para PHP, existem diversos frameworks para testes, no entanto aqui será abordado apenas alguns muito rapidamente, tais como o PHPUNIT, CODECEPTION e BEHAT.

O CODECEPTION⁷ é um framework de testes automatizados para sistemas web, possui um conjunto flexível de funcionalidades, seus testes são fáceis de usar e de se manter. De acordo com a documentação encontrada em seu site, ele é pode ser criado no topo do PHPUnit. Ainda fornece algumas ferramentas bem construídas para tornar seus testes unitários mais simples e limpos.

BEHAT⁸ é um framework de teste para desenvolvimento orientado a comportamento gravado na linguagem de programação PHP. O Behat foi criado por Konstantin Kudryashov e seu código está hospedado no GitHub⁹, atualmente se encontra na sua 3.3.0 versão.

O PHPUNIT¹⁰ é um framework de testes unitários, para a linguagem de programação PHP, criado por Sebastian Bergmann, tendo sua versão inicial lançada em 15 de março de 2004. O framework faz parte da família XUnit, iniciada com o SUnit, e similar ao JUnit. O código do PHPUnit está hospedado no GitHub, atualmente o framework se encontra na sua 7.1.3 versão.

É possível encontrar uma documentação em português em seu site oficial. Na verdade, essa documentação é um manual com o passo a passo para realização dos testes. Nele há informações de como realizar a instalação, como escrever os testes, etc.

No próximo capítulo será apresentado um estudo de casos utilizando o PHPUNIT para analisar situações – códigos, afim esclarecer melhor o uso de testes, assim como os impactos causados por códigos mal escritos.

⁹ GitHub é uma plataforma de hospedagem de código-fonte, mais informações podem ser encontradas em <https://github.com/>

5.4 UM ESTUDO DE CASO SOBRE TESTES UNITÁRIOS COM PHPUNIT

Desde o nascimento dos sistemas computacionais, surgiu a necessidade de testa-los, para comprovar se eles estão funcionando da forma como devem funcionar. Antigamente esses testes eram feitos manualmente por algum profissional responsável pelo desenvolvimento, mas com o aumento da complexidade dos softwares, e com o passar do tempo surgiu a necessidade, de se realizar testes automatizados, pois eles aumentam a produtividade, e garantem a integridade daquilo que foi testado, informado se determinado método está funcionando como especificado ou não.

Como já foi mencionado no capítulo anterior, aqui será abordado um estudo simplificado utilizando o framework PHPUNIT. Nesse estudo será abordado e mostrado os resultados gerados, por meio da execução de testes unitários, alguns iram passar e outros falhar, esses testes serão executados por meio de classes de teste.

5.4.1 Como Começar os Testes com PHPUnit

Para uma pessoa leiga que deseja realizar testes unitários por meio do PHPUnit a primeira coisa a se fazer e entrar no site oficial disponível no seguinte endereço eletrônico < <https://phpunit.de/index.html> >, a segunda e se informar a respeito do mesmo e entender a sua documentação.

Depois de se informar um pouco a respeito do framework, siga as etapas abaixo, o exemplo que será mostrado será através do prompt de comando do Windows e do composer.

Crie um novo projeto, chamado testesUnitarios, dentro dele deixe da forma como desejar as classes que serão testadas, abra o CMD, em seguida vá até o diretório que deseja instalar o PHPUNIT, e escreva o seguinte comando: `composer require phpunit/phpunit` então o processo de instalação do PHPUNIT irá começar será gerado vários arquivos.

Para a realização dos testes você precisa ir apenas até o diretório do seu projeto por meio do CMD, dentro do diretório você já estará apto a realizar os testes unitários com os comandos do PHPUNIT, para testar você pode escrever por exemplo `phpunit --`

version, ele irá retornar a versão do PHPUNIT que foi instalada, não se esqueça de configurar as variáveis de ambiente para poder realizar os testes dos projetos em qualquer diretório, se as variáveis de ambientes não forem corretamente configuradas os testes deveram ser realizados dentro da pasta vendor\bin que fica dentro do diretório do PHPUNIT.

