INSTITUTO DOCTUM DE EDUCAÇÃO E TECNOLOGIA
FACULDADES INTEGRADAS DE CARATINGA CIÊNCIA DA COMPUTAÇÃO

LEONARDO WEBSTER RIBEIRO DA SILVA

ESTUDO DOS BENEFÍCIOS DA UTILIZAÇÃO DO LARAVEL FRAMEWORK NA MANUTENIBILIDADE DE *SOFTWARE*

CARATINGA - MG
2018

ESTUDO DOS BENEFÍCIOS DA UTILIZAÇÃO DO LARAVEL FRAMEWORK NA MANUTENIBILIDADE DE *SOFTWARE*

Monografia apresentada à banca examinadora da Faculdade de Ciência da Computação das Faculdades Integradas de Caratinga como exigência parcial para obtenção do grau de bacharel em Ciência da Computação, sob orientação do professor Maicon Vinícius Ribeiro.

CARATINGA - MG

2018

ERRATA

Silva, Leonardo Webster Ribeiro da. **Estudo Dos Benefícios Da Utilização Do Laravel Framework Na Manutenibilidade De *Software*.** 2018. 122 f. Trabalho de conclusão de curso (Curso em Ciência da Computação), Faculdades Integradas de Caratinga – Caratinga. 2018.

Folha	Linha	Onde se lê	Leia-se
77	Gráfico 21	Qual o Framework utilizado?	Frameworks:
79	Gráfico 23	Qual o Framework utilizado?	Frameworks:
80	Gráfico 24	Qual o Framework utilizado?	Frameworks:
81	15	*software*	*software*

AGRADECIMENTOS

Primeiramente gostaria de agradecer muito a Deus por ter me iluminado um caminho, me capacitando, me dando forças para continuar a seguir em frente e por todas as pessoas importantes pra mim que foram colocadas em meu caminho.

Agradeço o apoio da minha família, principalmente meus pais, Rozane e Carlos que sempre me apoiaram e incentivaram até o fim. Agradeço a minha namorada Camilla que por todo apoio, paciência e incentivo neste ano.

Agradeço também a todos os professores, amigos, colegas pela amizade e conhecimentos que compartilhamos. Agradeço especialmente meu professor e orientador Maicon Ribeiro, por suas orientações, pela paciência e pelo privilégio de ser ter sido aluno.

“Se mesmo depois da exaustão total da sua mente e do seu corpo, você ainda tiver algo que não consegue abandonar, então essa é a sua verdade, que ninguém pode tirar de você.”

Nobuhiro Watsuki

RESUMO

Com a constante evolução tecnológica, existe uma grande diversidade de *software*s que objetivam atender as diferentes necessidades dos usuários. No entanto, a criação desses é demorada e de alto custo. Diante disso se faz necessário intentar sobre formas para agilizar o processo de criação bem como diminuir o custo, para isso ocorrer é indispensável utilizar ferramentas que ajudem na manutenibilidade de *software*, a etapa mais demorada de sua fabricação.

O objetivo desse trabalho é averiguar se o Laravel Framework é uma ferramenta de auxílio para a manutenibilidade de *software*. Para isso, foi realizado um estudo bibliográfico sobre *Softwares* estudando o percurso para a sua criação, seu ciclo de vida, tipos de *software*, o que é manutenção e manutenibilidade de *software*, definindo o que é considerado como qualidade de *software*, também foi estudado sobre Frameworks e por fim sobre o Laravel.

A partir destes estudos realizados foi montado um questionário do tipo quantitativo utilizando a ferramenta do Google *Forms*, contendo perguntas objetivas para serem respondidas por profissionais ou atuantes da área, o questionário foi dividido em um todo de 8 seções incluindo apresentação e agradecimentos, estas perguntas objetivaram identificar a experiência dos entrevistados com a utilização de frameworks, em específico o Laravel.

Após a aplicação do questionário foi realizada uma coleta e análise dos dados e pôde-se verificar a partir das respostas dos 104 participantes que essa ferramenta é capaz de auxiliar na manutenção do *software,* mas não em sua totalidade uma vez que, para a maioria dos respondentes o Laravel framework não abrange todas as condições necessárias para a manutenibilidade de um *software*.

Palavras-chaves: *Software*. Frameworks. Laravel Framework. Manutenibilidade.

ABSTRACT

With the constant technológical evolution, there is a great diversity of software that aim to meet the different needs of users. However, the creation of these is time consuming and costly. In addition it is necessary to bring about ways to streamline the process of creation as well as decrease the cost, for this to occur it is essential to use tools that help in the maintainability of software, a step further delay of its manufacture.

The objective of this study is to demonstrate whether the laravel Framework is a tool to help to the maintainability of software. To do this, we performed a bibliographic study about Software studying the path for its creation, its life cycle, types of software, which is maintenance and maintainability of software, defining what is considered as software quality, was also studied about Frameworks and finally about the laravel.

From these studies it was mounted a questionnaire of quantitative type using the tool from Google Forms, containing objective questions to be answered by professionals or working in the area, the questionnaire was divided into a total of 8 sections including presentation and acknowledgments, these questions aimed to identify the interviewees' experience with the use of frameworks, in particular the laravel.

After the application of the questionnaire was performed by a collection and analysis of data and could be verified from the responses of one hundred and four participants that this tool is able to assist in the maintenance of the software, but not in its entirety since, for the majority of respondents The Laravel framework does not cover all the conditions necessary for the maintainability of a software.

Keywords: *Software*. *Software* maintenance. Maintainability. Frameworks. Laravel Framework.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ABNT	International Organization for Standardization (Organização Internacional de Normalização).
AMN	Asociación Mercosur de Normalización (Associação Mercosul de Normalização).
API	Application Programming Interface (Interface de programação de aplicativos).
CLI	Command-Line Interface (em português interface de linha de comando).
IEC	International Electrotechnical Commission (Comissão Eletrotécnica Internacional).
ISO	Organização Internacional de Normalização (International Organization for Standardization).
MVC	Model-view-controller, padrão de arquitetura de software que separa a representação da informação da interação com o usuário.
MySQL	É um SGBD relacional, open source que utiliza a linguagem SQL como interface.
ORM	Object-relational mapping (Mapeamento de objeto relacional).
ORMs	Plural de ORM.
PHP	Hypertext Preprocessor (Pré-processador de hipertexto).
SGBD	Sistema de Gerenciamento de Banco de Dados.
TI	Tecnologia da Informação.
URL	Uniform Resource Locator (Localizador uniforme de recursos).
URLs	Plural de URL.
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A cada dia que se passa nossa tecnologia evolui em um novo aspecto, há alguns anos tínhamos a ideia de que os computadores eram o futuro e de uma certa forma ouve um acerto nisso, mas não como imaginava. Antes se pensava no computador como uma máquina de escritório, algo usado estritamente para o trabalho, hoje existe praticamente um computador em toda as casa, em viagens, nas mãos ou bolsos de cada pessoa, nas cozinha, quase podemos dizer que eles estão em todos os lugares, de tamanhos que vão de algo menor que um grão de arroz a um que pode cobrir uma grande sala. E com um mercado tão rico em novas tecnologias o que não falta são diferentes tipos de *softwares* que surgem para atender a esta demanda.

Mas o processo de construção de um *software*, é algo demorado e caro, por seu valor elevado é essencial que seja seguido um processo de fabricação para que ele tenha o melhor funcionamento possível, evitando assim gastos desnecessários. Uma das etapas do desenvolvimento de um *software* é a manutenção, onde é realizada a correção de erros e a criação de novas funcionalidades e é nessa parte que segundo Sommerville (2011) detém, aproximadamente, cerca de dois terços do orçamento, um gasto maior até que a etapa de desenvolvimento do *software* que ocupa um terço do orçamento, ainda de acordo com Sommerville (2011) em alguns casos o custo da manutenção de *software*, dependendo do tipo de sistema, pode custar até quatro vezes mais que o custo do desenvolvimento.

Por se tratar de algo tão custoso, mas ao mesmo tempo tão importante para a sociedade se faz necessário pensar em formas que se consiga baixar o valor do desenvolvimento sem abaixar a qualidade, assim como Sommerville (2007) diz a produção de *softwares* confiáveis com qualidade e baixo custo de fabricação bem administrados torna-se cada vez mais importantes.

De acordo com Ribeiro (2013), atualmente existem técnicas, ferramentas ou métodos de fabricação de *software* que podem ser muito úteis, como orientada a objetos, engenharia de *software* baseada em componentes, a programação orientada a serviços, a programação orientada a aspectos, frameworks, padrões de projeto (Design Patterns) e Web Services.

Este trabalho aborda uma das ferramentas citadas por Ribeiro (2013), os frameworks. Frameworks são ferramentas utilizadas para agilizar a construção de um *software*, segundo Silva M. (2016) possuem pontos flexíveis chamados de hot spots, essa flexibilidade se refere a capacidade desses pontos/componentes serem adeptos a sofrerem modificações, ou em outras palavras, são pontos em que os envolvidos no desenvolvimento do *software* possuem grande autonomia.

E para que o trabalho tenha um enfoque melhor, foi analisado e definido utilizar um dos frameworks mais utilizados segundo Silva M. (2016), onde foi observado que a partir de sua pesquisa o Laravel foi apontado como o mais utilizado ficando com 38,60%.

2 REFERENCIAL TEÓRICO

Neste capítulo, serão explicados os conceitos-chave que serviram de embasamentos na construção deste trabalho. Abordando os assuntos referentes a engenharia de *software*, frameworks e o Laravel framework.

2.1 SOFTWARE

Apesar de muitos não saberem reconhecer o sentido da palavra *software* ele está presente em todas as áreas do cotidiano atual, ele é facilmente encontrado nos objetos a volta, com objetivos e embalagens, formas e tamanhos diferentes, portanto, é certo dizer que o *software* é um componente fundamental na sociedade moderna.

Segundo Pressman (2011), *softwares* são instruções que, quando executadas, fornecem características, funções e desempenho desejados, não é algo físico e que não se desgasta. Para Sommerville (2011) "*Softwares* são programas de computador e documentação associada. Produtos de *software* podem ser desenvolvidos para um cliente específico ou para o mercado em geral". Apesar de Pressman (2011) e Sommerville (2011) parecerem divergir um pouco suas ideais, ambos seguem em paralelo, a diferença entre elas é a formalidade empregada na construção de sua definição. Em um sentido mais abrangente, seguindo a ideia de Pressman (2011) e Sommerville (2011), *software* ou programa de computador é uma sequência de comandos e instruções lógicas que executam uma tarefa, ele possui toda uma documentação, dados e configurações, ele não se desgasta pelo tempo, não é algo físico.

Pressman (2011) diz que um *software* por natureza é "um transformador de informações, ele produz, gerência, adquiri, modifica e transmite informações, ele pode trabalhar com informações tão simples como um único bit ou tão complexas como uma apresentação de multimídia", para ele o *software* pode possuir duplo papel, pois ele é tanto um produto quanto um veículo que distribui um produto.

2.1.1 TIPOS DE *SOFTWARE*

Assim como tudo possui uma classificação, os *softwares* também possuem as classificações de *softwares*, podem se diferenciar conforme o autor utilizado, portanto, neste subcapítulo serão descritas duas ideias de autores mais renomados na área de engenharia de *software* e será apresentada uma análise de suas classificações ao final do subcapítulo.

Para Sommerville (2011) existem dois tipos de produtos de *software*:

- **Produtos Genéricos:** são *softwares* feitos para atender a uma gama muito grande de clientes, portanto, não possuem muitas restrições, são postos à venda para qualquer cliente interessado. Exemplos destes tipos de produto são os reprodutores de música, navegadores, editores de texto e outros.
- **Produtos sob encomenda:** são *softwares* feitos para atender clientes específicos, clientes particulares. São *softwares* feitos especialmente para este cliente, geralmente são projetos com restrições que procuram atender a necessidade do cliente. Exemplo destes produtos são sistemas de tráfego aéreo, financeiros e educacionais.

Pressman (2011) aponta a existência de sete tipos de *softwares*:

- ***Software* de sistema:** são programas criados para atender outros programas. Atendem *softwares* de sistemas e processam determinadas estruturas informações complexas como, por exemplo: editores, compiladores e outros utilitários de gerência de arquivos. Também podem processar dados amplamente indeterminados como, por exemplo: drivers de *software* de rede ou processadores de telecomunicações.
- ***Software* de aplicação:** são programas independentes que atendem necessidades específicas para um negócio. As aplicações desta área trabalham com análise de dados comerciais ou técnicos de uma forma que auxilie na tomada de decisões.
- ***Software* de engenharia/científico:** são programas de “cálculos de massa” focados em cálculos mais pesados e complexos nas áreas científicas. Alguns exemplos de aplicação são: astronomia, vulcanologia, análise de estresse automotivo, dinâmica orbital, projeto auxiliado por computador, biologia molecular, análise genética e meteorologia, entre outros.

- ***Software* embarcado:** são programas que vêm juntos com produtos, eles fazem parte de suas funcionalidades, controlando características e funções para o usuário e para o próprio sistema. Suas funções são limitadas e específicas, mas significativas, em geral, fornecem ao usuário a capacidade de controle sobre algo. Alguns exemplos de aplicação são: painéis de automóveis, micro-ondas, televisões e monitores.
- ***Software* para linha de produtos:** são programas que promovem capacidades específicas de utilização de muitos clientes diferentes. Estes *softwares* para linha de produtos, podem se concentrar em mercados limitados e pequenos, ou com um consumidor em massa.
- **Aplicações Web/aplicativos móveis:** são programas voltados para redes de ampla variedade de aplicações, contemplando aplicativos voltados para navegadores e dispositivos móveis;
- ***Software* de inteligência artificial:** são programas que fazem uso de algoritmos não numéricos, para solução de problemas complexos não passíveis de computação ou análise direta. Algumas áreas de aplicação desta área incluem robótica, sistemas especialistas, reconhecimento de padrões e redes neurais, entre outras.

Analisando o pensamento de Sommerville (2011) sobre os tipos de *software*, é possível perceber uma simplicidade e objetividade na classificação do *software* maior que a apresentada por Pressman (2011), no entanto, suas visões dos tipos de *software* em certos pontos tangenciam, um exemplo claro disso pode ser observado na explicação de Sommerville (2011) sobre os produtos feitos sob encomenda, e na explicação de Pressman (2011) quanto aos tipos de *software* de aplicação, engenharia/científico e embarcado. Segundo eles estes *software*s são produzidos para atender características específicas, para problemas e clientes específicos.

Assim, é possível perceber que alguns pensamentos de Sommerville (2011) e Pressman (2011) seguem uma linha de raciocínio muito próxima, apesar de suas diferenças na explicação.

2.1.2 PRODUÇÃO DE *SOFTWARE*

Para a construção ou produção de algo, é comum que se siga algumas instruções ou processos, na produção de *software* é comum que a utilização de algum processo de *software*, apesar de não ser algo necessariamente obrigatório, segundo Macoratti a utilização de um processo de *software* é apontada como um fator primordial para o sucesso de empresas de desenvolvimento de *software*, portanto, para compreender melhor o processo de produção de um *software* neste subcapítulo serão apresentadas algumas informações acerca do assunto.

Segundo Macoratti (apud Jalote 2005) o processo de *software* são um conjunto de atividades, ligadas por padrões de relacionamento entre elas, para que às atividades operem corretamente de acordo com os padrões requeridos e o resultado desejado seja produzido. Macoratti (apud Jalote 2005) diz também que o resultado desejado de qualquer *software* é um de alta qualidade e custo baixo, ele afirma também que um processo que não aumenta a produção ou não pode produzir *software* de boa qualidade, não é um processo adequado.

Para Humphrey (1990), existem razões pelas quais é necessário definir um processo padrão na produção de *softwares,* elas são:

- Diminuição dos problemas relacionados aos treinamentos, revisões e suporte às ferramentas;
- As experiências adquiridas podem ser incorporadas ao processo padrão, contribuindo para aperfeiçoamento dos processos definidos;
- Aperfeiçoamento do tempo e esforços gastos na definição dos processos de *software* mais adequados a projetos;

Segundo Schwartz (1975) e Pressman (2006), as principais fases no processo de um *software* são:

1. **Especificação de Requisitos do *software* (Planejamento**): Identificação das necessidades ou requisitos para as funcionalidades a serem executadas.
2. **Projeto de Sistema (Projeto):** Descrição de todos os componentes necessários definidos nos requisitos do sistema para codificação do sistema.
3. **Programação (Codificação):** Geração da codificação do sistema e a lógica envolvida.

4. **Verificação e Integração (Testes):** Análise de satisfação de requisitos iniciais e do resultado final do produto.

Pressman (2006) cita que as últimas duas partes do processo eram repetidas a medida da necessidade, a cada modificação ou evolução do *software*. Sommerville (2011) utiliza um modelo muito similar ao já citado por Pressman e Schwartz, porém, para ele o projeto de sistema e programação era uma só, e ele utiliza mais uma fase, que é dita como a evolução de *software*.

2.1.3 CICLO DE VIDA DE UM *SOFTWARE*

Para a produção de um produto final de qualidade, existem certas diretrizes que costumam ser seguidas, elas elaboram um ciclo de vida do produto, desde a obtenção de matéria prima ao produto final objetivado. Segundo Florentin (2015) o ciclo de vida de um *software* é geralmente definido por fases que conectam o início e o fim de um projeto, este tipo de organização é um mecanismo de controle gerencial do processo de produção.

Macoratti e Florentin (2015) possuem uma linha de raciocínio muito similar quanto o ciclo de vida de um *software*, Macoratti cita uma combinação das classificações dadas pelos autores Schwartz, Pressman, Sommerville para identificar as atividades realizadas nas fases no processo de *software*, segue a mesma ideia de Florentin (2015), que utiliza em seu estudo Sommerville e Bezerra em sua bibliografia auxiliar, a seguir, segue a combinação de suas ideais referente às etapas do ciclo de vida de um *software*:

1. **Especificações de *software*:**

A especificação define o *software* que será produzido, impondo objetivos e restrições de acordo com as necessidades do cliente (Figura 1).

1.1. Engenharia de Sistema: estabelecer uma solução que satisfaça o problema e atenda às necessidades do cliente e usuários, envolvendo questões extra ao *software*;

1.2. Análise de Requisitos: realiza um levantamento das utilidades do *software* a ser implementado. A análise produz um relatório com as especificações de requisitos em formato de documento.

1.3. Especificação de Sistema: descrição do funcionamento do sistema. Pode-se incluir planos de teste para validar adequadamente às funcionalidades.

Figura 1 - Ciclo de Vida do *Software* – Especificação
Fonte: FLORENTIN, 2015

2. **Desenvolvimento de *software*:**

Na etapa de desenvolvimento é trabalhado o projeto e desenvolvimento do *software*, que segue as especificações dadas na etapa de especificação (Figura 2).

Figura 2 - Ciclo de Vida do Software – Desenvolvimento
Fonte: FLORENTIN, 2015

2.1. Projeto Arquitetural: Desenvolvimento de um modelo dos conceitos para o sistema.

A modularização pode auxiliar em fatores como:

- Lidar com a complexidade do sistema;
- Manter coesão dos subsistemas;
- Reduzir acoplamento global do sistema;
- Decidir como desenvolver cada parte do sistema;
- Auxilia na divisão de tarefas, permitindo um desenvolvimento em paralelo;
- Permite a reutilização de código para subsistemas;
- Compartilhamento de subsistemas permitindo que várias aplicações em um mesmo subsistema;

O modelo conceitual é composto de módulos relacionados, estes módulos podem variar quanto a seu nível de dependência.

2.2. Projeto de Interface: Estudo e definição da interface de cada módulo do sistema.

Alguns dos pontos citados como benefícios do projeto de interface são:

- Interface amigável, favorecendo menor necessidade de suporte e melhor operabilidade;
- Melhorando a qualidade dos produtos de *software* e aumentando a satisfação do usuário;
- Padronização de componentes;

2.3. Projeto Detalhado: Parte do processo de produção de um *software* é tradução dos módulos para o pseudocódigo. O pseudocódigo é uma maneira simples de escrever algum código.

2.4. Codificação: Etapa de iniciação do processo de desenvolvimento em linguagem de computador.

3. **Validação de *Software***

Na validação é realizado um conjunto de análises e testes a fim de garantir a qualidade e o funcionamento das funcionalidades do *software* a partir das especificações feitas (Figura 3).

Figura 3 - Ciclo de Vida do *Software* – Validação
Fonte: FLORENTIN, 2015

3.1. Teste de Unidade e Módulo: a realização de testes para verificar a presença de erros e comportamento adequado a nível das funções e módulos básicos do sistema;

3.2. Integração: é a reunião dos diferentes módulos em um produto de *software* homogêneo, e a verificação da interação entre estes, quando operando em conjunto.

4. **Manutenção e Evolução de *software***

Nesta etapa é trabalhado a modificação e evolução das funcionalidades e requisitos do cliente para com o *software* (Figura 4).

4.1. Nesta fase, o *software* em geral entra em um ciclo iterativo que abrange todas as fases anteriores.

Figura 4 - Ciclo de Vida do *Software* – Evolução
Fonte: FLORENTIN, 2015

2.1.4 MANUTENÇÃO DE *SOFTWARE*

Uma das etapas da construção de um *software* é a manutenção, ela pode ser descrita como toda e qualquer modificação, como correções, melhorias ou adaptações devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.

Segundo Pigoski (1996, p.46) :

> "Manutenção é a totalidade das atividades necessárias para prover, minimizando o custo, suporte a um sistema de *software*. As atividades são executadas tanto nos estágios de pré-entrega quanto nos estágios de pós-entrega. As atividades de pré-entrega incluem o planejamento para entrada em operação, suportabilidade e definição de logística. As atividades de pós-entrega incluem modificação do *software* e operação de um help-desk".

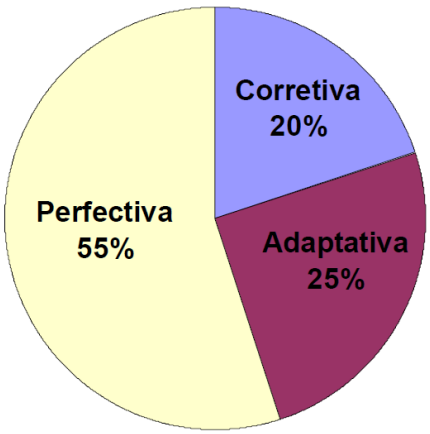
2.1.4.1 Tipos de Manutenção

Para os autores Lientz e Swanson (1980), a manutenção de *software* pode ser classificada em três categorias:

- Manutenções do tipo corretivas: Tem como objetivo reparar defeitos de funcionalidades, que surgem durante a utilização do sistema, o que

inclui correções de processamento, *bugs* de *software*, de performance ou implementação.

- Manutenções adaptativas: Tem como objetivo a adequação do *software* ao contexto no qual ele deve operar, para que assim ele acompanhe as mudanças dos ambientes externos, como regras de negócio, mudança de constituições e leis, mudança no hardware, nova versão de sistema operacional e etc.
- Manutenções Perfectivas: Visa manutenções de aprimoramento do



software além dos requisitos originais, acrescentando novos recursos, funcionalidades e ferramentas, buscando sempre melhorar o sistema.

Figura 5 - Percentuais de esforço de manutenção
Fonte: Pigoski, 1996

A Figura 5 mostra o gráfico montado por Pigoski (1996) com os porcentuais de esforços aplicados em cada tipo de manutenção. Segundo ele a ideia de existência de um sistema ideal, sem nenhum erro e sem necessidade de manutenção é uma equivocação, pois os requisitos do usuário estão em constante evolução, sempre buscando vantagem em relação aos concorrentes, reduzir gastos operacionais, ou simplesmente adequar a novas leis, regras e processos.

2.1.5 QUALIDADE DE *SOFTWARE*

O conceito qualidade é algo muito relativo, para um computador por exemplo, como qualidade pode ser observado fatores como processador, memória RAM, placas gráficas, armazenamento, número de entradas USB, consumo de energia,

tamanho, designer, peso, cor e outros fatores. As qualidades de um produto podem ser consideradas fortemente relacionadas a sua utilização, os requisitos que pretende atender.

Um cliente que utiliza um computador para apenas acessar a internet e outras coisas de baixo processamento como datilografia não precisaria se preocupar com vários dos itens citados acima pois uma máquina simples pode atender seus requisitos, já um cliente que utiliza programas que requer um alto consumo de processamento como edição de imagens ou vídeo, programação e jogos, precisaria de uma máquina mais potente para atender seus requisitos.

Como se percebe, a qualidade está relacionada diretamente ao atendimento dos requisitos do cliente para com um produto, na produção de *softwares*, estas condições geralmente são especificadas por especialistas que analisam as necessidades do cliente para a construção do projeto.

> Qualidade é a totalidade de características e critérios de um produto ou serviço que exercem suas habilidades para satisfazer às necessidades declaradas ou envolvidas. (ISO 9126-1, 2003).

Segundo Pressman (2011) e Sanders, Curran (1994), apontam que a qualidade de um *software* engloba o grau de atendimento às funções e características apontadas no modelo de requisitos. Assim, um produto que segue as especificações e requisitos, satisfazendo todas as necessidades do cliente sob todos os aspectos do produto é considerado um *software* de boa qualidade.

Durante o decorrer da história diversos métodos de avaliação foram criados para auxiliar na melhora dos *softwares* em geral. Esses modelos ajudam projetistas e programadores a criar algoritmos cada vez melhores e mais úteis.

Segundo Gomes (2016) em 1976 Boehm, Brown e Lipow definiram o um modelo de árvore de atributos de qualidade de *software* (figura 6), em seu modelo destacaram a aquisição do pacote de *software*, que deve ter os seguintes atributos de nível médio na estrutura hierárquica: portabilidade, confiabilidade, eficiência, engenharia humana e facilidades de teste, uso e modificação.