5.4.2 Escrevendo Testes Unitários para o PHPUNIT

Crie um novo arquivo de código fonte PHP no seu local de costume, o arquivo deve conter obrigatoriamente uma referência à classe TestCase, essa referência pode ser a seguinte **use PHPUnit\Framework\TestCase** mas pode vir a mudar.

Depois disso crie uma classe dentro do arquivo, ela deve obrigatoriamente estender a classe TestCase, agora você pode desenvolver todo o código da sua classe que será testada.

Ex:

```
use PHPUnit\Framework\TestCase;
class Calcular extends TestCase{
    public function calcula (){
        // código fonte
    }
}
```

Depois que a sua classe de teste estiver pronta abra o CMD(prompt de comando do Windows) e abra o diretório onde está a classe de teste, então utilize o seguinte comando para iniciar o teste phpunit nomedaclasse.php e tecle enter então o framework irá começar a realizar o teste e retornar o relatório.

Agora será mostrado um teste unitário em uma classe que realiza o cálculo de dois números, em uma parte do código precisamos declarar método de asserção assertEquals, ele funciona da seguinte maneira recebe dois argumentos **\$expected** e **\$actual** é verifica se os argumentos passados batem, vamos supor que resultado recebe o valor de uma soma, então o primeiro argumento do método assertEquals é 3, e o segundo resultado se os dois não baterem o teste irá falhar, esse é só um de vários métodos que o framework oferece.

Figura 1 Código Fonte

```
<?php
use PHPUnit\Framework\TestCase;
class Calcular extends TestCase {
```

```

public $resultado,$n1 = 1,$n2 = 1;

public function soma (){
    $this->resultado = $this->n1 + $this->n2;
    $this->assertEquals(3,($this->resultado));
}
?>

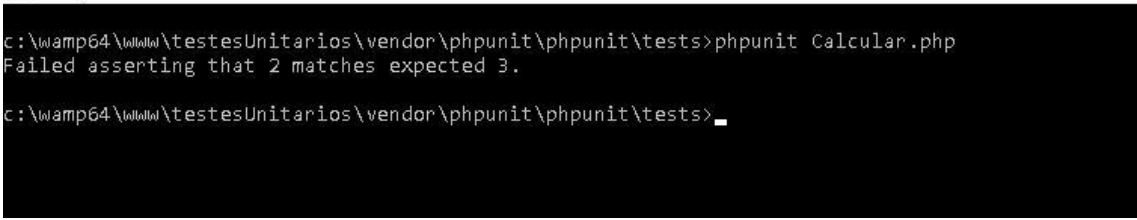
```

Próprio autor, 2018

11

Qual o resultado da execução acima? Uma falha como explicado acima.

Figura 2 Execução



```

c:\wamp64\www\testesUnitarios\vendor\phpunit\phpunit\tests>phpunit Calculador.php
Failed asserting that 2 matches expected 3.

c:\wamp64\www\testesUnitarios\vendor\phpunit\phpunit\tests>

```

Próprio autor, 2018

Agora será mostrado o exemplo de um teste que irá passar.

Figura 3 Exemplo

```

<?php
use PHPUnit\Framework\TestCase;
class TesteTCC extends TestCase {

    public $nome;

    public function TesteDoArray(){
        $this->nome = array(
            'alan'
        );

        if (!strcmp($this->nome,"alan")){
            $r = 10;
        }
    }
}

```

¹¹ expected e actual são apenas parâmetros que o método assertEquals espera receber.

```

    }
    $this->assertEquals(10,($r));
    }
}
?>

```

Próprio autor, 2018

O array declarado recebe um nome, ou seja, não está vazio mais em baixo o if recebe como argumento a negação, do array comparado com o mesmo nome que está dentro do array, então a condição é satisfeita e a variável \$r recebe o valor 10, ao receber os dois valores passados como argumento para o objeto assertEquals batem e o teste passa.

Figura 4 Resultado



```

C:\wamp64\www\testesUnitarios\vendor\phpunit\phpunit\tests>
w                                                                    1 / 1 (100%)
Time: 218 ms, Memory: 4.00MB
There was 1 warning:
1) Warning
No tests found in class "TesteTCC".
WARNINGS!
Tests: 1, Assertions: 0, Warnings: 1.
c:\wamp64\www\testesUnitarios\vendor\phpunit\phpunit\tests>

```

Próprio autor, 2018

CONSIDERAÇÕES FINAIS

Ao ler a pesquisa com atenção o leitor, será conscientizado a respeito do quanto é importante o desenvolvimento de um bom código fonte, e irá ter uma noção de como melhorar o desenvolvimento dos seus algoritmos, seja em uma linguagem ou em várias, o trabalho informa ao leitor ferramentas que ele pode utilizar para automatizar o desenvolvimento de um código limpo e organizado.

A pesquisa disponibiliza para o leitor as diversas tecnologias, para tratar os códigos programados como por exemplo o PHPUNIT, usar um sistema para tratar os códigos desenvolvidos pode acabar garantindo o sucesso ou fracasso do sistema que será desenvolvido.

Também foi falado da importância de se conseguir um código bem feito, linha a linha, com todas as técnicas e padrões seguidos não só desse material, mas de diversos outros, então o software estará mais que apto para ser finalizado e entregue.

O objetivo dessa pesquisa foi contribuir para dar um ponto de partida aos desenvolvedores que buscam se especializar na programação de computadores, pois o primeiro passo para se tornar um especialista é programar códigos bons e não códigos totalmente desorganizados. Nesse trabalho foi dada apenas a ponta do iceberg, se você realmente quer melhorar na programação, busque mais conhecimento e pratique.

REFERÊNCIAS BIBLIOGRÁFICAS

ADMINISTRADOR, SEMÂNTICA, PERFORMANCE E BOAS PRÁTICAS NO CÓDIGO FRONT-END.tecmedia, 18 fev.2015.Disponível em: < <http://tecmedia.com.br/semantica-performance-e-boas-praticas-no-codigo-front-end/> >. Acesso em: 02 setem.2018.

OLIVEIRA, Fernando Gonçalves de; SEABRA, João Manuel Pimentel. **METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE: UMA ANÁLISE NO DESENVOLVIMENTO DE SISTEMAS NA WEB.** Periódico Científico: Tecnologias em Projeção, Distrito Federal, v. 1, n. 9, p.20-34, jun. 2015. Disponível em: < <http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/view/497/463> >. Acesso em: 24 nov. 2018.

AFP, Google recorre de multa recorde de €4,3 bi por causa de sistema Android. EXAME, 9 out.2018. Disponível em: < <https://exame.abril.com.br/tecnologia/google-recorre-de-multa-recorde-de-e43-bi-por-causa-de-sistema-android/> >. Acesso em: 10 out.2018.

BBC, Perfis do Facebook são hackeados e têm mensagens privadas colocadas à venda.G1, 2 nov.2018.Disponível em: < <https://g1.globo.com/economia/tecnologia/noticia/2018/11/02/perfis-do-facebook-sao-hackeados-e-tem-mensagens-privadas-colocadas-a-venda.ghtml> >. Acesso em: 3 nov.2018.

BESSA, Programação Orientada a Objetos: uma introdução.hardware.com.br, 20 out.2007.Disponível em: < <https://www.hardware.com.br/artigos/programacao-orientada-objetos/> >. Acesso em: 05 setem.2018.

BLOOMBERG, Escândalo do Facebook isola Zuckerberg no setor de tecnologia. EXAME, 3 abri.2018. Disponível em: < <https://exame.abril.com.br/tecnologia/escandalo-do-facebook-isola-zuckerberg-no-setor-de-tecnologia/> >. Acesso em: 4 abri.2018.

CARNEIRO, Linux implementa código de conduta para seus programadores.tecmundo, 23 set.2018.Disponível em:< <https://www.tecmundo.com.br/mercado/134459-linux-implementa-codigo-conduta-programadores.htm> >. Acesso em: 24 set.2018.

CICILIA, Diretrizes de boas práticas para programação confiável. ti especialistas, 2 out.2012. Disponível em: < <https://www.tiespecialistas.com.br/diretrizes-de-boas-praticas-para-programacao-confiavel/> >. Acesso em: 01 setem.2018.

CIRIACO, JavaScript foi a linguagem mais usada em janeiro de 2018, aponta análise.tecmundo, 12 mar.2018.Disponível em: < <https://www.tecmundo.com.br/internet/128093-javascript-linguagem-mais-usada-janeiro-2018-aponta-analise.htm> >. Acesso em: 05 nov.2018.