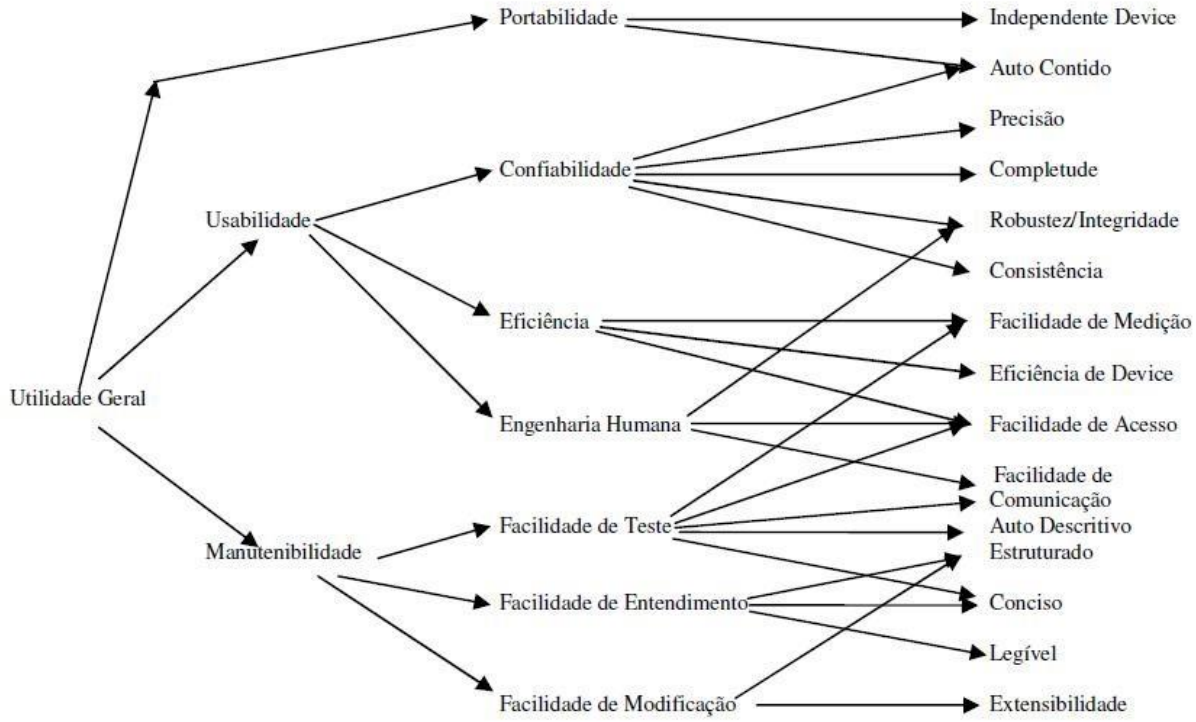


Figura 6 - Árvore de Características de Qualidade de *Software*
Fonte: GOMES, 2016.

Essa árvore de atributos é um dos modelos criados para auxiliar projetistas e programadores na criação algoritmos cada vez melhores e mais úteis.

Com os anos vários outros modelos surgiram, evoluindo para atender cada vez mais as novas necessidades e problemas identificados.

2.1.6 MANUTENIBILIDADE DE *SOFTWARE*

A manutenibilidade de *software* pode ser descrita como a facilidade com que o produto de *software* pode ser modificado. Essas modificações incluem qualquer modificação, como correções, melhorias ou adaptações devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.

Segundo Glass (1993) e Brusamolin (2004) a manutenibilidade é um atributo muito importante na produção de um *software*, um *software* de alto índice de manutenibilidade possui uma demanda menor de desenvolvimento, na medida que

os *software*s atuais podem evoluir para atender as novas necessidades, porém, isso também leva ao aumento de solicitações por manutenção.

2.1.7 NORMA DE QUALIDADE DE *SOFTWARE*

Com o alto nível de demanda que começou a surgir com o passar dos anos na área de desenvolvimento de *software,* o nível de demanda da área de análise de qualidade de *software* também aumentou substancialmente, segundo Guerra e Colombo (2009) a área de qualidade de *software* vem crescendo significativamente, pois cada vez mais os usuários tendem a exigir mais eficiência, eficácia, segurança e outras características de qualidade importantes para o *software.* Pressman (2011) também destaca que várias empresas reconheciam que um grande investimento todos os anos eram perdidos em *softwares* que não apresentavam características e funcionalidades de um bom *software*.

Com o crescimento desta demanda viu-se a necessidade de estabelecer normas para serem aplicadas a todos estes casos, o que levou ao surgimento de quatro organizações, a Associação Brasileira de Normas Técnicas (ABNT), International Organization for Standardization (Organização Internacional de Normalização - ISO), Asociación Mercosur de Normalización (Associação Mercosul de Normalização - AMN) e International Electrotechnical Commission (Comissão Eletrotécnica Internacional – IEC).

Em vinte e oito de setembro de 1940 foi fundada a Associação Brasileira de Normas Técnicas, que se trata de uma associação nacional de normalização privada e sem fins lucrativos. Ela é um membro fundado da ISO (Organização Internacional de Normalização) que é uma organização mundial fundada em 1947 e da AMN (Associação Mercosul de Normalização) criada em 1991. E também é membro da IEC (Comissão Eletrotécnica Internacional – IEC) uma organização mundial fundada em 1906. Estas organizações vêm trabalhando em normas para certificar produtos, sistemas outros itens, a fim de garantir credibilidade, ética e reconhecimento dos serviços prestados.

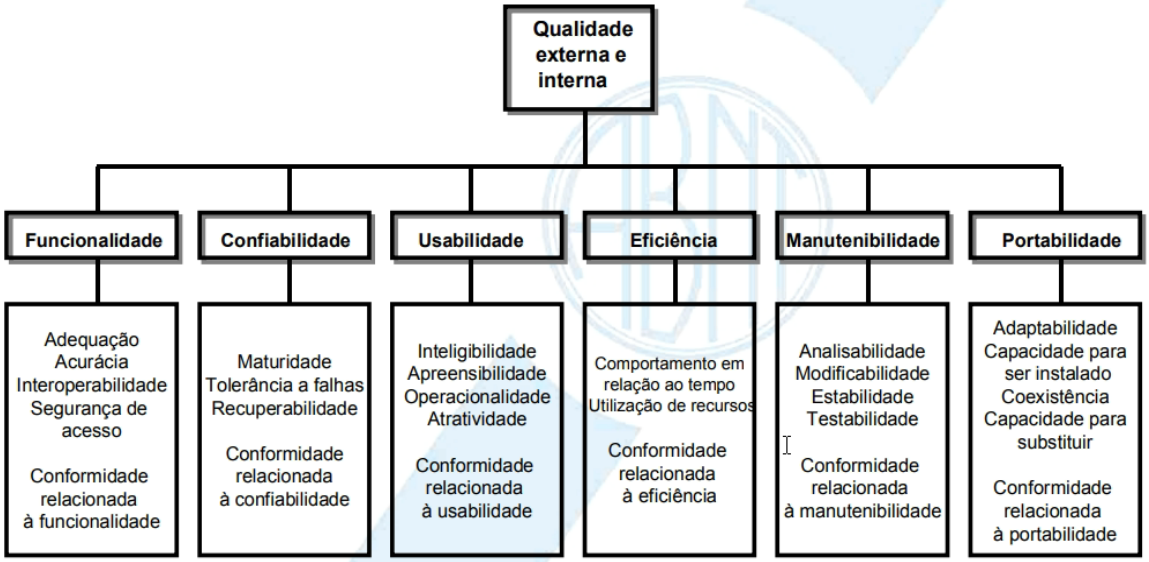


Figura 7 - Modelo de qualidade para qualidade externa e interna
Fonte: ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (2003)

A Figura 7 apresenta o modelo de qualidade de *software* presente na ISO 9126-1 (2003), segundo ela as qualidades de um produto podem ser divididas em seis categorias, dentre elas: funcionalidade; confiabilidade; usabilidade; eficiência e manutenibilidade e portabilidade. E cada uma destas categorias possui um conjunto de subcategorias que as complementam.

O atributo citado mais importante para este trabalho é a manutenibilidade, ela é definida como o a capacidade de um produto de *software* de ser modificado. Ainda na norma o atributo é dividido em cinco subcategorias:

- Analisabilidade: atributo que permite diagnóstico de deficiências ou causas de falhas no *software*, ou a identificação de partes a serem modificadas.
- Modificabilidade: atributo que permite que modificações especificadas sejam implementadas, essas modificações se referem a todo o projeto, código, documentação ou projeto.
- Estabilidade: atributo referente a capacidade de evitar efeitos imprevistos ocasionados por modificações.
- Testabilidade: atributo referente a capacidade de validar modificações no *software*.

- Conformidade relacionada à manutenibilidade: atributo relacionado a capacidade do *software* de estar de acordo com normas ou convenções relacionadas à manutenibilidade.

2.2 FRAMEWORK

Neste subcapítulo serão apresentados conceitos sobre framework na visão de alguns atores a fim de apresentar estas ferramentas para auxiliar no entendimento do presente trabalho.

Segundo Gamma (1995), a reutilização de código-fonte é o meio correto para analistas de *software*, pois eles devem ser capazes de resolver problemas não partindo de princípios elementares, ou do zero, mas sim de reutilizar soluções que já funcionaram no passado.

Dentro deste contexto, surgem ferramentas para auxiliar nesse processo. Uma ferramenta constantemente utilizada dentre essas são os frameworks. Para Pinto (2000) e Pree e Sikora (1997), um framework pode ser definido como uma arquitetura para uma família de subsistemas e oferece os construtores básicos para criá-los.

Segundo Minetto (2007, p 17) pode ser descrito como, “uma base de onde se pode desenvolver algo maior ou mais específico. É uma coleção de código-fonte, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de novos *software*s”.

2.2.1 VANTAGENS EM USAR UM FRAMEWORK

Os frameworks são ferramentas que podem auxiliar de diversas formas no desenvolvimento do de sistemas. Russel (2016, apud Silva M. 2016, p.23) e Minetto (2017) destacaram alguns de seus benefícios como sendo:

- A estrutura definida se torna familiar em todos os sistemas utilizados, gerando uma padronização nos projetos;

- Contribuição da comunidade para o melhoramento do código do framework, frameworks livres possuem um grande apoio da comunidade para a evolução e aprendizado do próprio;
- Alguns frameworks procuram transformar tarefas repetitivas em projetos em automatizadas, funcionalidades pré-definidas são realizadas sem a necessidade de reinventar para cada nova aplicação;
- Módulos, bibliotecas disponíveis para adicionar a aplicação;
- Facilidade na geração de testes, melhorando definidamente a testabilidade de funcionalidades do sistema;
- Facilidade na geração de documentação.
- Integração com ORMs (Object-relational mapping);
- Uso pré-estabelecido de padrões de projeto;
- Contribui para reusabilidade, analisabilidade e manutenibilidade do código;

Segundo Bustamante (2008), os frameworks nos domínios de aplicações web, são ferramentas que contém uma grande diversidade de funcionalidades prontas para serem usadas, e essas funcionalidades geralmente tratam de problemas comuns que aplicações web possuem. Para Souza (2010), atualmente a utilização de frameworks se tornou algo tão habitual para aplicações web, que hoje se tornou imprescindível seu uso, seja para construção de pequenos ou de grandes sistemas.

2.3 LARAVEL

Neste subcapítulo será apresentado o Laravel framework, explicando visões de autores sobre ele e relacionado essas informações com os atributos de *software* citados na norma ISO 9126-1 (2003).

Segundo Pelizza, Bertolini e Silveira (2017) o Laravel é um *framework* PHP para desenvolvimento de sistemas web, sua arquitetura é baseada em conceitos de MVC (Model-View-Controller). Pelizza, Bertolini e Silveira (2017) também afirma que o Laravel tem como principal objetivo, auxiliar no desenvolvimento de aplicações seguras e de alto desempenho de forma ágil e simplificada, com código limpo,

incentivo a boas práticas de programação e a utilização de padrões específicos a ele determinados.

2.3.1 MODEL-VIEW-CONTROLLER

Segundo Verma (2014) o MVC ou Model-View-Controller é um conceito que vem se tornado comum como estrutura de design. Segundo Verma (2014) o MVC utilizado pelo Laravel é um padrão que visa aumentar a modularidade de sistemas de *software*, sendo dividido em três componentes básicos, Model, View, Controller. Verma (2014) descreveu estes componentes da seguinte forma:

- Model: é o componente que trabalha a interação com o banco de dados manipulando os dados, lógica e regras.
- View: é o componente que trabalha a interação com o usuário, a exibição das informações de saída e entrada de várias formas.
- Controller: é o componente que trabalha executando funcionalidades e requisições que manipulam dados através do model, e recebem e enviam dados para a View.

Pelizza, Bertolini e Silveira (2017) explicam melhor estes componentes de forma mais relacionada ao Laravel comentando da seguinte forma:

- Model (em português modelo) pode ser descrito como um gerenciador de modelos da aplicação, que faz ligações com o banco de dados, analisando lógica, os dados e regras. É uma camada entre os dados e a aplicação, que pode armazenar diversos tipos de dados, de sistemas gerenciadores de bancos de dados, como o MySQL, ou arquivos eXtensible Markup Language (XML).
- View pode ser descrita como um componente que trabalha a interação com o usuário por meio de interfaces. Ele trabalha com a representação da aplicação web, exibição de dados que componente controller recebe do model. Para implementar de maneira mais fácil pode-se usar o pacote de blades do Laravel ou com código PHP. O Laravel inicia a sua

execução com a extensão de arquivo da blade ou PHP determinando quem deve prosseguir com a execução do modelo.

- Controller ou controlador é responsável por manipular informações através do model que corresponde a ele, receber solicitações de usuários através da view para exibição de dados. Ele é considerado uma espécie de ponte ou link entre o model e a view e dispõe de duas opções de desenvolvimento de lógica, o Router e Controller. Os Routers são para tratar páginas web estáticas. Controllers são para páginas que exibem informações dinâmicas.

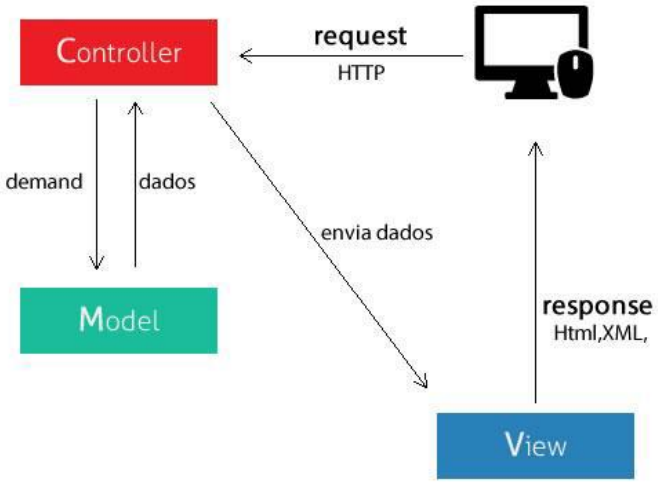


Figura 8 - Modelo de arquitetura MVC
Fonte: Pelizza, Bertolini e Silveira, 2017.

Pelizza, Bertolini e Silveira (2017) apresentam também uma representação gráfica (Figura 8) do fluxo de informações do modelo de arquitetura MVC em um contexto de Internet com uma requisição HTTP e uma resposta em formato HyperText Markup Language - HTML ou XML.

2.3.2 COMPONENTES LARAVEL

O Laravel assim como outros *frameworks* possui recursos que auxiliam no processo de desenvolvimento de aplicações, esses componentes podem influenciar de forma direta ou indireta no atributo de manutenibilidade de *software* descrito na ISO 9126-1 (2003). Nos subcapítulos a seguir serão citados alguns de seus componentes que podem auxiliar na manutenibilidade de *software*.

2.3.2.1 Arquivo de Rotas

Segundo Blazejuk (2017), o Laravel possui um arquivo de rotas, nele as URLs da aplicação são especificadas e mapeadas para métodos definidos nos Controladores (controllers).

Segundo a documentação o Laravel, da versão 5.7 o sistema permite diversas configurações para definição de rotas, como verbos HTTP (como GET, POST e outros), uso de namespace para facilitar acesso à rota, usar middlewares ou filtros de acesso, model binding e diversos outros recursos.

Implicitamente pela descrição de Blazejuk (2017), pode-se notar que a uma estrutura organizacional padronizada na utilização do arquivo de rotas que segundo Santos (2007), se encaixa na subcategoria de analisabilidade de *software*, pois segundo ele a padronização gerar facilidade de entendimento do aspecto e localização de elementos.

2.3.2.2 Artisan

Segundo Blazejuk (2017), o Laravel possui uma CLI (Command-Line Interface) chamada de Artisan, ele descreve também que através de sua utilização os desenvolvedores podem criar arquivos PHP executando comandos disponibilizados pelo Laravel. Blazejuk (2017) também acredita que com os comandos do Artisan e todos seus argumentos opcionais, esboços de classes e outros arquivos que compõem a aplicação que podem ser gerados, essa funcionalidade acelerara o processo de desenvolvimento minimizando a quantidade de erros de programação.

Algo não citado, mas implícito na explicação de Blazejuk (2017) é o fato da criação padronizada de arquivos gerados pelo framework, que segundo Santos (2007), se encaixa na subcategoria de analisabilidade de *software*, pois segundo ele a padronização pode gerar facilidade de entendimento do aspecto e localização de elementos.

2.3.2.3 Blades

Segundo Blazejuk (2017), o Laravel possui um sistema de templates denominado blade. O blade define uma linguagem que permite herdar templates, que são trechos de códigos que podem ser usados em mais de um lugar chamados (um exemplo de uso deste pode ser um menu, ou estrutura principal da página). Ainda segundo Blazejuk (2017) para criar estes uma blade os arquivos devem ter com extensão ".blade.php", já que são compilados em código PHP.

Segundo a documentação o Laravel, da versão 5.7 alguns dos modos para chamar esses trechos de código blade são @yield, @extends, @section, @stack, @push, entre outros, cada um possui uma regra e utilização diferente para cada situação.

De forma implícita esse componente auxilia muito na reutilização de código-fonte fazendo com que pequenos trechos de código sejam utilizados em diversos pontos do sistema sem necessidade de duplicidade, que segundo Ribeiro auxilia a reduzir o tempo e custo de fabricação de *software*s aumentando a lucratividade, além disso a reutilização de código-fonte auxilia na manutenibilidade e analisabilidade de *software*, devido a padronização de estrutura.

2.3.2.4 Eloquent ORM

Segundo Blazejuk (2017), o Laravel possui um recurso que facilita o acesso a registros do banco de dados, o recurso se chama Eloquent ORM, este recurso facilita acesso a registros do banco através dos model criados, através deles é possível visualizar informações, realizar inserção, atualização e remoção das informações persistidas por métodos de um objeto representando uma entrada em uma tabela. Segundo ele é um trabalho para os desenvolvedores criar e atualizar estes models (modelos) para que a aplicação possa sempre acessá-los corretamente.

2.3.2.5 Migrations

Segundo Silva W. (2015), o Laravel possui um recurso chamado migrations, este tem como objetivo gerenciar cada mudança estrutural no banco de dados, o que quer dizer que para cada criação, alteração ou remoção de estrutura do banco existirá uma migration para realizar essa operação de forma automática.

Segundo Silva W. (2015) o fato de termos exatamente cada versão estruturada de nosso banco, nos dá uma funcionalidade interessante em qualquer caso de necessidade agora se tem controle cobre o banco e em caso de necessidade facilmente pode-se usar o “roll back”.

O controle de versões do banco pode auxiliar direta e indiretamente na manutenibilidade, mais especificamente o subatributos estabilidade, segundo a ISO 9126-1 (2003) o atributo de estabilidade é referente a capacidade de evitar efeitos inesperados de correntes das modificações, neste caso o controle do banco de dados das migrations podem auxiliar muito neste processo, devido à alta capacidade de modificação.

2.3.2.6 Testes

Segundo Pelizza, Bertolini e Silveira (2017), o Laravel é baseado em TDD (Test-Driven-Development), segundo ela isto pode ser considerado uma grande vantagem em relação a testabilidade de código, uma vez que a estrutura do framework é projetada para testes antes da codificação. Segundo ela essa testabilidade do framework evita pontos indesejados no seu desenvolvimento, como repetição de código e códigos confusos. Ela ainda relaciona a melhora na testabilidade devido ao fato do *software* fornecer integração com a biblioteca de testes unitários *PHPUnit*, o que possibilita a criação e execução de testes de diversos cenários.

Segundo Gabardo (2017), o objetivo da utilização do TDD e do *PHPUnit* é facilitar a criação e execução de testes unitários no decorrer do desenvolvimento do projeto.

A testabilidade do sistema é um fator muito importante na manutenibilidade de *software* segundo ISO 9126-1 (2003), devido ao fato da testabilidade evitar cenários inesperados na execução de funcionalidades.

3 METODOLOGIA

O objetivo deste trabalho é avaliar as vantagens que o uso do *framework laravel* pode trazer quanto a manutenibilidade de *software*. Para avaliar esta finalidade, foram realizadas pesquisas bibliográficas a fim de conhecer mais sobre desenvolvimento de *software*, as etapas de sua produção, a análise de qualidade e a capacidade de manutenibilidade de um *software*. Foram realizadas também pesquisas sobre frameworks definindo suas características e vantagens da produção de *software*. E por fim foram também realizadas pesquisas sobre o framework laravel objetivando definir um pouco mais os conceitos do framework, conhecer suas funcionalidades e recursos quanto a manutenibilidade de *software*.

Após a realização dos estudos bibliográficos sobre *software*, *framework* e *laravel*, foi elaborado um questionário (APÊNDICE A) com atuantes da área de computação (profissionais atuantes da área que utilizaram ou utilizam a ferramenta), com o intuito de conhecer suas experiências quanto a manutenibilidade e definir a viabilidade do uso da ferramenta para a produção de *softwares* de qualidade. Mais detalhes da construção de *software* serão detalhados nos subcapítulos a seguir.

3.1 PÚBLICO ALVO DO QUESTIONÁRIO

Como público do questionário foram escolhidos profissionais que trabalham com desenvolvimento de *software* para plataformas web, em específico profissionais que conhecem frameworks e o Laravel, pois assim há uma probabilidade maior de se obter resultados mais confiáveis sobre a ferramenta.

Para a divulgação do questionário foram utilizados grupos de discussão em redes sociais como Facebook, Google+ e aplicativos como *Telegram, Whatsapp, Slack*, blogs e fóruns, todos relacionados a programação web, Frameworks e *Laravel.* Além disso, foram coletados *e-mails* e contatos de profissionais da área que já utilizaram ou utilizam a ferramenta. Assim buscando o maior número possível de especialistas possíveis para a participação do questionário.

Na divulgação do questionário foi preparado um pequeno texto explicando sobre o presente trabalho com seus objetivos e esclarecendo a importância de participação do mesmo, que segue com o endereço de acesso junto ao texto.

3.2 COLETA DE DADOS

Para elaboração e disponibilização do questionário foi utilizado a ferramenta gratuita do Google, chamada de Google *Forms* (em português, Google Formulários), que permite a elaboração de formulários e relatórios com os resultados dos mesmos para análises. A ferramenta possibilita aos entrevistados responderem o questionário através de qualquer aparelho com acesso à internet e browser facilitando assim o alcance dos participantes.

Para melhor esclarecimento do intuito da pesquisa para com o público respondente, foi feita uma breve introdução na primeira seção sobre o questionário, informando a fundamentação das perguntas, esclarecendo quanto a proteção dos dados do participante e uma breve apresentação do autor.

3.3 ELABORAÇÃO DO QUESTIONÁRIO

A formulação do questionário foi feita a partir dos estudos realizados através dos autores, normas e documentações já citados neste trabalho como: Pressman (2011), Ribeiro (2013), Lopes (2017), Silva M. (2016), Gomes (2016), ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (2003) e a documentação do Framework Laravel.

3.3.1 Questionário

As questões foram disponibilizadas no questionário dividindo-as em seções, ao todo foram trinta e duas perguntas dispostas em oito seções, esta organização, procurou proporcionar uma leitura agradável e lógica, desenvolvendo melhor os

assuntos e usando isso para filtrar a participação dos entrevistados para a obtenção de informações.

Na primeira seção, foram apresentados aos entrevistados os assuntos que serão abordados, o autor do estudo e seus objetivos. A segunda seção, denominada "Identificação do Respondente" possui como objetivo coletar informações quanto a experiência do entrevistado, como formação acadêmica, experiência profissional e a identificação de experiência com projetos PHP, profissionais ou não. A terceira seção “Experiência em projetos PHP”, é voltada para coletar informações sobre as experiências no desenvolvimento de projetos em PHP, e uma identificação sobre seu conhecimento com Laravel Framework. A quarta seção “Laravel Framework”, coleta informações sobre os conhecimentos do entrevistado sobre o Laravel. A quinta seção “Outro Framework”, é uma seção opcional que coleta informações sobre outro framework que o entrevistado possa ter utilizado que o levou a deixar de utilizar o Laravel. A sexta seção “Manutenibilidade de *Software*”, coleta informações sobre as preocupações no desenvolvimento do sistema que usava o Laravel. A sétima seção “Laravel Framework e Manutenibilidade”, que procura relacionar os subatributos da manutenibilidade de *software* citados na ISO 9126 com as funcionalidades encontradas no framework a finalidade de analisar a opinião dos entrevistados quanto aos benefícios que a utilização da ferramenta traz na construção de projetos. A oitava seção é o agradecimento aos participantes da pesquisa.

3.3.1.1 Primeira Seção: Apresentação

Nesta seção foi apresentado o objetivo da pesquisa, os alicerces de construção do questionário com uma média de tempo para resposta e uma pequena apresentação do autor para o entrevistado.

3.3.1.2 Segunda Seção: Identificação do Respondente

Nesta seção foram distribuídas quatro questões, sendo elas de um a quatro. Essas questões foram utilizadas visando reunir informações referentes ao perfil dos usuários respondentes.

Foram elaboradas três questões para a coleta de informações a respeito do participante quanto a sua formação acadêmica e experiência profissional na área. O tipo de pergunta aplicada para estas questões foram as de múltipla escolha assim poderia ser marcado apenas uma opção. As questões seguem descritas a seguir:

- 1 - Qual sua formação acadêmica?
  As opções de resposta desta questão incluíam: Técnico; Graduando/Graduado; Pós-Graduando/Pós-graduado; Mestrando/Mestre; Doutorando/Doutor; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;
- 2 - Há quanto tempo você exerce atividades na área computação?
  As opções de resposta desta questão incluíam: Não atuo; Um ano ou menos; Dois a quatro anos; Cinco a dez anos; Mais de dez anos;
- 3 - Qual cargo ocupa atua atualmente?
  As opções de resposta desta questão incluíam: Apoio Estratégico (Diretor, CEO, CIO); Apoio Gerencial (Gerente de projetos, gerente de qualidade, gerente de teste); Apoio Operacional (Desenvolvedor, analista de sistemas, webdesigner, analista de testes); Estudante; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;

A questão quatro tem o objetivo de coletar a informação sobre sua experiência com desenvolvimento de projetos em PHP. O tipo de pergunta aplicada para esta questão foi a de múltipla escolha, assim poderia ser marcado apenas uma opção. A questão segue descrita a seguir:

- 4 - Você já participou do desenvolvimento de algum projeto em linguagem PHP sendo profissionalmente ou não?
  As opções de resposta desta questão incluíam: Sim; Não;

Caso o entrevistado não possua experiência alguma profissionalmente com o desenvolvimento de projetos em PHP, o mesmo é direcionado a página de agradecimentos do questionário finalizando sua participação.

3.3.1.3 Terceira Seção: Experiência em projetos PHP

Nesta seção foram distribuídas cinco questões, sendo elas da cinco a nove. Elas são voltadas para a coleta de dados sobre a experiência do entrevistado com o desenvolvimento de projetos em PHP.

As questões cinco e seis, foram elaboradas com o objetivo de identificar a experiência do entrevistado com o desenvolvimento de projetos em PHP, analisando o tempo de experiência e quantas pessoas em média estavam envolvidas nos projetos que participou. O tipo de pergunta aplicada para estas questões foram as de múltipla escolha assim poderia ser marcado apenas uma opção. As questões seguem descritas a seguir:

- 5 - A quanto tempo em média participa de projetos em PHP?
  As opções de resposta desta questão incluíam: Não atuo; Um ano ou menos; Dois a quatro anos; Cinco a dez anos; Mais de dez anos;
- 6 - Quantas pessoas em média participaram dos projetos que participou?
  As opções de resposta desta questão incluíam: Apenas eu; De duas a quatro pessoas; De cinco a oito pessoas; Mais de oito pessoas;

A questão sete foi montada a partir dos conhecimentos de Peters e Pedrycz (2001) e Sommerville (2011), onde segundo eles a manutenção de *software* possui três categorias, Manutenção Corretiva, Manutenção Adaptativa, Manutenção de Aperfeiçoamento. O tipo de pergunta aplicada para esta questão foi a de múltipla escolha, assim poderia ser marcado apenas uma opção. A questão segue descrita a seguir:

- 7 - Dentre as correções do projeto, qual era a que mais teve o costume de realizar?
  As opções de resposta desta questão incluíam: Manutenção Corretiva; Manutenção Adaptativa; Manutenção de Aperfeiçoamento; Não realizo manutenção em produtos de *software*; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;

A pergunta oito possuía o objetivo identificar com os frameworks que o entrevistado já utilizou, assim poderia se ter informação sobre sua experiência com frameworks. O tipo de pergunta aplicada para esta questão foi a de caixas de seleção, assim poderia ser marcado mais de uma opção. A questão segue descrita a seguir:

- 8 - Quais Frameworks, já utilizou em seus projetos?
  As opções de resposta desta questão incluíam: CakePHP; CodeIgniter; Laravel; Limonade; Lumen; Phalcon; Silex; Slim; Symfony; Wave; Zend Explessive; Zend Framework; Não sei opinar; Não usava; Outros;

A questão nove possuía o objetivo verificar se o entrevistado já trabalhou em um projeto utilizando o Laravel Framework, sendo profissional ou não. Ela também é utilizada para separar os entrevistados pelo conhecimento sobre os assuntos tratados na próxima seção. O tipo de pergunta aplicada para esta questão foi a de múltipla escolha, assim poderia ser marcado apenas uma opção. A questão segue descrita a seguir:

- Você já utilizou o Framework Laravel em algum projeto sendo profissionalmente ou não?
  As opções de resposta desta questão incluíam: Sim; Não;

Caso o entrevistado marque que não utilizou o Laravel em projeto algum, o mesmo é direcionado a página de agradecimentos do questionário finalizando sua participação.

3.3.1.4 Quarta Seção: Laravel Framework

Nesta seção foram distribuídas cinco questões, sendo elas da dez a quatorze. Elas procuram coletar informações quanto a experiência do entrevistado com o Laravel Framework.

A questão dez possuía o objetivo identificar possíveis motivos que levaram o usuário a utilizar o Laravel, sua montagem foi feita a partir dos conhecimentos de Silva M. (2016), documentação do Framework, e do site oficial do Framework. O tipo de pergunta aplicada para esta questão foi a de caixas de seleção, assim poderia ser marcado mais de uma opção. A questão segue descrita a seguir:

- 10 - Quais motivos o levaram a utilizar o Laravel Framework:
  As opções de resposta desta questão incluíam: Popularidade do Framework; Comunidade maior que pode auxilia na utilização; Facilidade de aprendizado na sua utilização; Facilidade no reaproveitamento de código Fato do framework ser Livre; Organização, estrutura de código e arquivos; Possuir documentação de fácil acesso e entendível; Recomendação de amigos ou empresa; Relacionamento de entidades; Framework com suporte para teste; Utilização de Sistema de Templates; Utilização de arquivo de rotas; Utilização do QueryBuilder; Utilização de interações via linha de comando; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;

A questão onze possui a finalidade de identificar o tempo em que o entrevistado possui experiência com o framework. O tipo de pergunta aplicada para esta questão foi a de múltipla escolha, assim poderia ser marcado apenas uma opção. A questão segue descrita a seguir:

- 11 - Há quanto tempo conhece a ferramenta Laravel Framework?
  As opções de resposta desta questão incluíam: Um ano ou menos; Dois a quatro anos; Cinco a dez anos; Mais de dez anos;

A questão doze possui o objetivo de identificar a frequência média de utilização do framework pelo entrevistado. O tipo de pergunta aplicada para esta questão foi a de escala linear, quanto maior o número maior a média de utilização. A questão segue descrita a seguir:

- 12 - Ne uma escala de 1 a 10, com que frequência utiliza o Laravel Framework?
  As opções de resposta desta questão incluíam: uma linha horizontal de um a dez.

A questão treze tem a finalidade de tentar entender o quão difícil é o processo de aprendizado da ferramenta para os entrevistados. O tipo de pergunta aplicada para esta questão foi a de escala linear, quanto maior o número maior a dificuldade encontrada no processo de aprendizagem. A questão segue descrita a seguir:

- 13 - Na sua opinião, qual o nível de dificuldade encontrado no processo de aprendizado do Framework?
- As opções de resposta desta questão incluíam: uma linha horizontal de um a dez.

A questão quatorze procura verificar se o entrevistado encontrou outro framework que se demonstrou mais adequado, o levando a deixar de utilizar o Laravel. O tipo de pergunta aplicada para esta questão foi a de múltipla escolha, assim poderia ser marcado apenas uma opção. A questão segue descrita a seguir:

- 14 - Houve algum framework que têm se demonstrado mais adequado em relação aos projetos com os quais têm trabalhado? O qual tenha te motivado a deixar de utilizar o Laravel.
  As opções de resposta desta questão incluíam: Sim; Não;

Caso o entrevistado marque que não, o mesmo é direcionado a sexta seção continuando a sequência de perguntas somente sobre o Laravel, caso a resposta seja sim, o mesmo segue para a próxima seção, onde possuem perguntas que buscam identificar o framework, e avaliá-lo da mesma forma feita na seção atual.

3.3.1.5 Quinta seção: Outro Framework

Nesta seção foram distribuídas seis perguntas, sendo elas de quinze a vinte. Elas possuíam o objetivo coletar informações sobre a experiência com o framework citado na seção anterior.

A questão quinze procura identificar o framework que o entrevistado citou na seção anterior, pergunta quatorze. O tipo de pergunta aplicada para esta questão foi a de múltipla escolha, assim poderia ser marcado apenas uma opção. A questão segue descrita a seguir:

- 15 - Qual o Framework utilizado?
  As opções de resposta desta questão incluíam: CakePHP; CodeIgniter; Laravel; Limonade; Lumen; Phalcon; Silex; Slim; Symfony; Wave; Zend Explessive; Zend Framework; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;

As questões de dezesseis a dezenove foram adaptadas das perguntas da quarta seção de maneira genérica, elas possuem os mesmos objetivos, e estrutura muito similar. Os tipos de perguntas aplicadas também são do mesmo modelo da quarta seção. A questão segue descrita a seguir:

- 16 - Quais motivos o levaram a utilizar do Framework:
  As opções de resposta desta questão incluíam: Popularidade do Framework; Comunidade maior que pode auxilia na utilização; Facilidade de aprendizado na sua utilização; Facilidade no reaproveitamento de código Fato do framework ser Livre; Organização, estrutura de código e arquivos; Possuir documentação de fácil acesso e entendível; Recomendação de amigos ou empresa; Relacionamento de entidades; Framework com suporte para teste; Utilização de Sistema de Templates; Utilização de arquivo de rotas; Utilização do QueryBuilder; Utilização de interações via linha de comando; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;
- 17 - Há quanto tempo conhece o Framework?
  As opções de resposta desta questão incluíam: Um ano ou menos; Dois a quatro anos; Cinco a dez anos; Mais de dez anos;

- 18 - Ne uma escala de 1 a 10, com que frequência utiliza este Framework?
  As opções de resposta desta questão incluíam: uma linha horizontal de um a dez.

- 19 - Na sua opinião, qual o nível de dificuldade encontrado no processo de aprendizado do framework?
  As opções de resposta desta questão incluíam: uma linha horizontal de um a dez.

A pergunta vinte utiliza os conceitos de Sommervile (2011), onde segundo ele um dos atributos essenciais para um *software* é a Manutenibilidade de *software*, que é a capacidade do produto de ser modificado para atender as necessidades de seus clientes, nela o entrevistado diz de o quanto o framework utilizado auxilia na obtenção deste atributo. O tipo de pergunta aplicada para esta questão foi a de escala linear, quanto maior o número mais o framework auxilia nesse na obtenção deste atributo. A questão segue descrita a seguir:

20 - Segundo Sommervile (2011) um dos atributos essenciais para um *software* é a Manutenibilidade, que é a capacidade do produto de ser modificado para atender as

necessidades de seus clientes. Na sua opinião em uma escala de 1 a 10 quanto este Framework auxilia na obtenção deste atributo?

As opções de resposta desta questão incluíam: uma linha horizontal de um a dez.

3.3.1.6 Sexta seção: Manutenibilidade de *Software*

Nesta seção foram distribuídas seis questões, sendo elas da vinte e um a vinte e seis. Elas possuíam o objetivo coletar informações sobre o nível de preocupação do entrevistado durante o desenvolvimento de um *software* envolvendo Laravel framework no quesito manutenibilidade.

A pergunta vinte e um procura identificar o nível de familiarização do entrevistado quanto a manutenção de *software*. O tipo de pergunta aplicada para esta questão foi a de escala linear, quanto maior o número maior o nível de conhecimento do entrevistado quanto a manutenibilidade de *software*. A questão segue descrita a seguir:

- 21 - De modo geral, como você considera seu nível de conhecimento acerca de manutenção de *software*?
  As opções de resposta desta questão incluíam: uma linha horizontal de um a dez.

As questões de vinte e dois a vinte seis, foram elaboradas com base na ISO 9126, que sita como subatributos da manutenibilidade de *software*: Analisabilidade; Modificabilidade; Estabilidade; Testabilidade; Conformidade relacionada à manutenibilidade. As perguntas criadas possuíam o objetivo identificar o nível de preocupação do entrevistado com o desenvolvimento de *softwares* com os atributos citados. O tipo de pergunta escolhido para essa coleta é múltipla escolha, assim poderia ser marcado apenas uma opção para cada pergunta. As questões seguem descritas a seguir:

- 22 - Com que frequência é considerada a preocupação do desenvolvimento de um sistema com analisabilidade?
  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Nunca; Outro;
- 23 - Com que frequência é considerada a preocupação do desenvolvimento de um sistema com modificabilidade?
  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Nunca; Outro;
- 24 - Com que frequência é considerada a preocupação do desenvolvimento de um sistema com estabilidade?
  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Nunca; Outro;
- 25 - Com que frequência é considerada a preocupação do desenvolvimento de um sistema com testabilidade?
  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Nunca; Outro;
- 26 - Com que frequência é considerada a preocupação do desenvolvimento de um sistema com conformidade relacionada à manutenibilidade?
  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Nunca; Outro;

3.3.1.7 Sétima seção: Laravel Framework e Manutenibilidade

Esta é a última seção de perguntas, ela possui seis questões, numeradas de vinte e sete a trinta e dois. As questões desta seção possuíam o objetivo de coletar informações acerca da experiência com o Laravel framework.

As perguntas de vinte e sete a trinta e um foram criadas a partir das informações disponibilizadas na documentação do framework. São perguntas onde foram relacionadas funcionalidades do framework com os subatributos citados na ISO 9126. O tipo de pergunta escolhido para essa coleta é múltipla escolha, assim poderia ser marcado apenas uma opção para cada pergunta. As questões seguem descritas a seguir:

- 27 - O fato do Laravel ter recursos como o Artisan Console que buscam auxiliar no desenvolvimento de sistemas, estabelecendo uma padronização e organização na criação de arquivos, pode-se dizer que ele auxilia na analisabilidade de código-fonte?

  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Não sei opinar; Nunca; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;
- 28 - A utilização de recursos como arquivo de rotas e templates ou blades no Laravel auxilia no processo de reutilização de código-fonte?

  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Não sei opinar; Nunca; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;
- 29 - O fato do Laravel ser construído com suporte para testes com o PHPUnit, auxilia na estabilidade e testabilidade de recursos do sistema evitando assim efeitos inesperados decorrentes de modificações no *software*?

  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Não sei opinar; Nunca; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;
- 30 - pode-se dizer que a utilização do Eloquent no Laravel auxilia no processo de implementação e manutenção de código-fonte?

  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Não sei opinar; Nunca; Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;
- 31 - pode-se dizer que a utilização das Migrations, auxilia na construção, modificação e controle de versão do banco de dados?

  As opções de resposta desta questão incluíam: Em todos os casos; Na maioria dos casos; Eventualmente; Raramente; Não sei opinar; Nunca;

Outros... é uma opção de texto para caso as anteriores não definam o entrevistado;

A pergunta trinta e dois segue a mesma lógica conceito da pergunta 20 aplicada na quinta seção. A questão segue descrita a seguir:

- 32 - Segundo Sommervile (2011) um dos atributos essenciais para um *software* é a Manutenibilidade, que é a capacidade do produto de ser modificado para atender as necessidades de seus clientes. Na sua opinião em uma escala de 1 a 10 quanto este Framework auxilia na obtenção deste atributo?
  As opções de resposta desta questão incluíam: uma linha horizontal de um a dez.

3.3.1.8 Oitava Seção: Agradecimentos

Nesta seção foram feitos agradecimentos aos participantes pela participação e realização do envio do questionário para o Google Forms.

3.4 COLETA E TRATAMENTO DE DADOS

O questionário ficou disponível a partir do dia 24 de outubro de 2018 até o dia 16 de novembro de 2018 as 23 horas. Neste período foram coletadas 104 respostas.

O Google Forms disponibiliza na própria ferramenta uma análise simples das respostas obtidas com as perguntas em forma de gráficos, esses gráficos auxiliam o pesquisador a analisar e acompanhar a trajetória dos resultados de sua pesquisa, porém devido á simplicidade dos mesmos, viu-se que seria de maior valor realizar uma análise mais detalhada dos resultados.

Desta forma, para realizar uma análise mais detalhada foi utilizada a ferramenta Power BI, uma ferramenta de análise de dados desenvolvida pela

Microsoft em 2016. A ferramenta permite relacionar as perguntas e gerar gráficos a partir destas associações, proporcionando assim um resultado mais interessante para análise.

Para alimentar a ferramenta Power BI foi necessário realizar uma exportação dos dados da ferramenta Google Forms, a exportação dos dados é feita de maneira automática e em forma de planilha criada no Google *Sheets,* que através de um mecanismo de publicação da tabela foi importada via Web para o Power BI.

4 RESULTADOS

Nesta sessão serão apresentados os resultados obtidos através do questionário aplicado anteriormente.

Para a apresentar os dados obtidos de forma clara os resultados serão divididos em subcapítulos referentes a cada sessão de pergunta do questionário. Os dados serão apresentados utilizando gráficos que estarão acompanhados de um texto Explicando os resultados obtidos.

4.1 Identificação do Respondente

A primeira seção teve como objetivo coletar experiência sobre os participantes, conhecendo um pouco sobre sua experiência profissional e atuação.

A primeira pergunta teve como objetivo identificar qual era a formação dos participantes para se conhecer o nível de conhecimento acadêmico, com isso pode-se observar que, dentre os 104 participantes cerca de 2 dos entrevistados (2,5%) eram Mestrandos ou Mestres, exatamente 22 dos entrevistados (21,15%) eram Pós-Graduandos ou Pós-graduados, cerca de 62 dos entrevistados (59,61%) eram Graduandos ou Graduados, exatamente 16 dos entrevistados (15,38%) eram Técnicos e para os que preencheram o tipo outros, exatamente 1 entrevistado (0,96% dos entrevistados) se declarou como Autodidata e 0,96% exatamente 1 entrevistado (0,96%) se declarou como Ensino Médio. A partir deste resultado pode-se observar que a maioria dos entrevistados possuíam alguma formação acadêmica, começando do técnico ao mestre. Seguem estas informações no Gráfico.

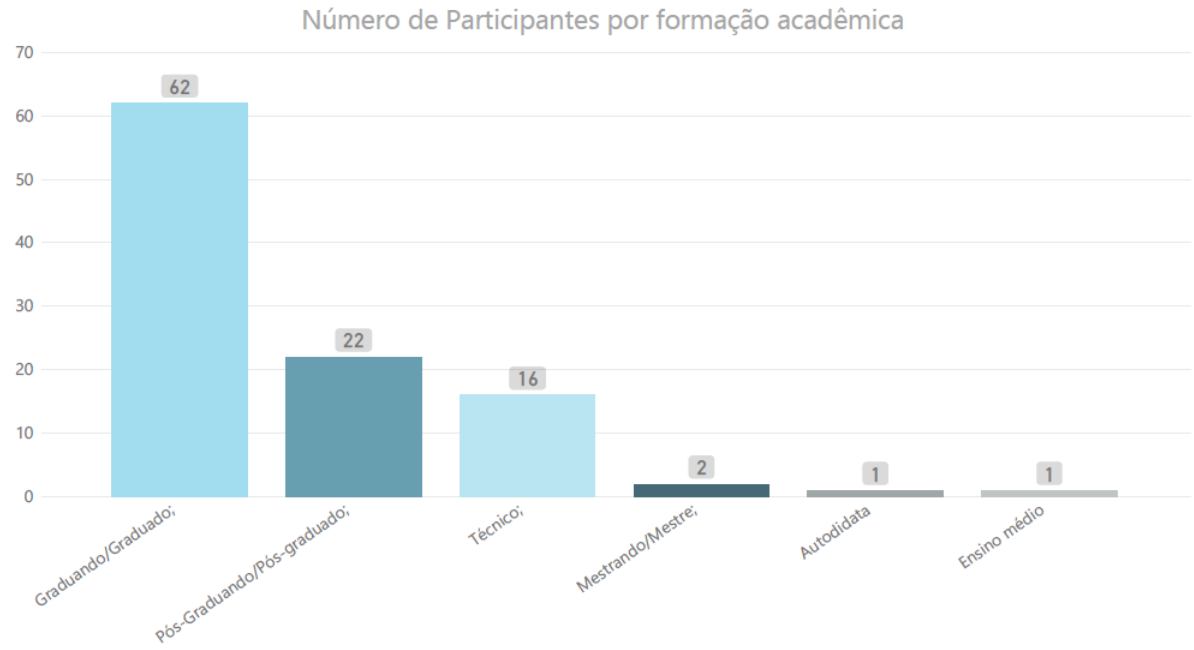


Gráfico 1 – Formação acadêmica dos entrevistados.
Fonte: Próprio autor.

Em relação ao tempo de experiência dos entrevistados na área de computação (TI), cerca de 15 dos entrevistados (14,22%) declaram trabalhar mais de dez anos, exatamente 32 dos entrevistados (30,76%) afirmam que trabalham entre cinco e dez anos, exatamente 36 dos entrevistados (34,61%) alegam trabalhar a cerca de dois a quatro anos, exatos 19 entrevistados (18,26%) declararam que trabalham a um ano ou menos na área e outros 2 dos entrevistados (1,92%) disseram não atuar ainda na área. Visto que a maior parte dos entrevistados tem pelo menos dois anos de experiência, isso demonstra que a maior parte dos entrevistados possui já uma bom tempo de atuação, comprovando que possuem uma boa experiência na área. Seguem estas informações no Gráfico 2.

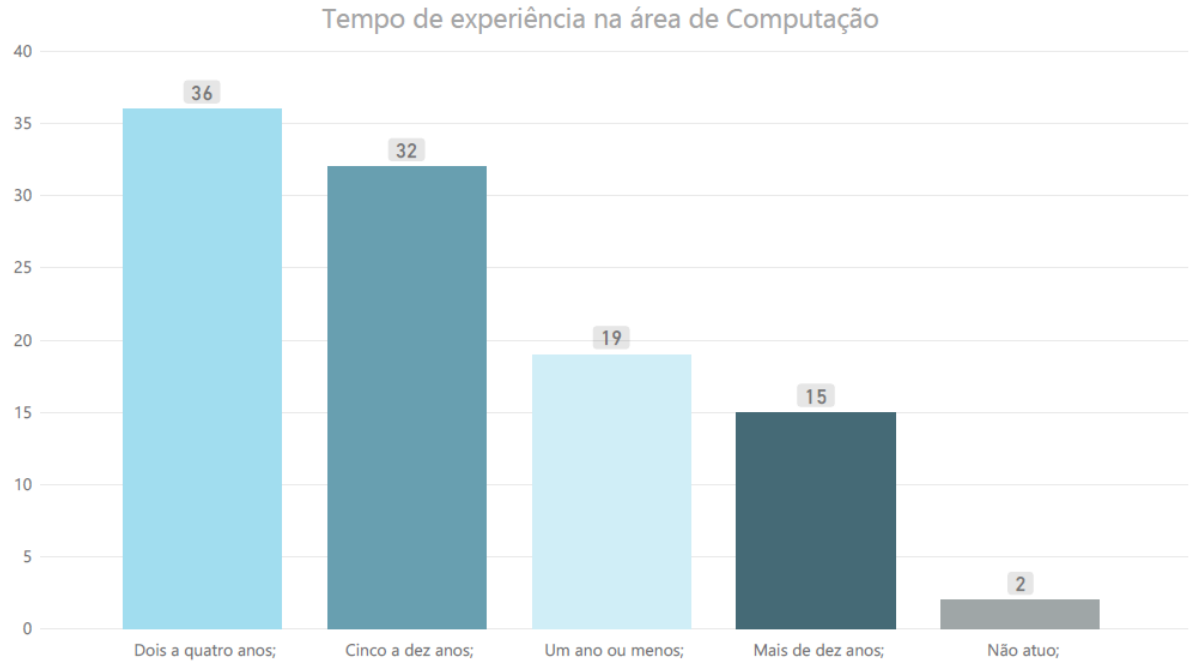


Gráfico 2 – Tempo de experiência dos entrevistados.
Fonte: Próprio autor.

Em relação a sua ocupação atual dos entrevistados, cerca de 4 dos entrevistados (3,84%), declararam atuar na área de Apoio Estratégico (Diretor, CEO, CIO), cerca de 10 dos entrevistados (9,61%), afirmaram atuar na área de Apoio Gerencial (Gerente de projetos, gerente de qualidade, gerente de teste), 84 dos entrevistados (80,76%) declararam, atuar na área de Apoio Operacional (Desenvolvedor, analista de sistemas, webdesigner, analista de testes), exatamente 4 dos entrevistados (3,84%) afirmaram trabalhar em outras áreas e cerca de 2 dos entrevistados (1,92%) afirmaram serem estudantes. A partir destas respostas foi possível identificar que a maioria dos entrevistados eram pessoas que participam de forma direta no ambiente de produção, isso pode ser percebido por 80,76% dos entrevistados serem do Apoio Operacional e 9,61% serem do Apoio Gerencial, duas áreas que atuam de forma direta na produção de software. Seguem estas informações no Gráfico 3.