COSTA.10 falhas de software que marcaram a história.base2, 29 jan.2014. Disponível em: < <http://www.base2.com.br/2014/01/29/10-falhas-software-marcaram-historia/> >. Acesso em: 20 agost.2018.

CRONAPP.Os maiores desafios e barreiras no desenvolvimento de softwares.CronApp, 9 out.2017. Disponível em: < <https://www.cronapp.io/os-maiores-desafios-e-barreiras-no-desenvolvimento-de-softwares/> >. Acesso em: 25 agost.2018.

DINO, Filantropos e investidores sociais debatem o impacto da tecnologia no terceiro setor durante evento organizado pelo IDIS, em São Paulo. EXAME, 20 set.2018.Disponível em: < <https://exame.abril.com.br/negocios/dino/filantropos-e-investidores-sociais-debatem-o-impacto-da-tecnologia-no-terceiro-setor-durante-evento-organizado-pelo-idis-em-sao-paulo/> >. Acesso em: 22 set.2018.

DINO, Tecnologia movimenta US\$ 3,7 trilhões, aumenta receitas em 70% e ganha o centro das atenções. EXAME, 16 abr.2018.Disponível em: < <https://exame.abril.com.br/negocios/dino/tecnologia-movimenta-us-37-trilhoes-aumenta-receitas-em-70-e-ganha-o-centro-das-atencoes/> >. Acesso em: 17 abri.2018.

ESPANÃ, Tecnologia e Humanidade. EXAME, 1 set.2017.Disponível em: < <https://exame.abril.com.br/blog/o-que-te-motiva/tecnologia-e-humanidade/> >. Acesso em: 07 setemb.2018.

FLORENZANO.Comando sem checagem afeta servidor e apaga 16,5 mil processos do TCE-AM. canaltech, 3 outb.2017.Disponível em: < <https://canaltech.com.br/governo/funcionario-do-tce-am-executa-script-errado-no-sql-e-apaga-165-mil-processos-101596/> >. Acesso em: 20 julh 2018.

FONSECA, Google está de olho nas startups que usam esta tecnologia.EXAME,16 mai.2018.Disponível em: < <https://exame.abril.com.br/pme/google-esta-de-olho-nas-startups-que-usam-esta-tecnologia/> >. Acesso em 17 mai.2018.

G1.Honda faz recall de 325 mil carros no Brasil para trocar 'airbags mortais'. Auto Esporte, 13 junh.2016. Disponível em: < <http://g1.globo.com/carros/noticia/2016/06/honda-faz-recall-de-325-mil-carros-no-brasil-para-trocar-airbags-mortais.html> >. Acesso em: 05 agost.2018.

MARTIN, C.R.Clean Code. São Paulo: Alta Books, 2009. 440p.

MÜLLER. Corretora brasileira de Bitcoin perde R\$ 1 milhão por saques duplicados. tecmundo, 13 març.2018. Disponível em: < <https://www.tecmundo.com.br/mercado/128143-saques-bitcoin-duplicados-geram-prejuizo-r-1-milhao-foxbit.htm> >. Acesso em: 14 julh.2018.

OLIVEIRA, Convergindo a tecnologia e a experiência do cliente. EXAME, 16 abri.2018. Disponível em: < <https://exame.abril.com.br/blog/relacionamento-antes-do-marketing/convergindo-a-tecnologia-e-a-experiencia-do-cliente/> >. Acesso em: 20 abri.2018.

PIRES, O que é Programação Orientada a Objetos e porque você precisa saber! Bencode, 2016. Disponível em: < <https://becode.com.br/programacao-orientada-a-objetos-poo/> >. Acesso em: 04 setem.2018.

PRESS, Facebook é multado no Reino Unido por violação de dados de usuários. G1, 25 out.2018. Disponível em: < <https://g1.globo.com/economia/tecnologia/noticia/2018/10/25/facebook-e-multado-no-reino-unido-por-violacao-de-dados-de-usuarios.ghtml> >. Acesso em: 28 out.2018.