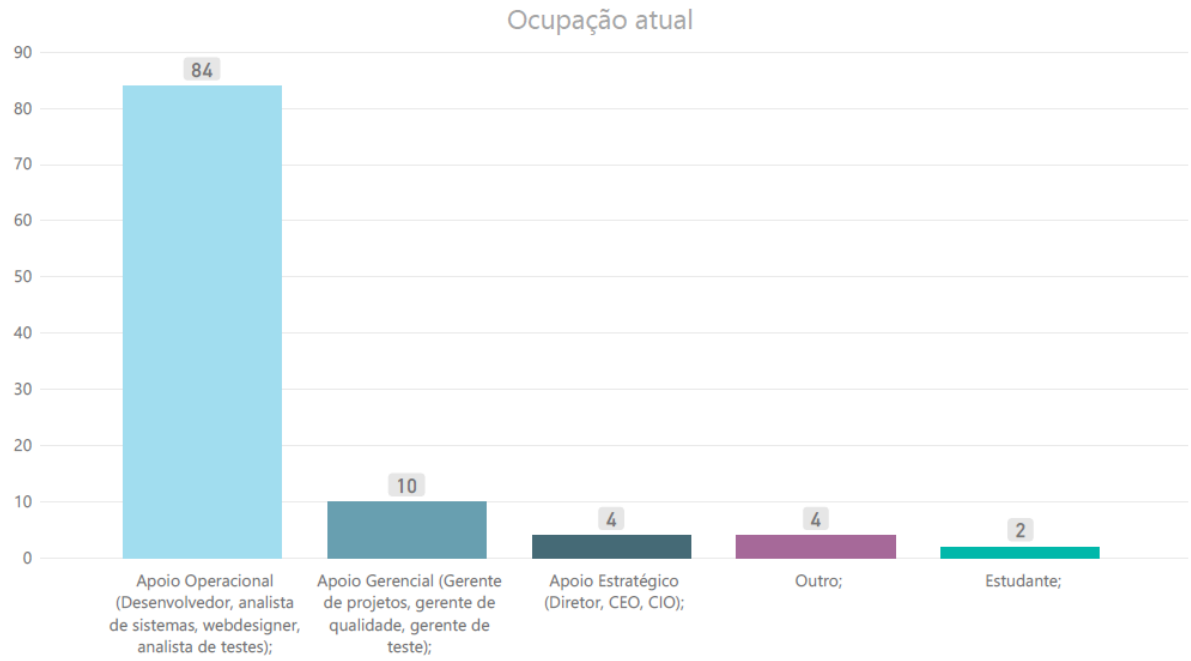


Gráfico 3 – Cargos que ocupam os entrevistados atualmente.
Fonte: Próprio autor.

Para filtrar os participantes foi elaborada uma questão nesta sessão para identificar se o entrevistado já havia participado de projetos na linguagem PHP, a pesquisa revelou que, cerca de 74 dos entrevistados (93,67%) já tinham experiência em projetos PHP outros 4 dos entrevistados (6,33%) não tinham experiência. Este filtro demonstrou que, a maior parte dos entrevistados já tinham algum conhecimento em projetos PHP o que demonstra que seus conhecimentos são uteis na seção seguinte. Seguem estas informações no Gráfico 4.

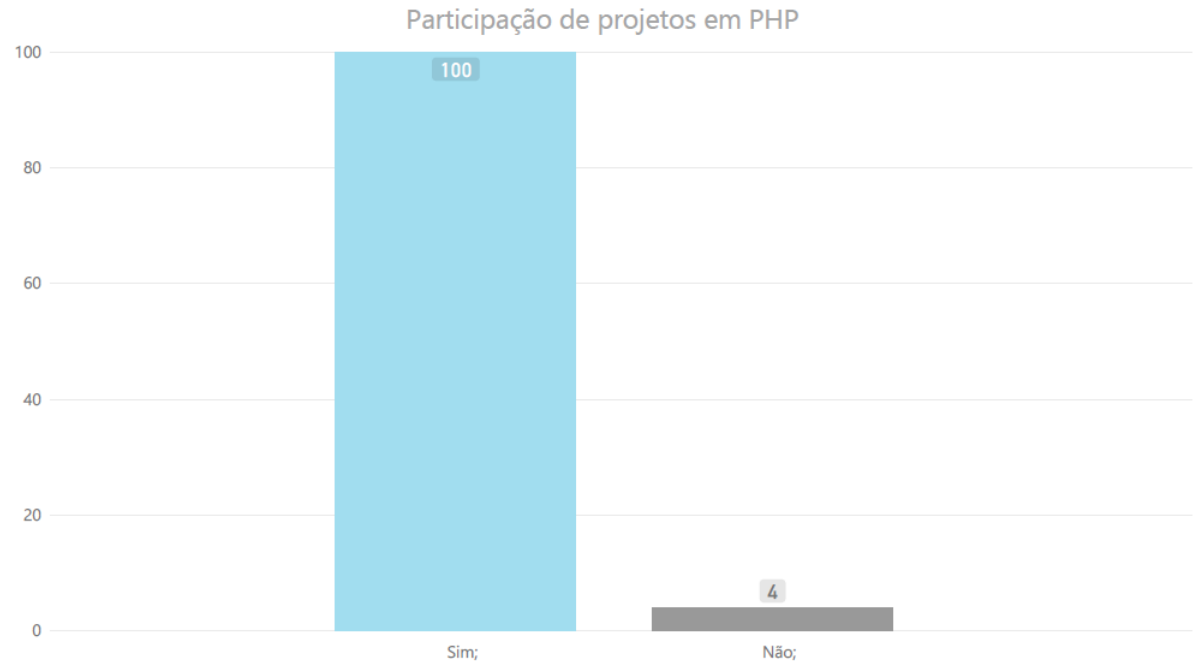


Gráfico 4 – Entrevistados que participaram de desenvolvimento de algum projeto em linguagem PHP.
Fonte: Próprio autor.

A partir desta última questão a quantidade de entrevistados cai, devido a filtragem feita, os entrevistados que não conhecem sobre projetos PHP são direcionados ao final do questionário, dentre os 104 participantes, 100 continuaram, apenas 3 Graduandos ou Graduados e 1 de ensino médio foram direcionados para o final do questionário. Através deste gráfico que relaciona a formação acadêmica com os entrevistados que possuem conhecimentos com desenvolvimento PHP pode ser observado que, a maior parte dos entrevistados possuem experiência na era, deixando apenas três Graduandos/Graduados e um do Ensino Médio sem participar da próxima seção. Segue a relação dos que vão continuar relacionadas a formação acadêmica no Gráfico 5.

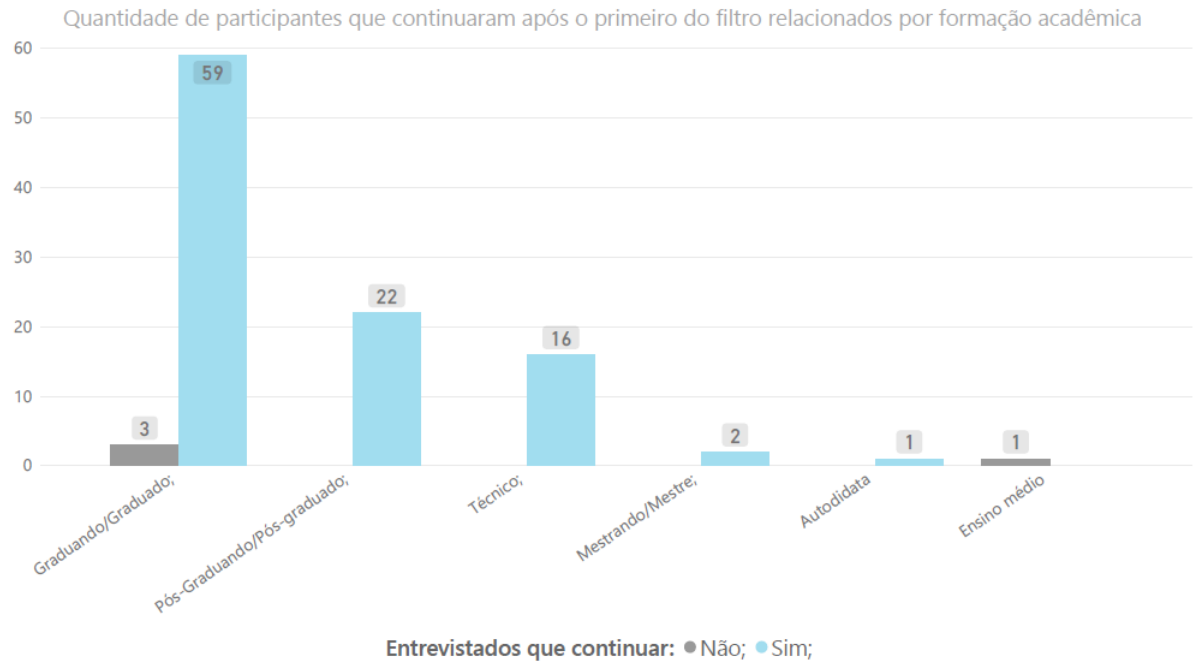


Gráfico 5 - Relação dos que entrevistados que continuaram na pesquisa após o filtro da primeira seção.
Fonte: Próprio autor.

4.2 Experiência em projetos PHP

A segunda seção teve como objetivo coletar informações sobre suas experiências com a linguagem PHP.

Os valores a seguir serão apresentados usando apenas a porcentagem pois a quantidade de entrevistados são exatos 100.

Sobre média de idade dos projetos trabalhados pelos entrevistados, dos 100 entrevistados que passaram para esta seção, 1% afirmaram trabalhar com projetos com mais de dez anos de idade, 9% dos entrevistados declararam trabalhar em projetos com média de cinco a dez anos de idade, 59% dos entrevistados alegaram trabalhar em projetos com média de idade de dois a quatro anos e outros 31% revelaram que trabalham com projeto de média de idade de um ano ou menos. Com este resultado pode-se demonstrar que a maioria dos entrevistados possui experiência com projetos relativamente novos, isso pode ser percebido devido a apenas 10 deles trabalharem com projetos mais com mais de 5 anos, isso pode

indicar que seus conhecimentos não estão limitados a um desenvolvimento fechado a novas tecnologias, tendo assim maior possibilidade de conhecimento sobre os frameworks ou novos recursos que possam ser importantes na manutenibilidade. Estas informações seguem no Gráfico 6.

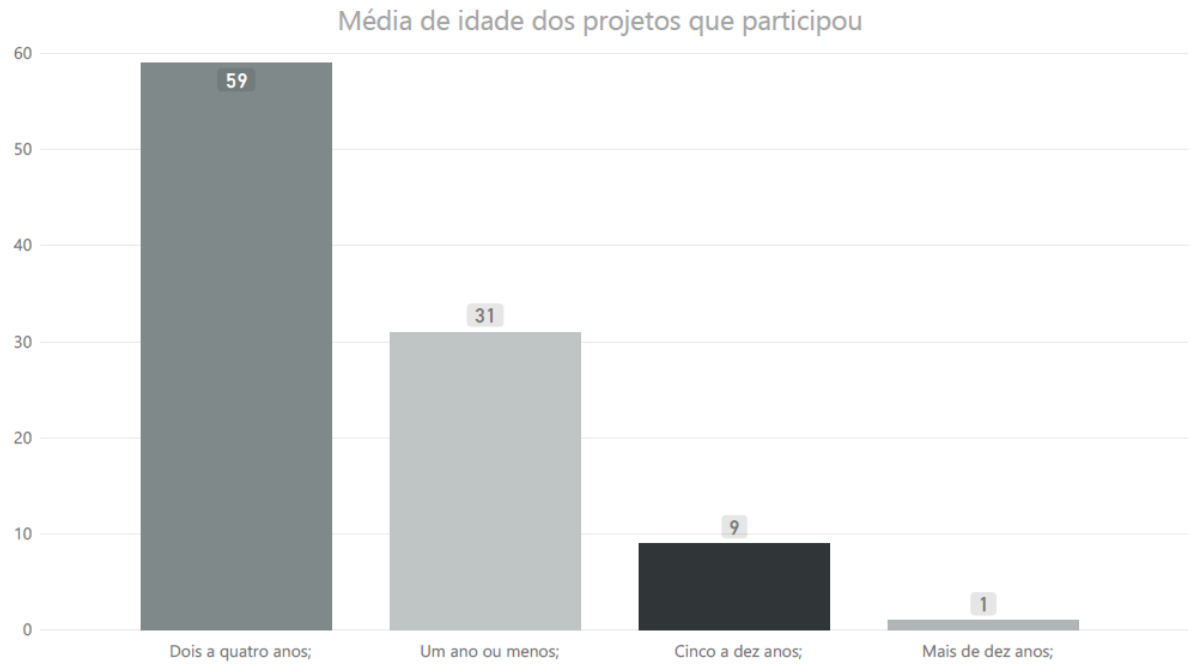


Gráfico 6 - Média de idade dos projetos que participaram.
Fonte: Próprio autor.

Com relação à média de pessoas que participaram dos projetos se obteve o seguinte resultado, 9% afirmaram que a média de pessoas em sua equipe era acima de oito pessoas, 15% alegaram a média de participantes de sua equipe era de cinco a seis, 70% dos entrevistados declararam que sua equipe era em média de duas a quatro pessoas e outros 6% afirmaram geralmente trabalhavam sozinhos. Através desta pergunta, é possível identificar se os entrevistados tem o costume de trabalhar de grupo, muitos frameworks auxiliam no desenvolvimento de sistemas em grupo, e pelo resultado coletado a maior parte dos entrevistados trabalham no geral em grupo, apenas 6 disse trabalhar sozinho. Estas informações seguem no Gráfico 7.

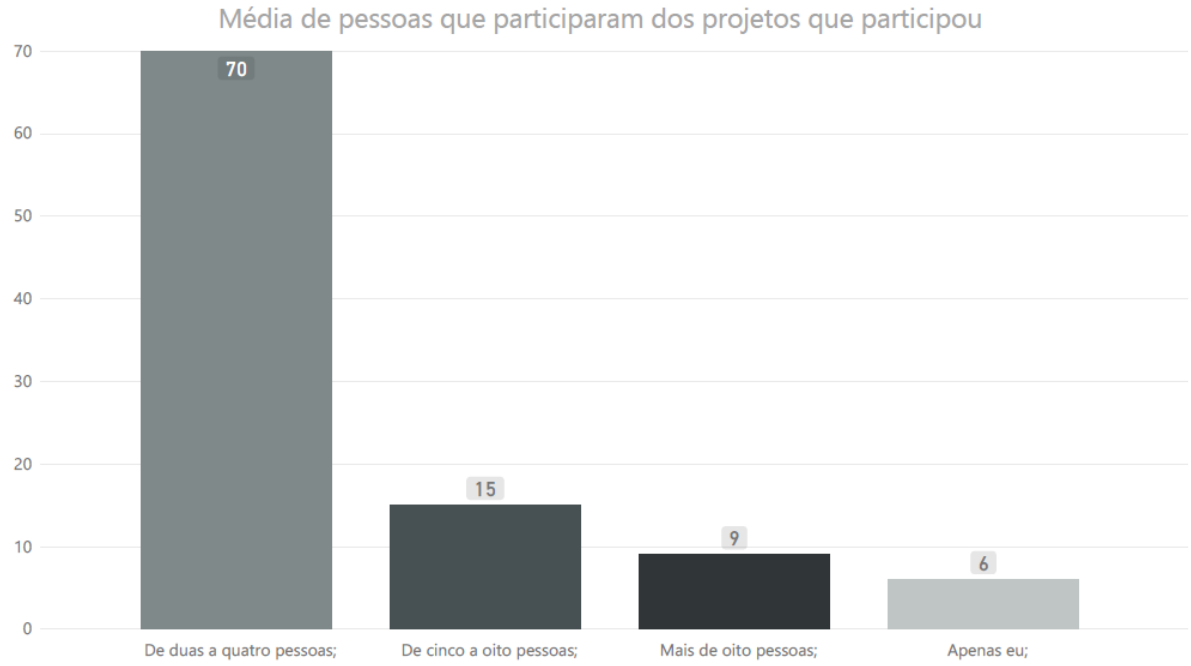


Gráfico 7 – Média de pessoas que participaram dos projetos.
Fonte: Próprio autor.

Visto que as informações com relação aos tipos de correção mais executadas, usadas neste trabalho são antigas, esta pergunta tem como objetivo trazer novos dados a respeito do assunto, através disso pode-se identificar que dentre os 100 participantes, 40% dos entrevistados afirmaram realizar mais manutenções de aperfeiçoamento, 34% declararam realizar mais manutenções corretivas, 22% revelaram realizar mais manutenções adaptativas, 1% alegaram não realizar manutenções e outros 3% declararam que realizam manutenções diferentes das mencionadas. Estas informações seguem no Gráfico 8.

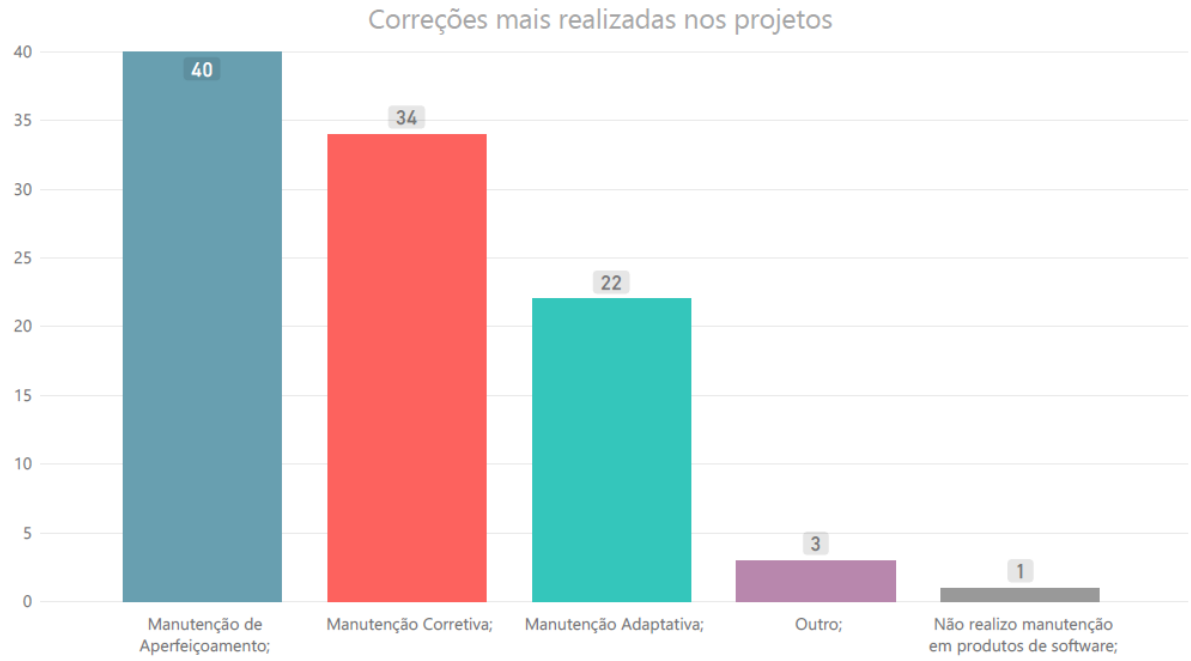


Gráfico 8 - Correções do projeto, com mais costume de se realizar.
Fonte: Próprio autor.

Para o entendimento da pesquisa dos frameworks utilizados pelos entrevistados, é importante entender que foi feita em múltipla escolha, o que significa que os entrevistados puderam selecionar mais de uma resposta, através desta questão obteve-se o seguinte resultado, 89 dos entrevistados afirmaram já terem utilizado o Laravel, 30 declaram terem utilizado o Lumen, 24 marcaram que já utilizaram o CakePHP, 24 declararam utilizar o Zend Framework, 20 informaram que já utilizaram o CodeIgniter, 19 já utilizaram o Slim, 17 alegaram já terem utilizado o Symfony, 13 informaram que já utilizaram o Phalcon, 9 informaram que utilizaram o Silex, 6 utilizaram Zend Explessive, 1 já utilizou o Limonade, 10 informaram utilizar frameworks não listados e 6 afirmara que não utilizaram. Com estas informações pode-se perceber que o Laravel é um dos frameworks mais conhecidos, visto que 89% dos participantes afirmaram conhecer ele, em sequência Lumen, CakePHP e Zend Framework foram os outros mais conhecidos. Estas informações seguem no Gráfico 9.

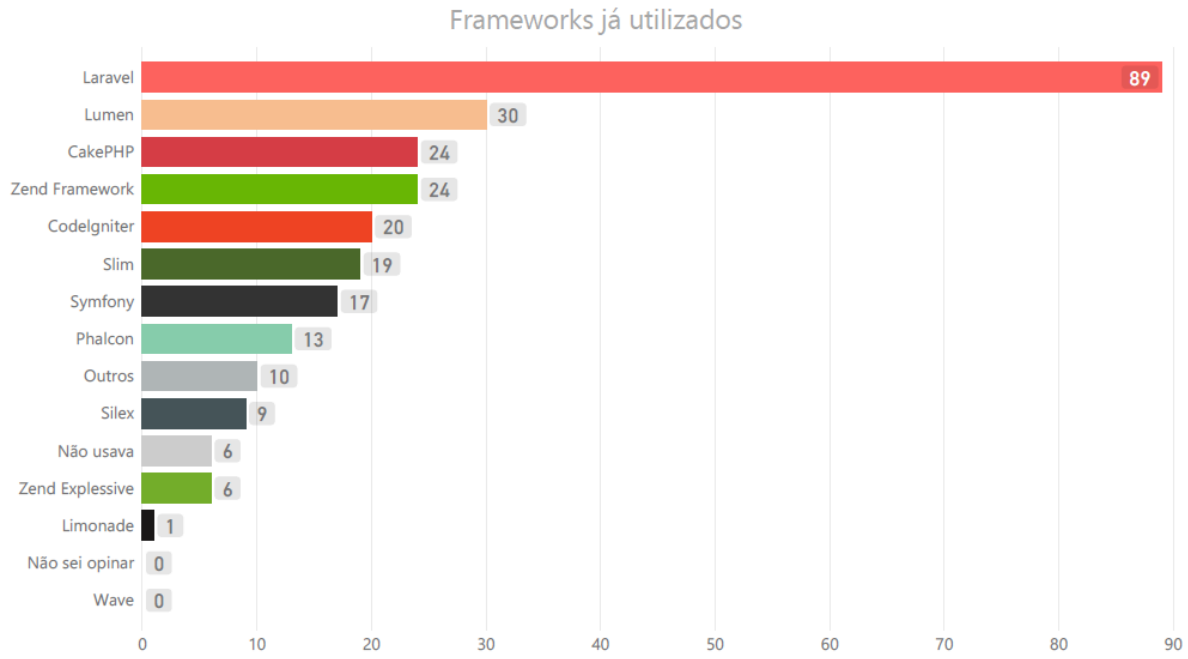


Gráfico 9 – Frameworks já utilizados pelos entrevistados.
Fonte: Próprio autor.

A última pergunta da seção teve como objetivo novamente realizar um filtro para identificar se o entrevistado teria conhecimento dos assuntos tratados nas seguintes seções, o resultado foi que, 90% dos entrevistados responderam que já utilizaram o Laravel, os outros 10% não utilizaram. Estas informações seguem no Gráfico 10.

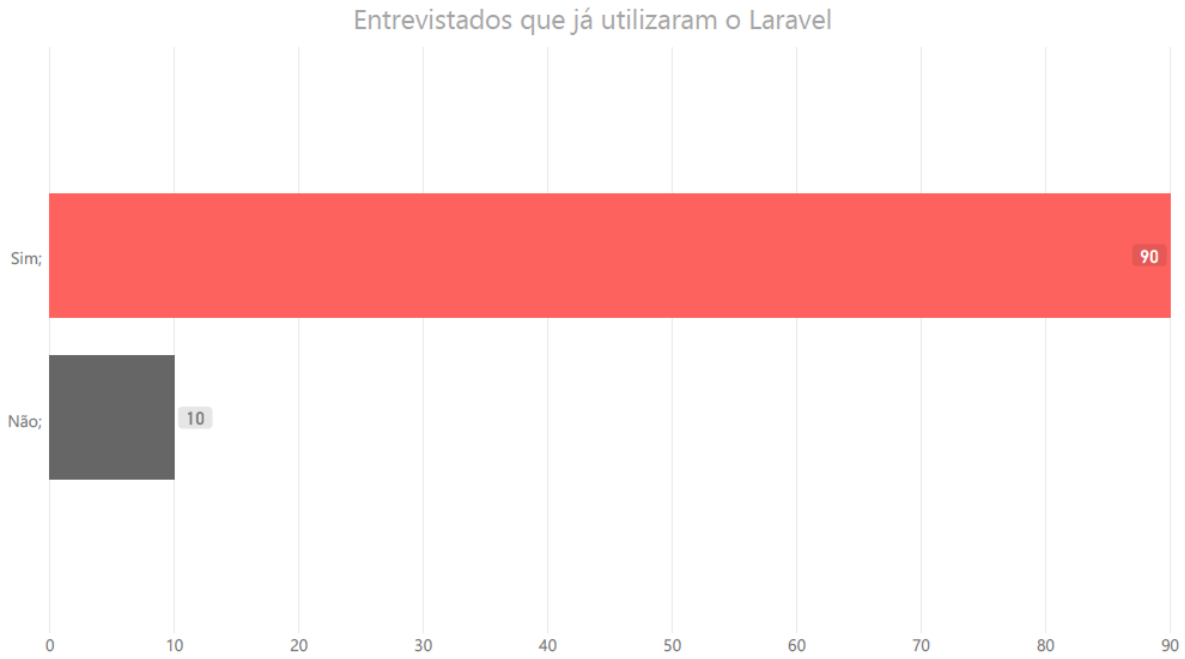


Gráfico 10 - Já utilizou o Framework Laravel.
Fonte: Próprio autor.

A partir desta última questão a quantidade de entrevistados cai novamente, devido a filtragem feita, os entrevistados que possuem conhecimento nenhum sobre o Laravel Framework são direcionados ao final do questionário, dentre os 100 participantes, 90 continuaram, 7 Graduandos ou Graduados, 2 Pós-Graduandos ou Pós-graduados e 1 Técnico foram direcionados para o final do questionário. Segue a relação dos que vão continuar relacionadas a formação acadêmica no Gráfico 11.

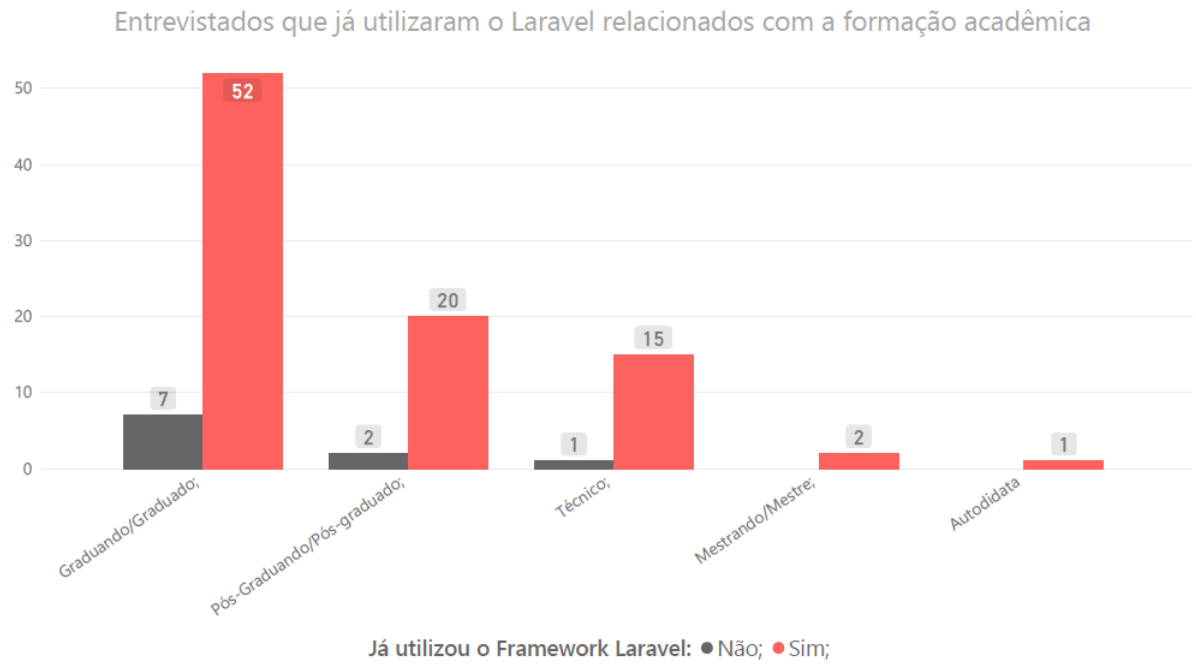


Gráfico 11 - Relação dos que entrevistados que continuaram na pesquisa após o filtro da segunda seção.
Fonte: Próprio autor.

4.3 Laravel Framework

Esta seção, é coletou informações com relação a experiência do entrevistado com o framework.

Nesta seção, foi identificado uma possível confusão de um entrevistado na resposta mandada para a primeira questão, onde ao solicitar que marcasse um dos motivos da utilização do Laravel ele selecionou a opção outros e escreveu o nome de outro framework sem qualquer explicação na resposta do motivo ou qualquer informação do tipo, portanto sua resposta será desconsiderada na primeira questão e omitida.

Na primeira questão apresentada nesta seção, buscou-se entender o que motivou os entrevistados a utilizarem o Laravel, nessa questão as respostas eram de múltipla escolha, o que significa que os entrevistados puderam selecionar mais de uma resposta, através desta questão obteve-se o seguinte resultado, dos 90 participantes, 76 (84,44% dos entrevistados) marcaram que utilizaram o Laravel pela sua facilidade de aprendizado, 73 (81,11% dos entrevistados) marcaram que utilizaram o Laravel por sua documentação ser de fácil acesso e entendível, 44

(48,88% dos entrevistados) declararam que utilizaram o Laravel por sua facilidade no reaproveitamento de código, 40 (44,44% dos entrevistados) marcaram que utilizaram por sua comunidade ser maior que auxilia na utilização, 39 (44,33% dos entrevistados) afirmaram que utilizaram por sua popularidade, 38 (42,22% dos entrevistados) marcaram que utilizaram o Laravel pelo fato do *software* ser livre, 38 (42,22% dos entrevistados) marcaram que utilizaram o Laravel pelo fato da utilização do QueryBuilder, 33 (36,66% dos entrevistados) declararam que utilizaram por sua organização e estrutura de código e arquivos, 31 (34,44% dos entrevistados) afirmaram que utilizaram o Laravel por suas interações via linha de comando, 30 (33,33% dos entrevistados) marcaram que utilizaram o Laravel por sua utilização de sistema de templates, 28 (31,11% dos entrevistados) afirmaram que utilizaram o Laravel por seus testes serem mais intuitivos, 23 (25,55% dos entrevistados) declararam que utilizaram o Laravel por sua utilização de arquivo de rotas, 20 (22,22% dos entrevistados) declararam que utilizaram o Laravel por recomendações de amigos ou empresa, 13 (14,44% dos entrevistados) marcaram que relacionamento de entidades, o restante dos resultados apresentados pela utilização do campo Outros onde o próprio usuário preenche, 1 (1,11% dos entrevistados) afirmaram que utilizaram o Laravel por utilizar facades, 1 (1,11% dos entrevistados) alegaram que utilizaram o Laravel por utilizar migrations, 1 (1,11% dos entrevistados) declararam que utilizaram o Laravel por utilizar Schedule Console, 1 (1,11% dos entrevistados) marcaram que utilizaram o Laravel por sua performance, 1 (1,11% dos entrevistados) marcaram que utilizaram o Laravel por poder ser usado com equipes pequenas, 1 (1,11% dos entrevistados) marcaram que utilizaram o Laravel por ser um *software* que já usava ele. Com isto pode-se perceber alguns dos recursos que podem ter atraído os desenvolvedores a utilizar o laravel. Estas informações seguem no Gráfico 12.

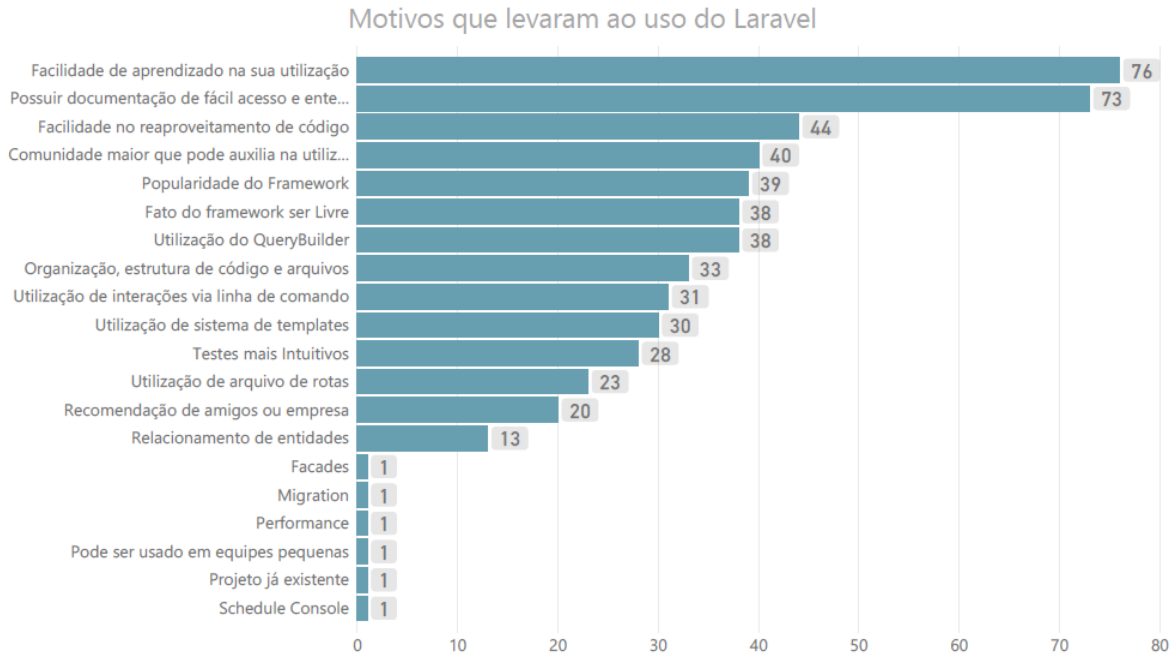


Gráfico 12 - Quais motivos o levaram a utilizar o Laravel Framework.
Fonte: Próprio autor.

Com relação ao tempo que os entrevistados conhecem o Laravel recebeu se o seguinte resultado, 9 (10%) dos entrevistados declararam que conheciam o Laravel a uma média de cinco a dez anos, outros 60 (66,67%) disseram que conheciam o Laravel a uma média de dois a quatro anos, outros 21 (23,33%) afirmaram conhecer a um ano ou menos. Com esta informação pode-se identificar um pouco da experiência de utilização dos entrevistados. Estas informações seguem no Gráfico 13.

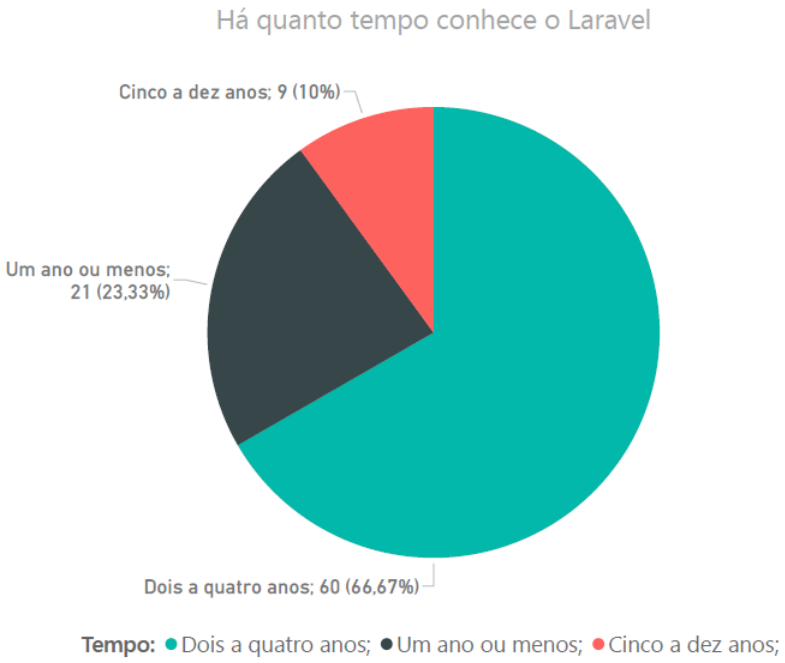


Gráfico 13 - Há quanto tempo conhece a ferramenta Laravel Framework.
Fonte: Próprio autor.

Quanto ao costume de uso do framework, foi preparado uma questão que coletava através de uma escala a frequência (escala de 1 a 10) de uso do framework, através dela se obteve o seguinte resultado, 20 (22,22%) dos entrevistados selecionaram como frequência dez, 14 (15,55%) dos entrevistados marcaram nove, 22 (24,44%) dos entrevistados optaram por selecionar 8, 16 dos entrevistados apontaram o nível de utilização como sete, 10 (11,11%) dos entrevistados escolheram seis, 4 (4,44%) dos entrevistados alegaram terem marcado cinco, 2 (2,22%) dos entrevistados apontaram três, 1 (1,11%) dos entrevistados marcou dois e 1 (1,11%) dos entrevistados assinalou um. Através de um cálculo feito somando as porcentagens é possível ver que 91,11% dos entrevistados assinalaram que utilizam o Laravel com nível 6 ou superior, o que demonstra que sua utilização é muito comum para os entrevistados. Estas informações seguem no Gráfico 14.

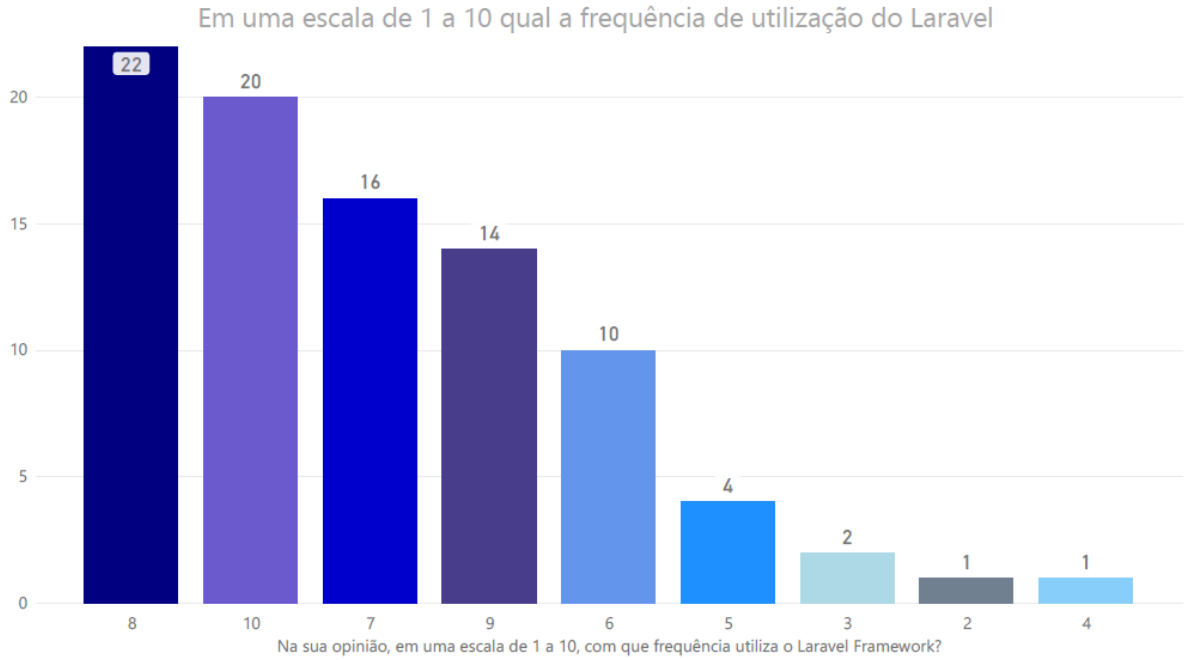


Gráfico 14 – Frequência de utilização do Laravel Framework.
Fonte: Próprio autor.

Com relação as dificuldades encontrada na aprendizagem, foi preparado uma questão que coletava através de uma escala a frequência (escala de 1 a 10) o nível de dificuldade encontrada, o resultado deu-se da seguinte forma, 28 (31,11%) dos entrevistados marcaram três como nível de dificuldade, 26 (28,88%) dos entrevistados marcaram 4 como nível de dificuldade, 9 (10%) dos entrevistados marcaram um como nível de dificuldade, 7 (7,77%) dos entrevistados marcaram cinco como nível de dificuldade, 6 (6,66%) dos entrevistados marcaram dois como nível de dificuldade, 6 (6,66%) dos entrevistados marcaram seis como nível de dificuldade, 6 (6,66%) dos entrevistados marcaram oito como nível de dificuldade, 1 (1,11%) dos entrevistados marcaram sete como nível de dificuldade e 1 (1,11%) dos entrevistados marcaram 9 como nível de dificuldade. Através de um cálculo onde se soma as porcentagens relacionadas a dificuldade, pode-se identificar que 84% dos entrevistados marcaram que o nível de dificuldade no processo de aprendizagem é nível 5 ou inferior, o que demonstra que os entrevistados tiveram pouca dificuldade durante os estudos da ferramenta. Estas informações seguem no Gráfico 15.

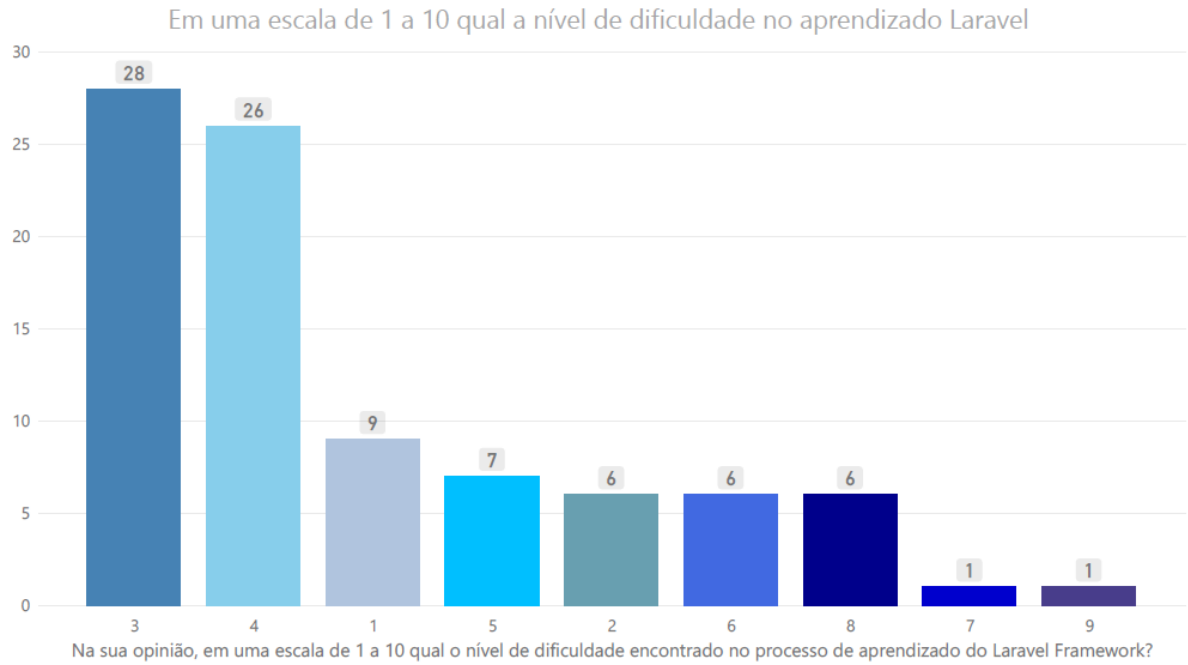


Gráfico 15 - Nível de dificuldade encontrado no processo de aprendizado do Framework.
Fonte: Próprio autor.

A última questão desta seção teve como objetivo analisar se os entrevistados encontraram outros frameworks que considerassem mais adequados que o Laravel, caso tivessem encontrado levaria a uma outra página do questionário onde seriam feitas algumas perguntas referentes a este framework. O resultado desta questão deu-se na seguinte forma, dos 90 entrevistados, 8 (8,89%) marcaram ter encontrado e foram direcionados para a seção cinco do questionário, outros 82 (91,11%) marcaram não e seguiram para a seção seis do questionário onde a pesquisa seguiu normalmente. Estas informações seguem no Gráfico 16.

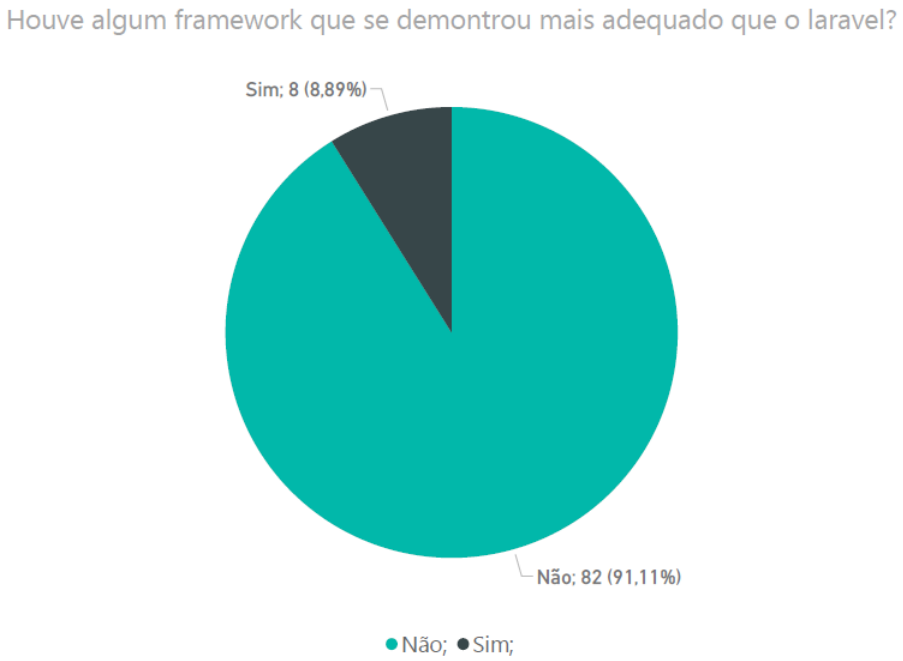


Gráfico 16 - Houve algum framework que têm se demonstrado mais adequado em relação aos projetos com os quais têm trabalhado? O qual tenha te motivado a deixar de utilizar o Laravel.
Fonte: Próprio autor.

.

4.4 Outro Framework

As perguntas dessa seção são inteiramente voltadas para o framework citado na última questão respondida.

Nesta seção, foi identificado uma contradição ou confusão dos entrevistados na resposta mandada, para evitar análise de dados incorretas, as respostas deles nesta seção serão omitidas, dos 8 respondentes em 3 deles foram identificados com este problema, sendo assim, serão contadas apenas 5 participantes.

A primeira questão procura identificar este framework, os resultados desta questão se deram da seguinte forma, 3 (60%) dos entrevistados declararam utilizar o Synfony, 1 declarou utilizar o CodeIgniter e 1 declarou utilizar o CakePHP. Estas informações seguem no Gráfico 17.

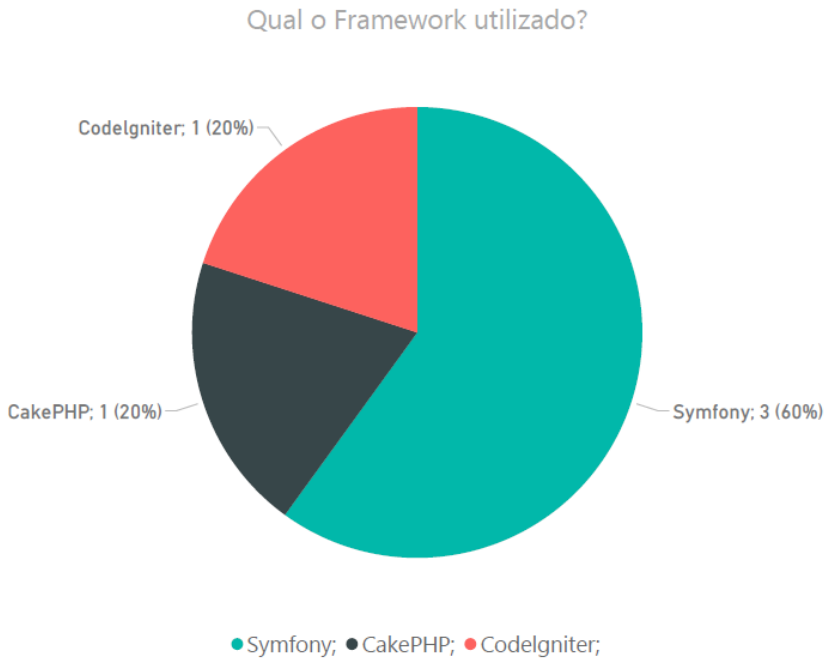


Gráfico 17 - Qual o Framework utilizado.
Fonte: Próprio autor;

A segunda questão procura identificar motivos que levaram o entrevistado a utilizar o framework, nessa questão as respostas eram de múltipla escolha, o que significa que os entrevistados puderam selecionar mais de uma resposta, através desta questão obteve-se o seguinte resultado, 5 dos entrevistados informaram que um dos motivos foi a facilidade de aprendizagem, 4 dos entrevistados informaram que um dos motivos da utilização foi a documentação de fácil acesso e entendível, 3 dos entrevistados marcaram que um dos motivos era a comunidade maior, 3 responderam que um dos motivos era a popularidade do framework, 3 dos respondentes declararam que um dos motivos e o fato de ser um *software* livre, 3 dos respondentes afirmaram que um dos motivos é a organização, estrutura de código e arquivos, 3 dos entrevistados declararam que um dos motivos era a recomendação de amigos ou empresa, 2 afirmaram ser o relacionamento de entidades, 2 declararam a utilização de templates, 1 declarou testes mais intuitivos, 1 declarou utilização de arquivo de rotas, 1 informou ser a utilização de interações via linha de comando e 1 afirmou ser pela utilização do QueryBuilder. Estas informações seguem no Gráfico 18.