RAIM. Os 5 grandes desafios no processo de Desenvolvimento de Software. OFICINA DA NET, 11 març 2010. Disponível em: < https://www.oficinadanet.com.br/artigo/desenvolvimento/os_desafios_no_processo_de_desenvolvimento_de_software >. Acesso em: 26 agost.2018.

REUTERS, Falha no WhatsApp permitiu invasão a contas por chamada de vídeo. EXAME, 10 out.2018. Disponível em: < <https://exame.abril.com.br/tecnologia/falha-no-whatsapp-permitiu-invasao-a-contas-por-chamada-de-video/> >. Acesso em: 20 out.2018.

ROGÉRIO, Desenvolvimento em CSS – Manual de boas práticas. PINCELADAS DA WEB, 14 fev.2018. Disponível em: < <http://www.pinceladasdawe.com.br/blog/2008/02/14/desenvolvimento-em-css-manual-de-boas-praticas/> >. Acesso em: 03 setem.2018.

ROHR, Autor do vírus Mirai terá de pagar US\$ 8,6 milhões por ataque a universidade. G1, 2 nov.2018. Disponível em: < <https://g1.globo.com/economia/tecnologia/blog/altieres-rohr/noticia/2018/11/02/autor-do-virus-mirai-tera-de-pagar-us-86-milhoes-por-ataque-a-universidade.ghtml> >. Acesso em: 4 nov.2018.

ROHR, Grupo que criou vírus responsável por apagão na Ucrânia tem envolvimento com NotPetya e espionagem, diz empresa. G1, 19 out.2018. Disponível em: < <https://g1.globo.com/economia/tecnologia/blog/altieres-rohr/post/2018/10/19/grupo-que-criou-virus-responsavel-por-apagao-na-ucrania-tem-envolvimento-com-notpetya-e-espionagem-diz-empresa.ghtml> >. Acesso em: 23 out.2018.

ROHR, O que fazer para saber se o WhatsApp está seguro ?G1, 1 nov.2018. Disponível em: < <https://g1.globo.com/economia/tecnologia/blog/altieres-rohr/post/2018/11/01/o-que-fazer-para-saber-se-o-whatsapp-esta-seguro.ghtml> >. Acesso em: 2 nov.2018.

ROHR, Pesquisador divulga brechas em roteadores da D-Link.G1, 22 out.2018. Disponível em: < <https://g1.globo.com/economia/tecnologia/blog/altieres-rohr/post/2018/10/22/pesquisador-divulga-brechas-em-roteadores-da-d-link.ghtml> >. Acesso em: 23.out.2018.

SOUZA, Conheça as linguagens de programação que mais crescem no mercado de TI.tecmundo, 21 dez.2017. Disponível em: < <https://www.tecmundo.com.br/software/125425-conheca-linguagens-programacao-crescem-mercado-ti.htm> >. Acesso em: 05 nov.2018.

SOUZA. EVOLUÇÃO DA TECNOLOGIA DE INFORMAÇÃO.WEB ARTIGOS, 12 setem.2010. Disponível em: < <https://www.webartigos.com/artigos/evolucao-da-tecnologia-de-informacao/47044/> >. Acesso em: 23 agost.2018.

SUTHERLAND, J. Scrum A Arte de Fazer o Dobro do Trabalho na Metade do Tempo. São Paulo: LeYa, 2014. 231p.

VIEIRA, Demanda cresce por profissionais de TI durante a crise. EXAME, 18 abri.2017. Disponível em: < <https://exame.abril.com.br/carreira/demanda-cresce-por-profissionais-de-ti-durante-a-crise/> >. Acesso em: 06 setem.2018.

VIEIRA, Java? C? Ranking da IEEE Spectrum revela a linguagem mais usada em 2017.tecmundo, 27 jul.2017. Disponível em: < <https://www.tecmundo.com.br/software/119960-java-c-ranking-ieee-spectrum-revela-linguagem-usada-2017.htm> >. Acesso em: 29 nov.2018.

WILLIAN, 10 Boas Práticas de Programação.DevWiliam, 06 junh.2016. Disponível em: < <http://www.devwilliam.com.br/extra/profissional/10-boas-praticas-de-programacao> >. Acesso em: 27 agost.2018.