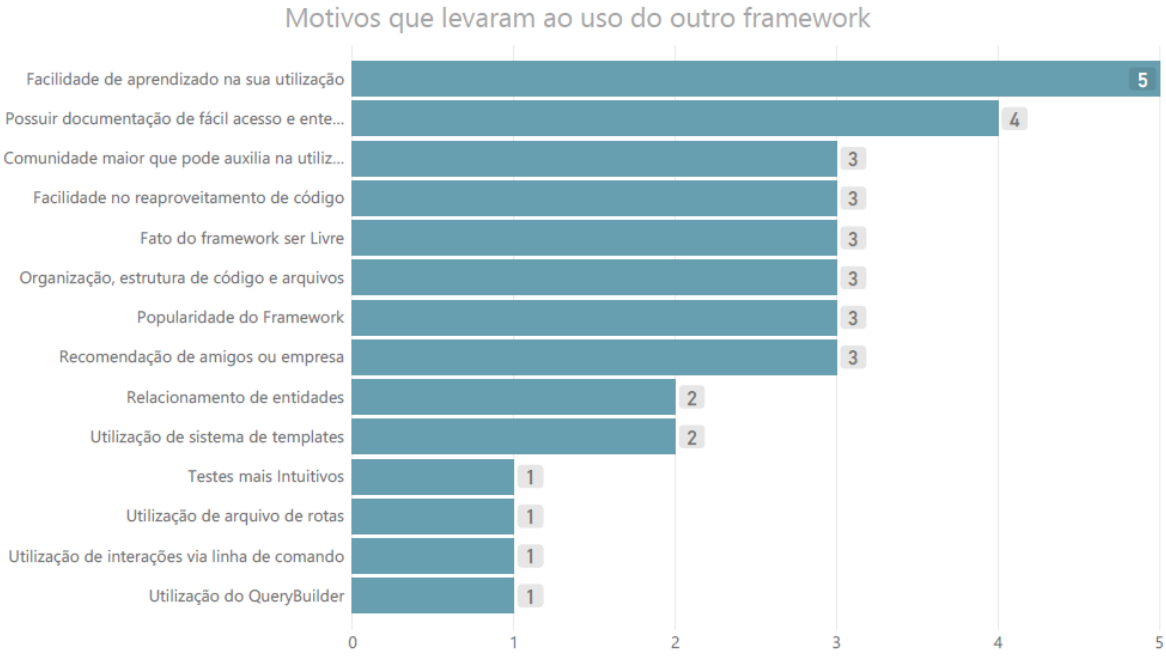


Gráfico 18 - Quais motivos o levaram a utilizar do Framework.
Fonte: Próprio autor;

A terceira questão identifica o tempo em que os entrevistados conhecem o framework, o resultado encontrado foi que, dentre os 5 entrevistados, 1 dos entrevistados conhecia o framework por uma média de tempo de 5 a 10 anos, 2 conheciam há mais de um ano e menos que quatro anos e 2 conheciam há um ano ou menos. Estas informações seguem no Gráfico 19.

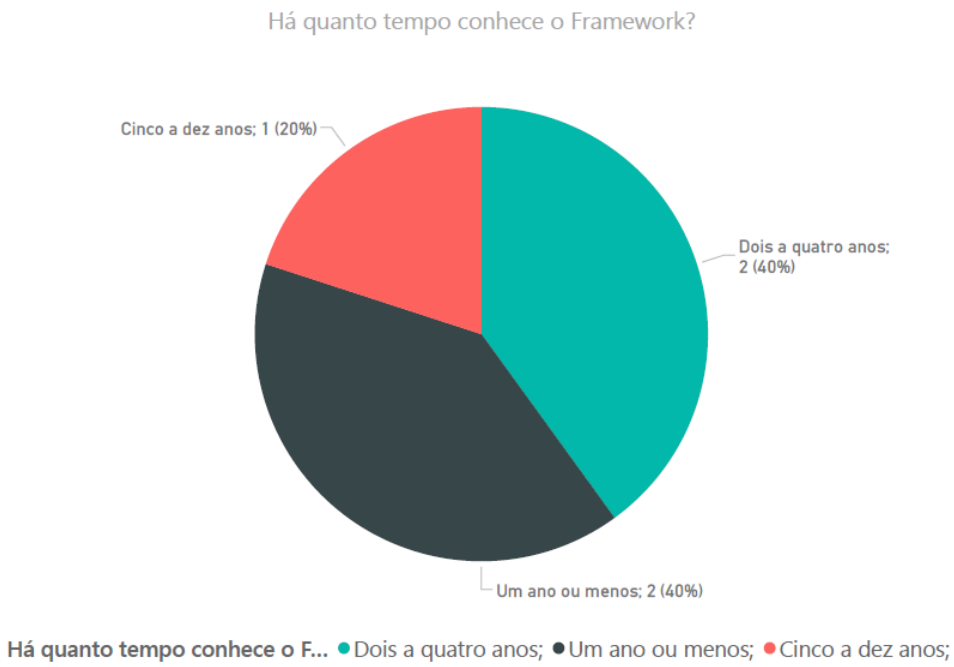


Gráfico 19 - Tempo que os entrevistados conhecem os frameworks analisados.
Fonte: Próprio autor;

Com relação as frequências de uso dos frameworks informados, foi preparado uma questão que coletava através de uma escala (escala de 1 a 10) a frequência de uso, o resultado deu-se da seguinte forma, dentre os 5 entrevistados, 4 marcaram utilizar ele com uma frequência nível oito e 1 marcou como nível 7. Estas informações seguem no Gráfico 20.

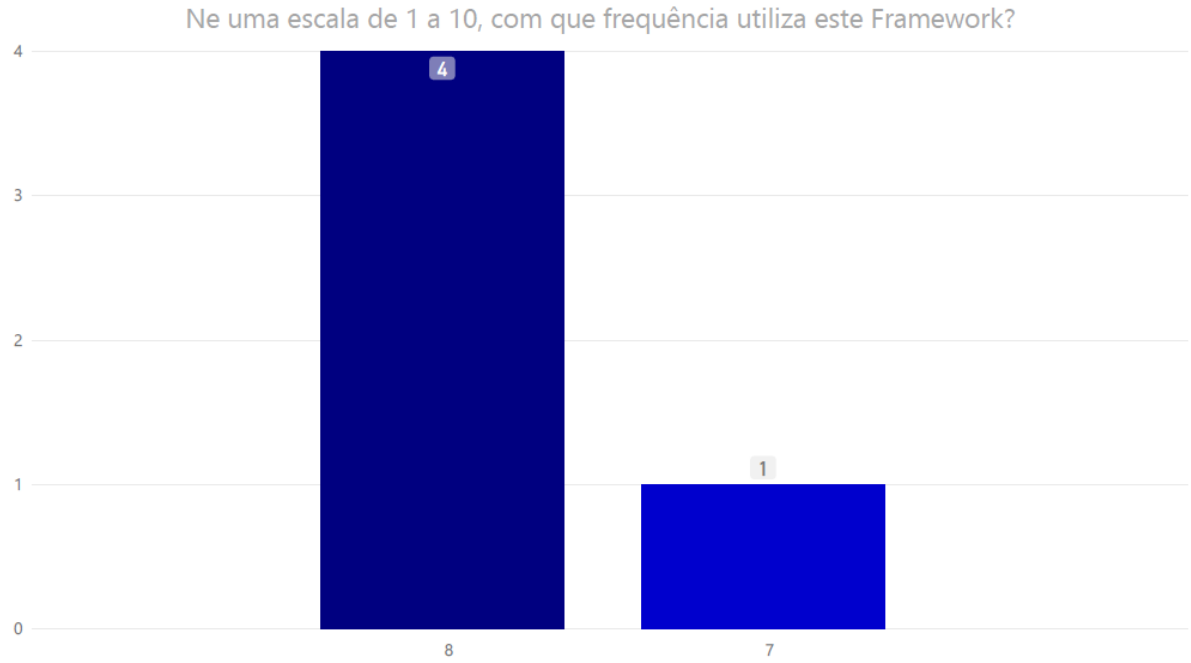


Gráfico 20 – Frequência que o entrevistado utiliza este Framework
Fonte: Próprio autor;

Para analisar melhor estes dados foi preparado um gráfico relacionando o framework com o nível de dificuldade, onde nele podemos ver que, 2 dos entrevistados deram marcaram o nível de frequência oito para o Symfony, 1 marcou oito de nível de frequência para o CakePHP, 1 marcou oito de dificuldade para o CodeIgniter e 1 marcou sete como nível de frequência para o Symfony. As respostas coletadas sobre a frequência de utilização dos frameworks apontam que a frequência que estes frameworks são muito utilizados devido ao nível de utilização ser maior que 6. Estas informações seguem no Gráfico 21.

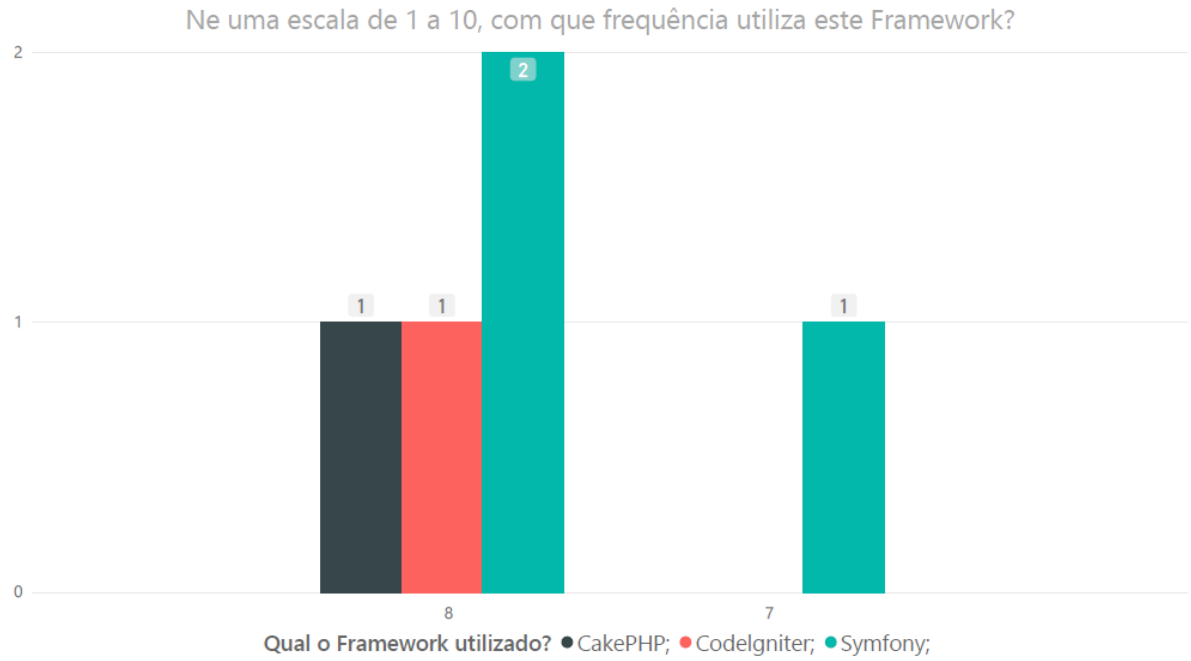


Gráfico 21 - Frequência que o entrevistado utiliza relacionado com o nome do Framework
Fonte: Próprio autor;

Para analisar a dificuldade encontrada na aprendizagem dos entrevistados, foi preparado uma questão que coletava através de uma escala (escala de 1 a 10) o nível de dificuldade de aprendizagem, o resultado deu-se da seguinte forma, 3 marcaram o nível de dificuldade como quatro, 1 marcou como três e 1 marcou como oito. Estas informações seguem no Gráfico 22.

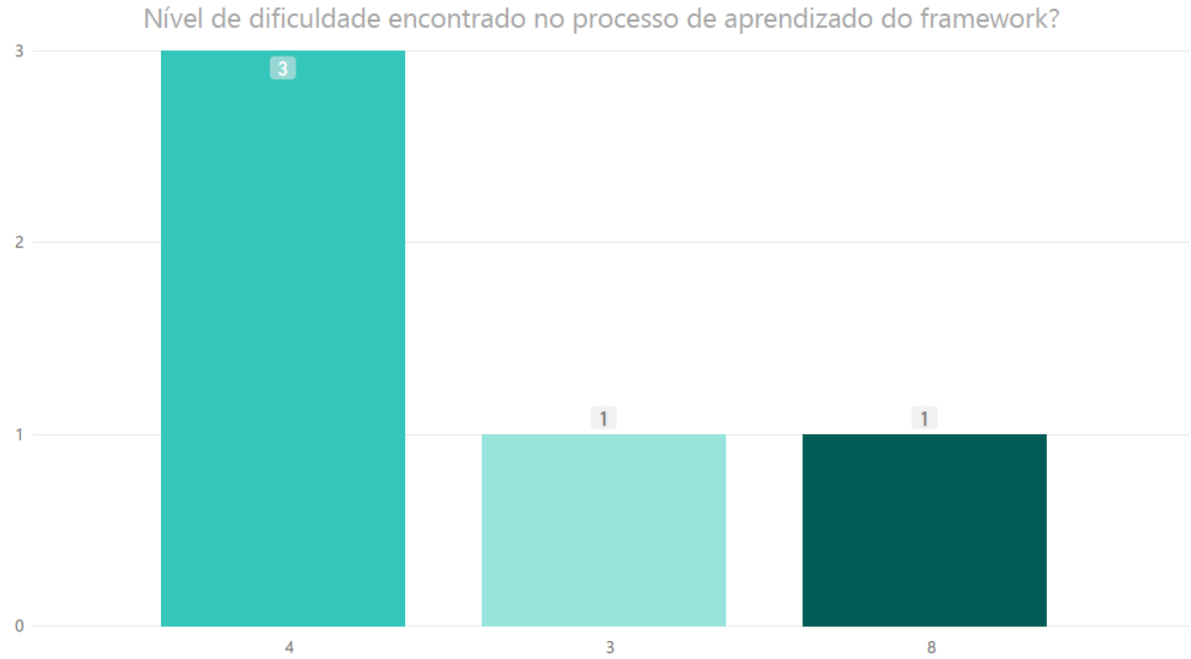


Gráfico 22 - Nível de dificuldade encontrado no processo de aprendizado do Framework.
Fonte: Próprio autor.

Para analisar melhor estes dados foi preparado um gráfico relacionando o framework com o nível de dificuldade, onde nele podemos ver que, 2 dos entrevistados marcaram o nível de dificuldade 4 para o Symfony, 1 marcou quatro de nível de dificuldade para o CakePHP, 1 marcou 3 de dificuldade para o CodeIgniter e 1 marcou oito como nível de dificuldade para o Symfony. As respostas coletadas sobre o nível de dificuldade de aprendizagem dos frameworks apontam que a dificuldade destes frameworks é bem baixa, porém para o Synfony ouve um entrevistado que considerou sua dificuldade com 8 o que demonstrou que este possui um nível de dificuldade maior para alguns. Estas informações seguem no Gráfico 23.

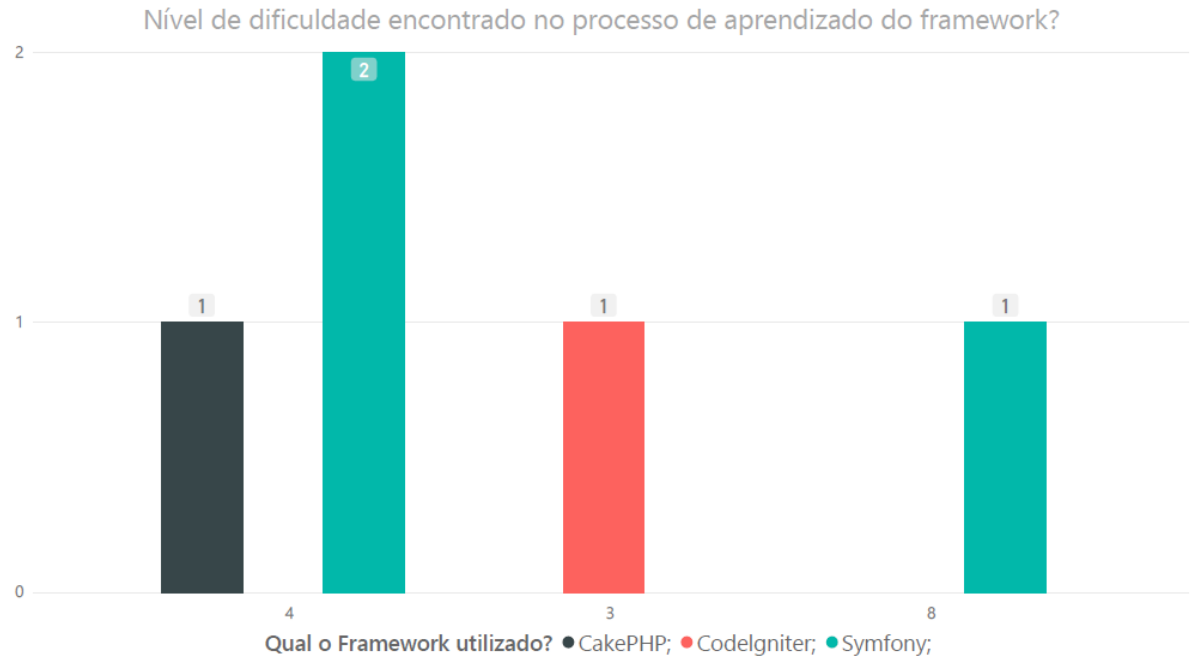


Gráfico 23 - Nível de dificuldade encontrado no processo de aprendizado relacionando com os nomes dos frameworks.
Fonte: Próprio autor.

Para analisar o quanto estes frameworks auxiliam na manutenibilidade de *software*, foi preparado uma questão que coletava através de uma escala (escala de 1 a 10) o nível de que o *software* auxiliava na manutenibilidade, o resultado deu-se da seguinte forma, 1 dos entrevistados declararam que o CodeIgniter auxilia na manutenibilidade em nível dez, 3 dos entrevistados declararam que o Symfony auxilia na manutenibilidade em nível nove e 1 dos entrevistados declararam que o CodeIgniter auxilia na manutenibilidade em nível três. Atraves destas respostas foi possível constatar que para os entrevistados, os frameworks Synfony e CodeIgniter possuem um nível de auxílio a manutenibilidade de software superior ao CakePHP que teve a nota 3. Estas informações seguem no Gráfico 24.

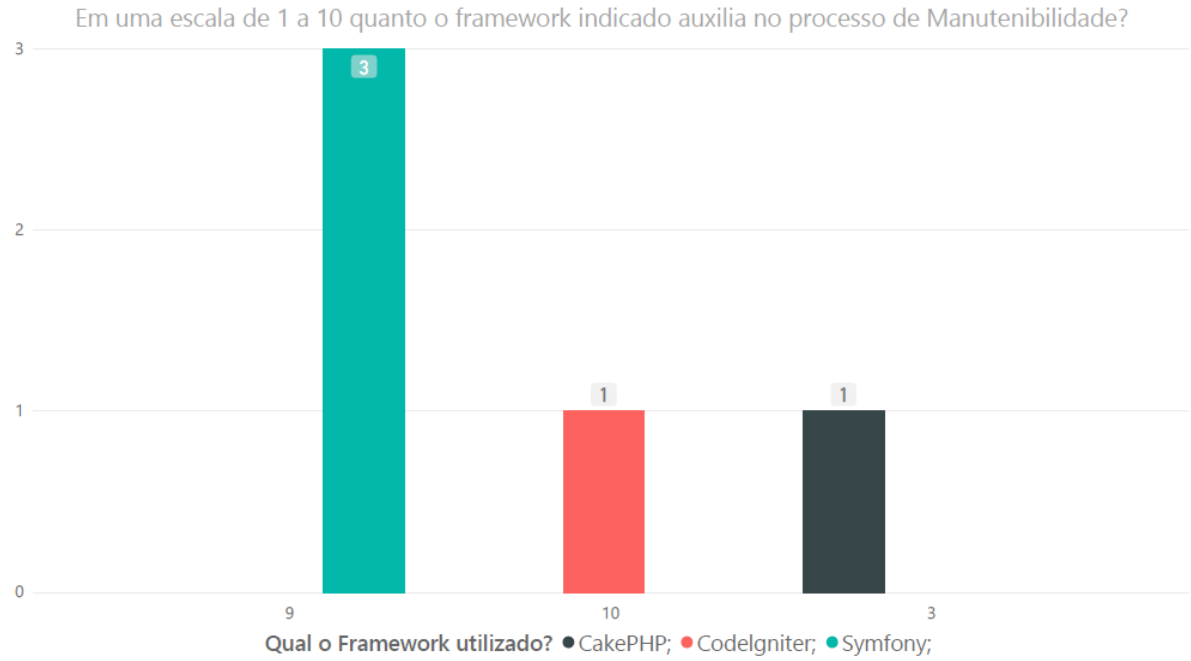


Gráfico 24 - Quanto o Framework indicado auxilia na manutenibilidade de *software*.
Fonte: Próprio autor.

4.5 Manutenibilidade de *Software*

Nesta seção foram feitas perguntas relacionadas a experiência do entrevistado com manutenibilidade de *software*.

Na primeira questão se busca avaliar o nível de conhecimento do entrevistado acerca da manutenção de *software*, foi preparado uma questão que coletava através de uma escala (escala de 1 a 10) o nível de conhecimento de manutenibilidade do entrevistado, o resultado deu-se da seguinte forma, 8 dos entrevistados marcaram que possuíam o nível de conhecimento dez, 10 dos entrevistados declararam que possuíam o nível de conhecimento nove, 23 dos entrevistados afirmaram que possuíam o nível de conhecimento oito, 17 dos entrevistados alegaram que possuíam o nível de conhecimento sete, 23 dos entrevistados assinalaram que possuíam o nível de conhecimento seis, 7 dos entrevistados apontaram que possuíam o nível de conhecimento cinco, 2 dos entrevistados declararam que possuíam o nível de conhecimento três. Os dados coletados demonstraram que a grande maioria dos entrevistados possui um conhecimento de nível me médio ou superior com relação a manutenção de software, o que demonstra que possuem conhecimentos acerca do assunto. Estas informações seguem no Gráfico 25.

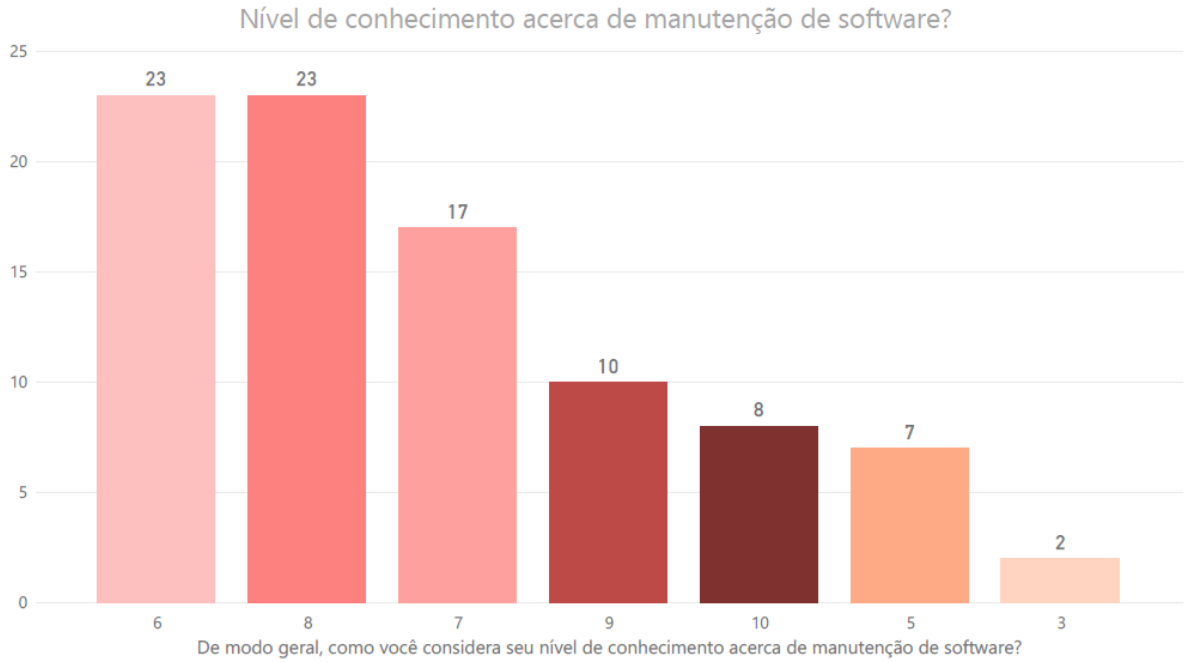


Gráfico 25 - Nível de conhecimento de manutenção de *software*.
Fonte: Próprio autor.

As próximas questões procuram analisar a frequência de preocupação do entrevistado com os atributos definidos na ISO 9126-1 (2003).

Quanto a importância na analisabilidade as respostas foram, dentre os 90 entrevistados, 52 (57,78%) responderam que “Na maioria dos casos”, 21 (23,33%) assinalaram “Em todos os casos”, 13 (14,44%) marcaram “Eventualmente”, 3 (3,33%) apontaram “Raramente”, 1 respondeu como outro dizendo “varia por time e projeto”. Analisando os dados obtidos pode-se perceber que há uma alta preocupação dos entrevistados relacionada a analisabilidade de software, visto que 81,11% dos entrevistados assinalaram como “Na maioria dos casos” e “Em todos os casos” e apenas 3.33% como “Raramente” e “Nunca” Estas informações seguem no Gráfico 26.

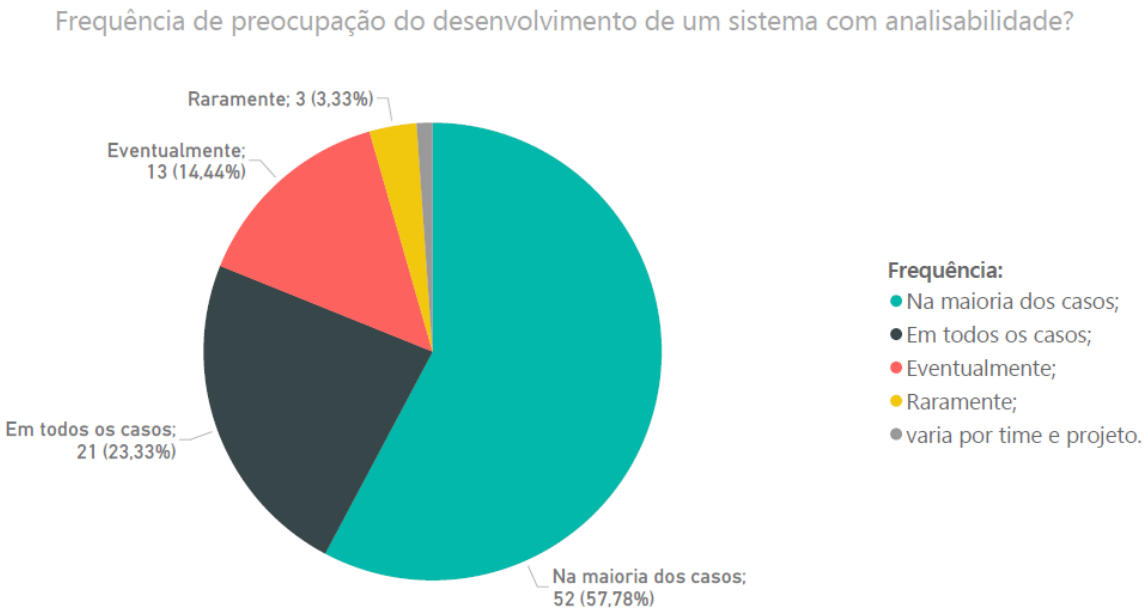


Gráfico 26 - Preocupação com analisabilidade.
Fonte: Próprio autor.

As respostas sobre a preocupação com modificabilidade se deram da seguinte forma, dentre os 90 entrevistados, 46 (51,11%) responderam que “Na maioria dos casos”, 23 (25,56%) assinalaram “Em todos os casos”, 17 (18,89%) marcaram “Eventualmente”, 2 (2,22%) apontaram “Raramente”, 1 (1,11%) respondeu como outro dizendo “varia por time e projeto”, e 1 (1,11%) respondeu como outro dizendo “não entendi a pergunta”. Pelos dados coletados pode-se perceber que há uma certa preocupação com a modificabilidade, visto que 76,67% dos entrevistados assinalaram como “Na maioria dos casos” e “Em todos os casos” e apenas 2.22% como “Raramente” e “Nunca”. Estas informações seguem no Gráfico 27.

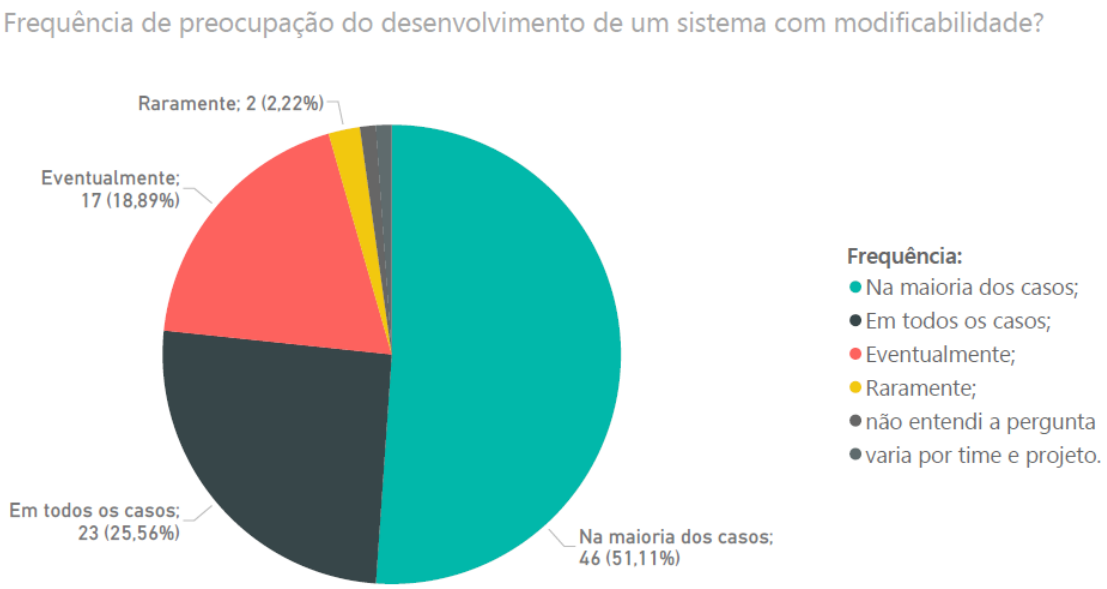


Gráfico 27 - Frequência de preocupação com modificabilidade.
Fonte: Próprio autor.

As respostas quanto a frequência de preocupação com estabilidade de *software* se deram desta forma, dentre os 90 entrevistados, 44 (48,89%) responderam que “Na maioria dos casos”, 40 (44,44%) assinalaram “Em todos os casos”, 4 (4,44%) marcaram “Eventualmente”, 1 (1,11%) apontaram “Raramente”, 1 (1,11%) respondeu como outro dizendo “varia por time e projeto”. Pelos dados colhidos pode-se identificar que há uma grande preocupação com a estabilidade de *software*, visto que 93,33% dos entrevistados assinalaram como “Na maioria dos casos” e “Em todos os casos” e apenas 1.11% como “Raramente” e “Nunca”. Estas informações seguem no Gráfico 28.

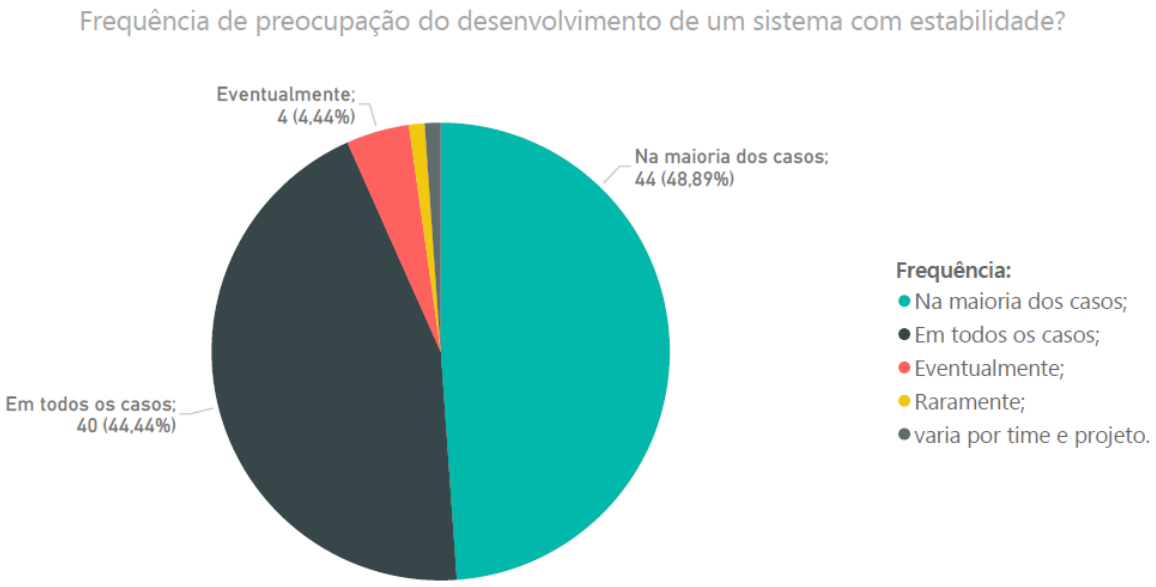


Gráfico 28 - Frequência de preocupação com estabilidade.
Fonte: Próprio autor.

Com relação a frequência de preocupação com a testabilidade os resultados apontaram que, dentre os 90 entrevistados, 40 (44,44%) responderam que “Na maioria dos casos”, 18 (20%) assinalaram “Em todos os casos”, 13 (14,44%) marcaram “Eventualmente”, 17 (18,89%) apontaram “Raramente”, 1 (1,11%) respondeu dizendo “Nunca”, 1 (1,11%) respondeu como outro dizendo “varia por time e projeto”. Através dos dados pode-se identificar que a preocupação com a testabilidade de *software* é visivelmente menor comparada as outras, visto que apenas 64,44% dos entrevistados assinalaram como “Na maioria dos casos” e “Em todos os casos” e 20% como “Raramente” e “Nunca”. Estas informações seguem no Gráfico 29.

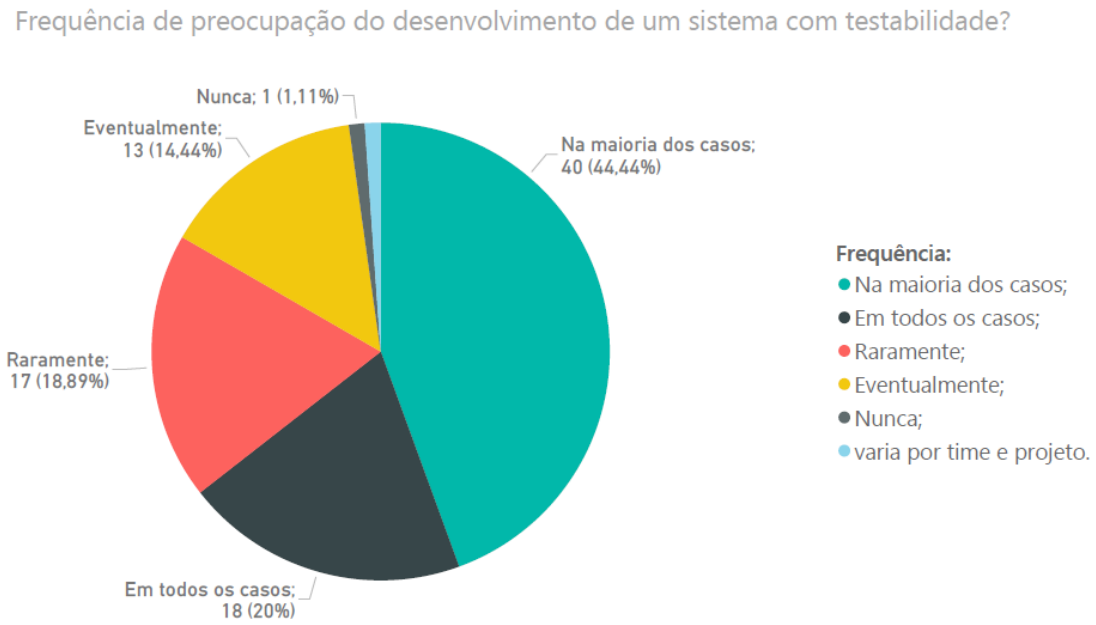


Gráfico 29 - Frequência de preocupação com a testabilidade.
Fonte: Próprio autor.

Quanto as respostas sobre a conformidade relacionada a manutenibilidade foram obtidos os seguintes resultados, dentre os 90 entrevistados, 50 (55,56%) responderam que “Na maioria dos casos”, 32 (35,56%) assinalaram “Em todos os casos”, 5 (5,55%) marcaram “Eventualmente”, 2 (2,22%) apontaram “Raramente”, 1 (1,11%) respondeu como outro dizendo “varia por time e projeto”. Através dos dados pode-se identificar que a preocupação com a conformidade relacionada a manutenibilidade de *software* é relativamente alta apesar da testabilidade ser baixa, 92,12% dos entrevistados assinalaram como “Na maioria dos casos” e “Em todos os casos” e 2,22% como “Raramente” e “Nunca”. Estas informações seguem no Gráfico 30.

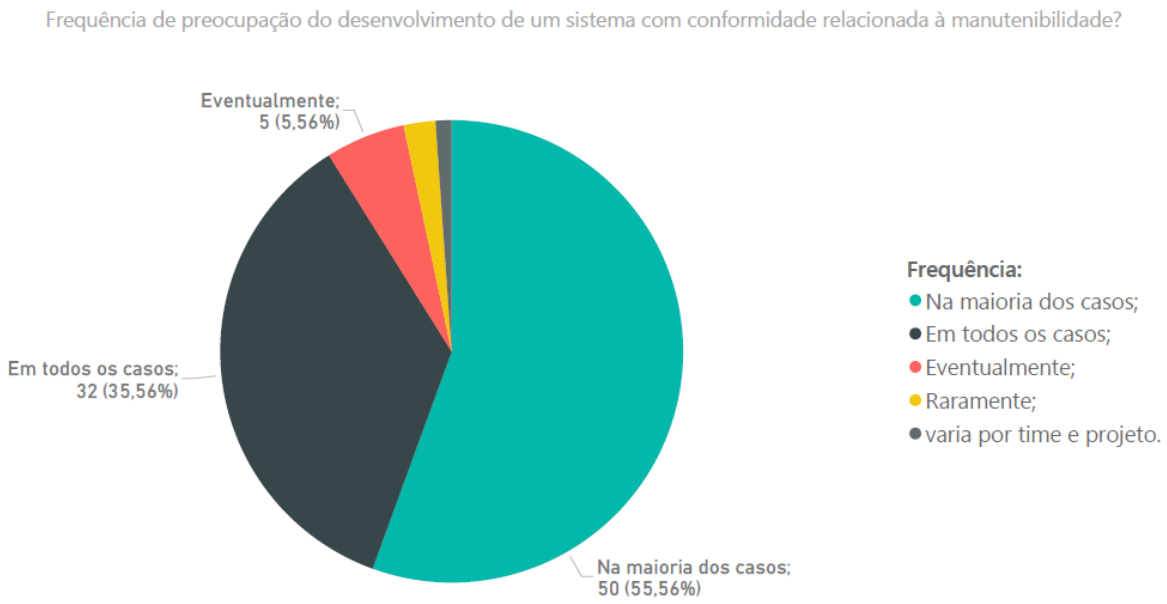


Gráfico 30 - Frequência de preocupação com a conformidade relacionada a manutenibilidade de *software*.
Fonte: Próprio autor.

4.6 Laravel Framework e Manutenibilidade

Esta seção relaciona conceitos de manutenibilidade de *software* com funções do Laravel a fim de coletar informações quanto a seu auxílio na manutenibilidade.

A primeira questão coletou informações sobre o artisan console e sua contribuição para a manutenibilidade, os resultados se deram da seguinte forma, dentre os 90 entrevistados, 46 (51,11%) responderam que “Na maioria dos casos”, 20 (22,22%) assinalaram “Em todos os casos”, 16 (17,77%) marcaram “Eventualmente”, 4 (4,44%) apontaram “Nunca”, 3 (3,33%) apontaram “Raramente”. Analisando os dados coletados é possível verificar que a maioria dos entrevistados concorda que a utilização do artisan console auxilia na padronização e analisabilidade de código-fonte, isso pode ser observado somando o número de entrevistados assinalaram como “Na maioria dos casos” e “Em todos os casos”, que totaliza cerca de 73.33%. Estas informações seguem no Gráfico 31.

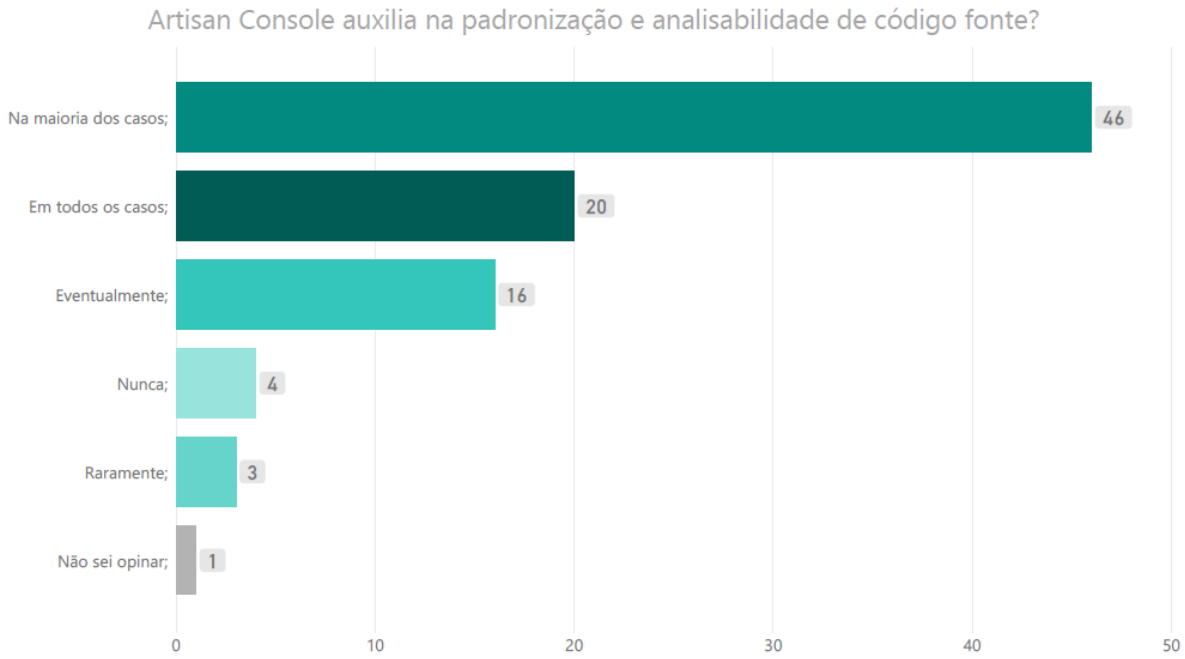


Gráfico 31 - Contribuição do artisan console para a analisabilidade.
Fonte: Próprio autor.

Quanto a contribuição para a manutenibilidade e reutilização de código-fonte do sistema de blades, os resultados se deram da seguinte forma, dentre os 90 entrevistados, 43 (47,77%) responderam que “Na maioria dos casos”, 25 (27,77%) assinalaram “Em todos os casos”, 16 (17,77%) marcaram “Eventualmente”, 3 (3,33%) apontaram “Raramente”, 3 (3,33%) apontaram “Não sei opinar”. Estes dados demonstraram que mais da metade dos entrevistados acredita que o sistema de templates (blades) auxilia na reutilização de código-fonte. Estas informações seguem no Gráfico 32.

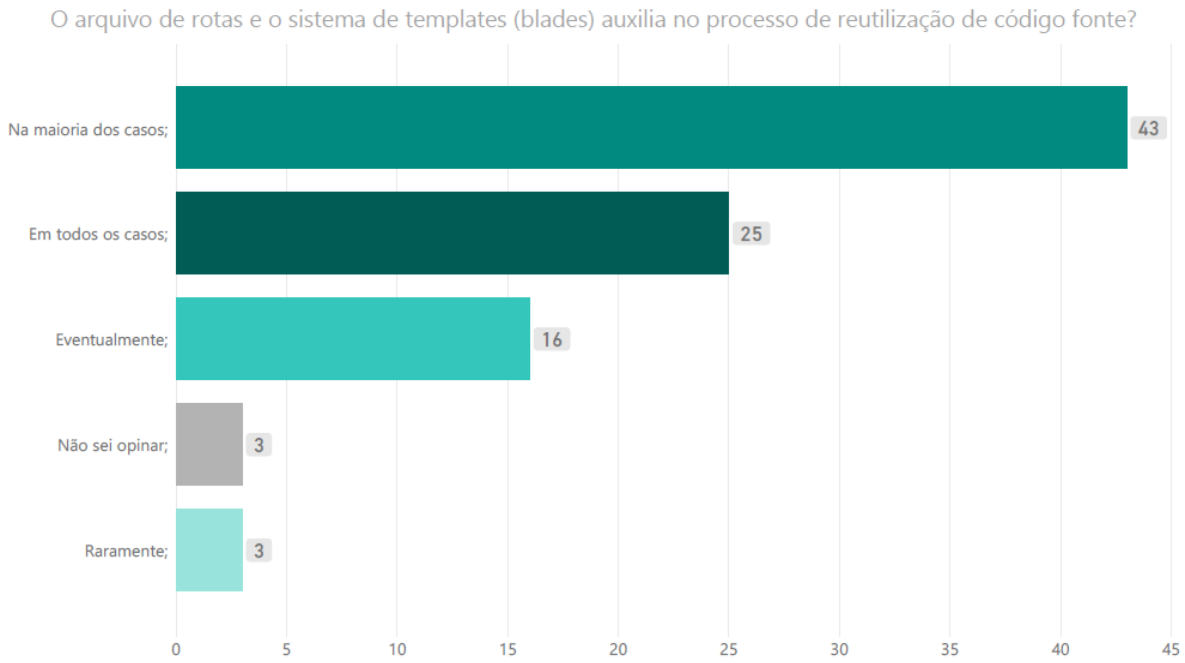


Gráfico 32 - Contribuição do sistema de templates (blade) para a reutilização de código-fonte.
Fonte: Próprio autor.

Quanto a contribuição de um sistema baseado em testes, o resultado se deu da seguinte forma, dos 90 entrevistados, 52 (57,77%) responderam que “Na maioria dos casos”, 21 (23,33%) assinalaram “Em todos os casos”, 11 (12,22%) marcaram “Eventualmente”, 5 (5,55%) apontaram “Raramente”, 1 (1,11%) respondeu “Nunca”. Estes dados demonstraram que mais de 80% dos entrevistados acredita que o laravel auxilia na testabilidade de software, visto que 57,77% responderam que “Na maioria dos casos” e 23,33% assinalaram “Em todos os casos”. Estas informações seguem no Gráfico 33.

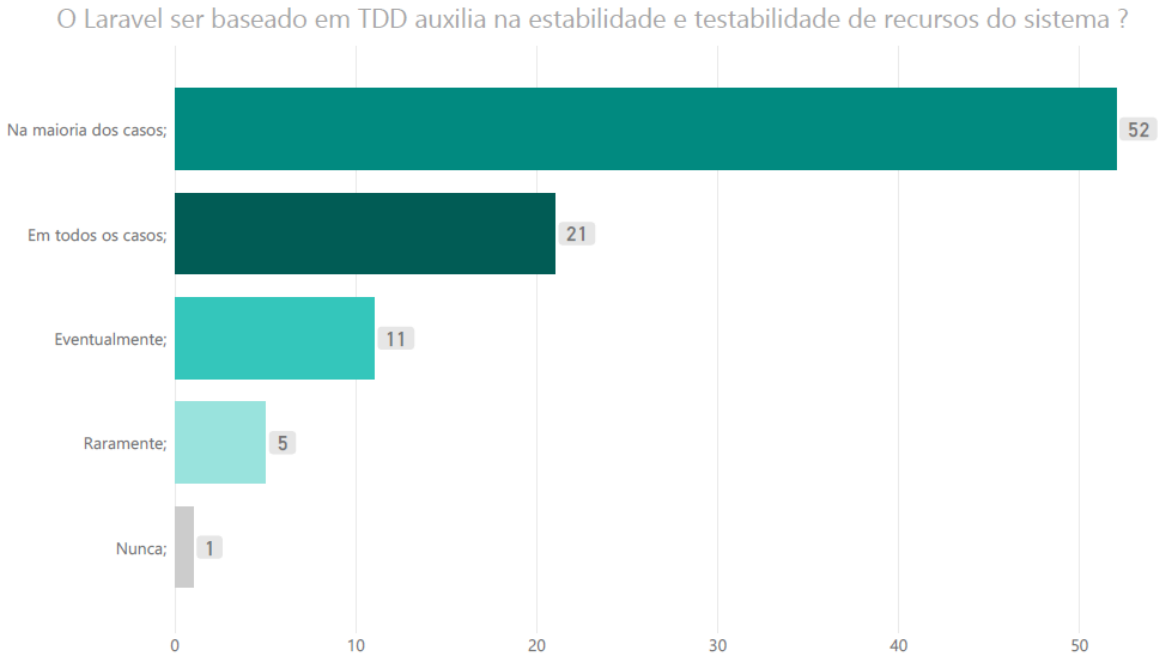


Gráfico 33 - Contribuição de um sistema baseado em testes para a testabilidade e estabilidade.
Fonte: Próprio autor.

Os resultados quanto a utilização do Eloquent no auxílio a implementação de código-fonte se deram da seguinte forma, dentre os 90 entrevistados, 44 (48,88%) responderam que “Na maioria dos casos”, 29 (32,22%) assinalaram “Em todos os casos”, 12 (13,33%) marcaram “Eventualmente”, 2 (2,22%) apontaram “Raramente”, 1 (1,11%) respondeu como outro dizendo “nada a ver essa pergunta”, 1 (1,11%) respondeu “Não sei opinar”, 1 (1,11%) respondeu “Nunca”. Segundo os dados coletados a maior parte dos entrevistados acredita que a utilização do Eloquent auxilia na implementação e manutenção de código-fonte, visto que 81.11% assinalaram “Na maioria dos casos” e “Em todos os casos”. Estas informações seguem no Gráfico 34.

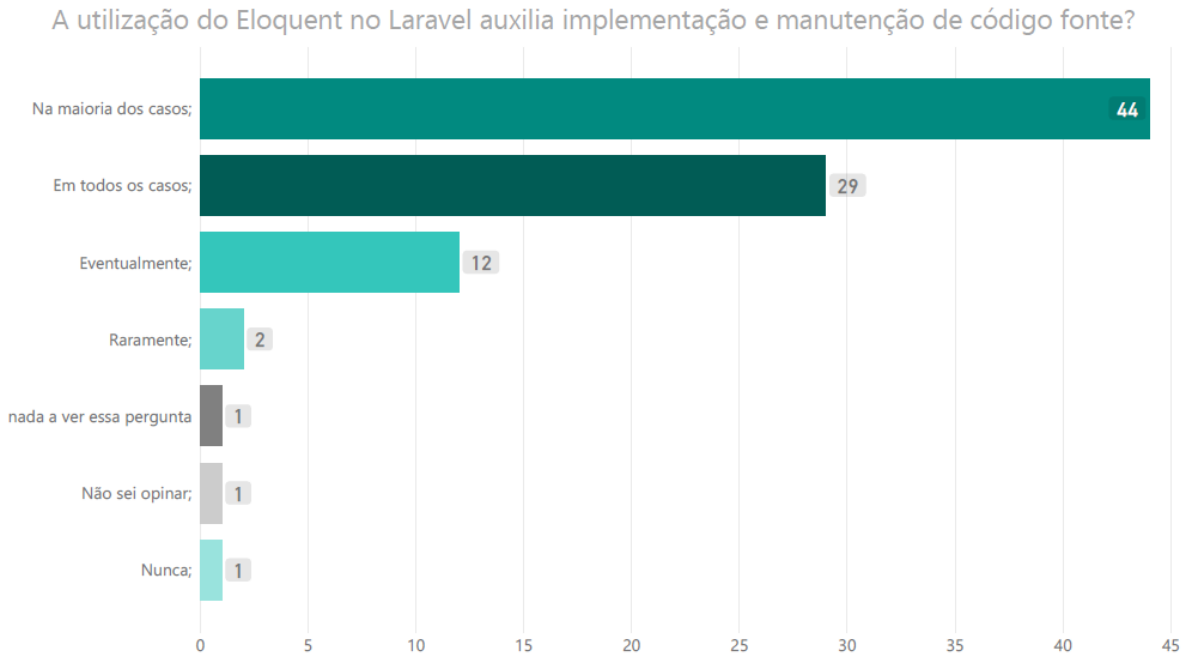


Gráfico 34 - Contribuição do Eloquent na implementação e manutenção de código-fonte.
Fonte: Próprio autor.

Quanto a contribuição das migrations para modificação e controle de versão do banco de dados, o resultado deu-se da seguinte forma, dentre os 90 entrevistados, 45 (50%) responderam que “Na maioria dos casos”, 38 (42,22%) assinalaram “Em todos os casos”, 5 (5,55%) marcaram “Eventualmente”, 1 (1,11%) respondeu como outro dizendo “Boa parte dos casos, já que ele é pobre no rollback”, 1 (1,11%) respondeu como outro dizendo “nada a ver essa pergunta”. Analisando os dados coletados é possível identificar que, a maioria dos entrevistados concorda que a utilização das migrations auxilia no controle de versão do banco. Estas informações seguem no Gráfico 35.

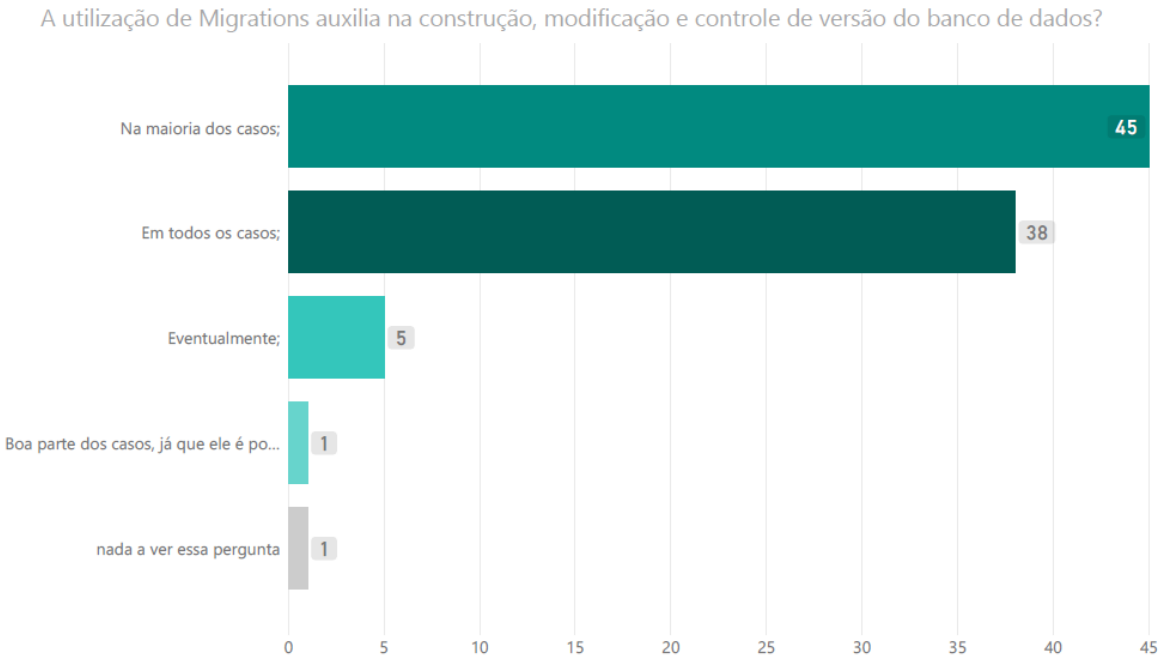


Gráfico 35 - Contribuição das migrations para modificação e controle de versão do banco de dados.
Fonte: Próprio autor.

Quanto a pergunta final, o Laravel realmente auxilia na manutenibilidade de *software*, deu-se o seguinte resultado, dentre os 90 entrevistados, 53 (58,89%) responderam que “Na maioria dos casos”, 26 (28,89%) assinalaram “Em todos os casos”, 9 (10%) marcaram “Eventualmente”, 2 (2,22%) apontaram “Raramente”. Segundo os resultados coletados, há maioria dos entrevistados acredita que o laravel auxilia na manutenibilidade de software, visto que ouve 93.33% de respostas positivas identificadas como, “Na maioria dos casos” e “Em todos os casos”. Estas informações seguem no Gráfico 36.

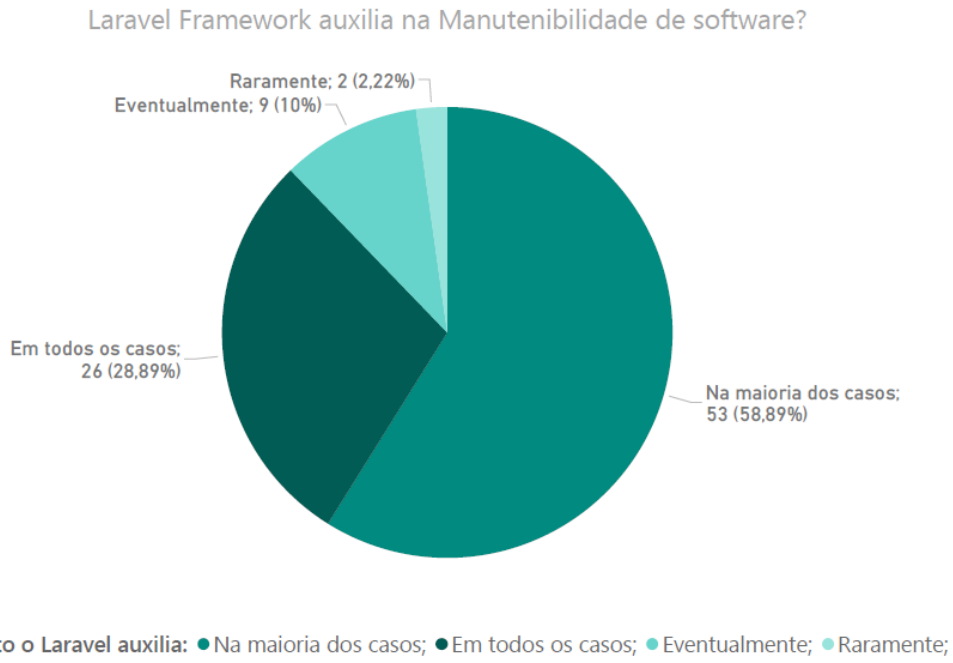


Gráfico 36 - O Laravel Framework auxilia na manutenibilidade de *software*.
Fonte: Próprio autor.

Para analisar melhor a opinião dos entrevistados foram preparados dois gráficos, um com a resposta final relacionada a formação acadêmica, e outro relacionado ao tempo de experiência na área do respondente.

O Gráfico 37 com a resposta final relacionada a formação acadêmica demonstrou que:

- Graduando/Graduados: 14 concordam que o Laravel auxilia na manutenibilidade de *software*, 5 acham que eventualmente ele auxilia e 33 acreditam que na maioria dos casos ele auxilia;
- Pós-Graduandos/Pós-graduados: 5 concordam que o Laravel auxilia na manutenibilidade de *software*, 4 acham que eventualmente ele auxilia, 9 acredita que na maioria dos casos ele auxilia e 2 acreditam que raramente;
- Técnico: 5 concordam que o Laravel auxilia na manutenibilidade de *software*, 10 acreditam que na maioria dos casos ele auxilia;
- Mestrando/Mestre: 5 concordam que o Laravel auxilia na manutenibilidade de *software*;
- Autodidata: 1 acredita que na maioria dos casos ele auxilia;

Apesar de a maioria acreditar que o laravel pode sim auxiliar na manutenibilidade é possível identificar que nem todos acreditam nisso, isto fica claro

quando analisamos melhor as respostas, pode-se perceber um o número de “Na maioria dos casos” o que pode indicar que apesar do framework atender a algumas necessidades ele pode não auxiliar em sua totalidade.

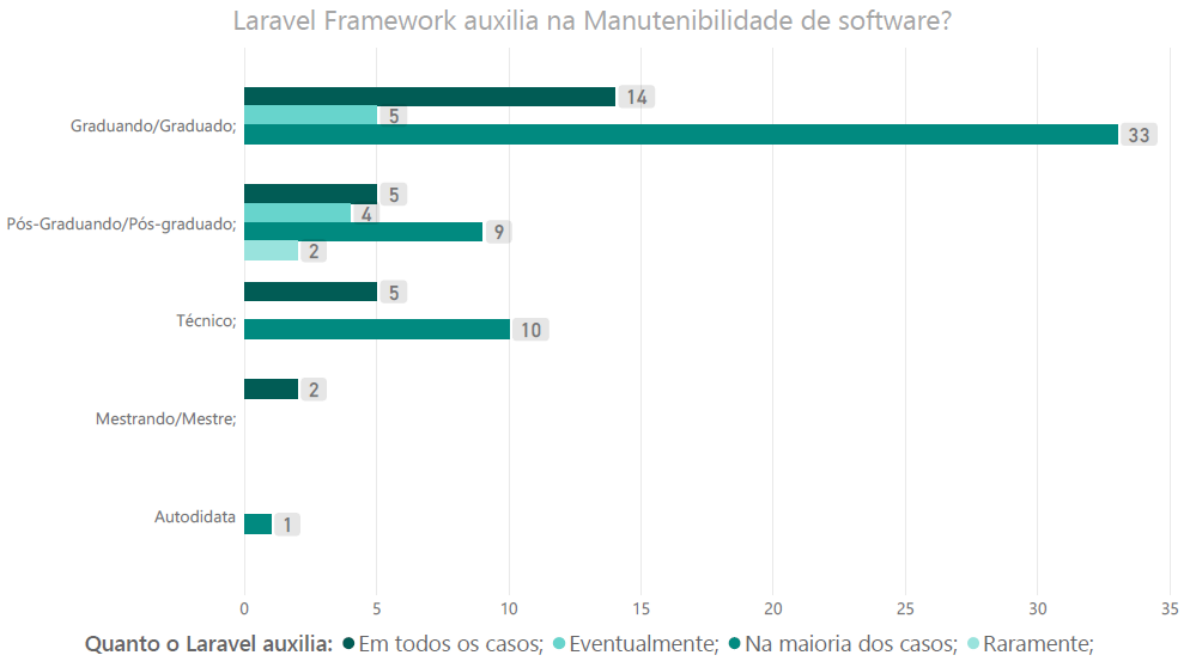


Gráfico 37 - O Laravel Framework auxilia na manutenibilidade de *software*, relacionado com a formação acadêmica.
Fonte: Próprio autor.

O Gráfico 38 com a resposta final relacionada ao tempo de experiência demonstrou que:

- Entrevistados com um ano ou menos de experiência: 6 concordam que o Laravel auxilia na manutenibilidade de *software*, 1 acha que eventualmente ele auxilia e 9 acreditam que na maioria dos casos ele auxilia;
- Entrevistados com dois a quatro anos de experiência: 6 concordam que o Laravel auxilia na manutenibilidade de *software*, 4 acham que eventualmente ele auxilia e 22 acreditam que na maioria dos casos ele auxilia;
- Entrevistados com cinco a dez anos de experiência: 6 concordam que o Laravel auxilia na manutenibilidade de *software*, 2 acham que

eventualmente ele auxilia e 4 acreditam que na maioria dos casos ele auxilia;

Através do gráfico é possível notar a maior parte dos entrevistados que acreditam totalmente na ferramenta no auxílio a manutenibilidade de software, são pessoas que possuem mais tempo de experiência com o framework.

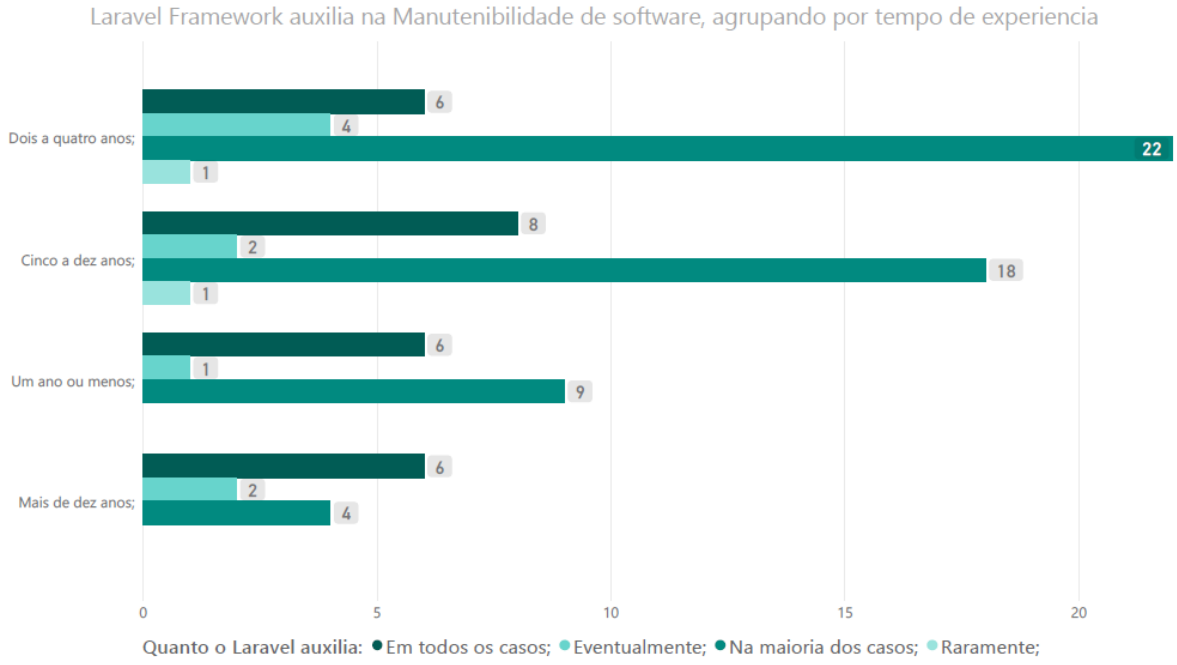


Gráfico 38- O Laravel Framework auxilia na manutenibilidade de *software*, relacionado com ao tempo de experiência.
Fonte: Próprio autor.

5 CONCLUSÃO

O presente trabalho teve como objetivo analisar os benefícios encontrados na utilização do Laravel framework em manutenibilidade de *software*.

Com base nas respostas coletadas através do questionário, foi possível identificar que de maneira geral a maioria dos entrevistados acredita que o Laravel pode auxiliar no contentamento dos atributos de manutenibilidade de *software*, visto que com base na última pergunta do questionário foi possível identificar que mais de 85% (este cálculo foi feito juntando os que responderam “Na maioria dos casos” com os que responderam “Em todos os casos”), porém o framework não consegue atender em sua totalidade todos os conceitos da manutenibilidade de software.

Com relação aos benefícios da utilização do Framework, pode-se identificar dentre as respostas que, as funcionalidades artisan console, sistema de templates (blades), testes, eloquent e migrations apresentaram um bom número de respostas positivas, demonstrando as mesmas podem ser benefícios que auxiliam na manutenibilidade de software.

Dentre os motivos de utilização do laravel e do outro framework pode-se perceber que o laravel possui mais motivos que o outro framework, visto que o laravel teve 20 tipos de respostas diferentes, os motivos, os que não foram citados no outro framework são Facades, Migration, Performance, Pode ser usado em equipes pequenas, Projeto já existente, Schedule Console.

Sendo assim, foi possível concluir que de maneira geral o Laravel pode auxiliar a satisfazer os atributos de manutenibilidade de *software* citados na ISO 9126-1 (2003), sendo capaz de acelerar o desenvolvimento de sistemas, minimizando erros de programação, auxiliando na reutilização e padronização de código-fonte evitando também duplicidade de código, mapeamento de rotas, organização da estrutura de arquivos, simplificação da comunicação com o banco de dados e gerenciamento estrutural, facilitando a criação de testes influenciando na estabilidade do sistema, melhorando os resultados de produção de sistemas, agregando qualidade, confiabilidade, manutenibilidade, testabilidade, reduzindo os custos e o tempo de fabricação e manutenção do software.

Por intermédio das respostas dos entrevistados, pode-se analisar que o Laravel pode ser considerado um dos frameworks mais conhecidos na atualidade, visto que mais de 80% dos entrevistados já o conheciam e que em comparação os demais frameworks citados tiveram menos da metade do reconhecimento dos entrevistados.

Segundo as respostas dos entrevistados, também foi possível identificar que que apesar da maioria deles já ter utilizado e ainda utilizar o Laravel, uma minoria dentre os entrevistados demonstrou utilizar outros frameworks, como Symfony, CodeIgniter e CakePHP.

Também pode ser analisado que apesar da maioria dos entrevistados terem conhecimento sobre manutenção de *software*, boa parte ainda não procura atender a todos os atributos da manutenibilidade citados na ISO 9126-1 (2003), visto que na testabilidade de *software* foi identificado que cerca de 20% dos entrevistados não consideram como uma preocupação frequente a testabilidade.

6 TRABALHOS FUTUROS

Como direção para possíveis trabalhos futuros:

- Realizar um estudo mais amplo para saber quais pontos do Laravel precisam ser aperfeiçoadas para auxiliar na manutenção de um *software* em sua totalidade.
- Realizar a mesma análise usando a norma mais recente de qualidade de *software*.
- Realizar estudos e pesquisas similares em outros frameworks e ou ambientes.

REFERÊNCIAS

ABNT, Associação Brasileira de Normas Técnicas. Conheça a ABNT. Disponível em: < http://www.abnt.org.br/abnt/conheca-a-abnt > Acesso em: 23 Outubro 2018.

ANQUETIL, Nicolas; DIAS, Márcio G. Proposta de uma disciplina sobre manutenção de softwares na graduação. XXV Congresso da Sociedade Brasileira de Computação, São Leopoldo, RS – 2005.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9126: Informação e documentação: Referências. Rio de Janeiro, p. 24. 2003.

BLAZEJUK, A. Prevenção de Vulnerabilidades em Aplicações Web Utilizando o Framework Laravel. Trabalho de conclusão de curso. Universidade Federal Do Rio Grande Do Sul – Porto Alegre. 2017.

BRUSAMOLIN, V. Manutenibilidade de Software. In: Revista Digital Online, v.2, jan. 2004.

CALAZANS, A. T. S.; OLIVEIRA, M. A. L. Avaliação de Estimativa de Tamanho para Projetos de Manutenção de Softwares. Toffano Seidel Calazans et al./Proc. of Argentine Symposium on Software Engineering, 2005.

DANTAS, A. V. F. PROJETO DE INTERFACES & USABILIDADE Como construir páginas na internet que sejam fáceis de usar. Disponível em:

<https://www.colegiosalvador.com.br/sgw/upload/Projeto_de_Interfaces_e_Usabilidade.pdf>. Acesso em: 25 de Julho 2018.

FILHO, W. P. P. F. Engenharia de Software: fundamentos, métodos e padrões. Editora LTC. 2003.

FLORENTIN, V. A. F. F. Análise e Desenvolvimento de Sistemas Especificação de Sistemas. Faculdade de Tecnologia de Alagoas - FAT, 2015.

GAMMA, E; HELM, R; JOHNSON, R; VLISSIDES, J. Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

GHEZZI, C.; JAZAYERI, M.; MANDRIOLI, D. Fundamentals of Software Engineering. New Jersey: Prentice Hall, 1991.

GOMES, V. MÉTRICAS DE QUALIDADE DE SOFTWARE. Disponível em: <https://www.tiespecialistas.com.br/metricas-de-qualidade-de-software/>. Acesso em: 19 Setembro 2018.

GUERRA, A. C.; COLOMBO, Regina Maria Thienne. Tecnologia da Informação: Qualidade de Produto de Software. PBQP Software 2009.

HUMPHREY, W. S., Managing the Software Process. Addison-Wesley Publishing,Company, Massachusetts, 1990.

Laravel. Laravel - the php framework for web artisans. Disponível em: <https://laravel.com//>. Acesso em: 09 Novembro 2018.

LIENTZ, B.P.; SWANSON, E.B: Software Maintenance Management. Reading, MA, Addison-Wesley, 1980.

LOPES, E. H. T. Estudo sobre a importância da manutenção de software. Trabalho de conclusão de curso. Faculdades Integradas de Caratinga – Caratinga. 2017.

MACONRATTI, J. C. O processo de Software. Disponível em: <http://www.macoratti.net/proc_sw1.htm>. Acesso em: 04 Junho 2018.

MAZZOLA, V. B. Engenharia de Software. Universidade Federal de Santa Catarina, 2010.

MINETTO, E. L. Frameworks para Desenvolvimento em PHP. Novatec Editora Ltda., 2007.

PADUELLI, M. M. Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica. 2007. 144 p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, 2007.

PELIZZA, A. C.; BERTOLINI, Cristiano; SILVEIRA Sidnei R. Um estudo sobre Técnicas de Teste de Software no Framework Laravel. Departamento de Tecnologia da Informação (UFSM), 2017.

PETERS, J. F; PEDRYCZ, W. Engenharia de Software. / James F. Peters, Witold Pedrycz; tradução de Ana Patricia Garcia – Rio de Janeiro: Campus, 2001.

PIGOSKI, T. M. Pratical Software Maintenance: Best Practices for Managing Your Software Investment. Wiley Computer Publishing, 1996.

PINTO, S. C. C. S. Composição em WebFrameworks. Departamento de Informática Pontifícia Universidade Católica do Rio de Janeiro, 2000.

Portal da Educação. A história da organização ISO. Disponível em: <https://www.portaleducacao.com.br/conteudo/artigos/educacao/a-historia-daorganizacao-iso/40732> Acesso em: 23 Outubro 2018.

PREE, W.; SIKORA, H. Design Patterns for Object-Oriented Software Development. ICSE ’97: Proceedings of the 19th International Conference on Software Engineering, p. 663– 664, 1997.

PRESSMAN, R. S. Engenharia de Software [recurso eletrônico]: uma abordagem profissional. Roger S. Pressman. 7ªed. Dados Eletrônicos. Porto Alegre: AMGH, 2011.

PRESSMAN, R. S. Engenharia de Software. 6ª edição. São Paulo: McGraw-Hill/Makron Books do Brasil, 2006.

RIBEIRO, M. V. O reúso de código-fonte como metodologia de aprimoramento da produção de softwares. Trabalho de conclusão de curso. Faculdades Integradas de Caratinga – Caratinga. 2013.

SANDERS, J.; CURRAN, E. A Framework for Success in Software Development and Support (ACM Press) Ed.AddisonWesley,1994.

SANTOS, R. P. Critérios de Manutenibilidade Para Construção e Avaliação de Produtos de Software Orientados a Aspectos. Monografia de graduação. Universidade Federal de Lavras. – Lavras, MG – 2007.

SCHWARTZ, J. I., Construction of software. In: Practical Strategies for Developing Large Systems. Menlo Park: Addison-Wesley, 1st. ed., 1975.

SHOOMAN, M. L. Software Engineering – Design, Reliability and Management. McGraw-Hill – Singapore – 1983.

SILVA, M. F. Utilização dos modelos full-stack framework e micro-framework para o desenvolvimento de aplicações web escaláveis em linguagem php. Trabalho de conclusão de curso. Faculdades Integradas de Caratinga – Caratinga. 2016.

SILVA, R. A.; MATOS, Simone; SOUZA, João H.;MOURA, Louisi F. A manutenção de software nas empresas. Congresso Internacional de Administração – 2010.

SILVA, W.W. R. Laravel 5.1: Essencial Alta produtividade no mundo Real. 2015.

SOMMERVILLE, Ian. Engenharia de Software. 8ª edição. São Paulo: Pearson Education do Brasil, 2007.

SOMMERVILLE, Ian. Engenharia de Software. / Ian Sommerville; Tradução Ivan Bosnic e Kalinca G. de O. Gonçalves; revisão técnica Kechi Hirama. 9ªed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, F. USO DE FRAMEWORKS PARA DESENVOLVIMENTO WEB E MITOS QUE JÁ DEVERIAM TER DESAPARECIDO. Disponível em: <https://www.profissionaisti.com.br/2010/01/uso-de-frameworks-para-desenvolvimento-web-e-mitos-que-ja-deveriam-ter-desaparecido/>. Acesso em: 27 Junho 2018.

TURINI, R. PHP e Laravel: Crie aplicações web como um verdadeiro artesão. São Paulo: Casa do Código, 2015.

VALENTE, Carlos. Tópicos avançados de engenharia de software. ESAB: Escola Superior Aberta do Brasil, 2008.

VENTORIN, A. J. Principais problemas relacionados ao desenvolvimento de sistemas. Faculdade Capixaba de Nova Venécia – UNIVEN.

VERMA, A. MVC Architecture: A Comparative Study Between Ruby on Rails and Laravel. Indian Journal of Computer Science and Engineering (IJCSE), 2014.

APÊNDICE A

1.1 APÊNDICE 1: QUESTIONÁRIO

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

O presente questionário é elaborado no âmbito de um trabalho de conclusão de curso de Ciência da Computação, o qual pretende analisar a utilização do Laravel Framework no auxílio da manutenibilidade de software.
As perguntas deste são fundamentadas em estudos de autores sobre qualidade de software e documentação da ferramenta.
Os dados recolhidos são confidenciais e serão utilizados apenas para fins acadêmicos. O tempo estimado para o preenchimento deste questionário é de aproximadamente 5 a 10 minutos.
Sua colaboração é muito importante para a conclusão deste trabalho.
Obrigado por participar!

Autor:
Leonardo Webster Ribeiro da Silva
8º Período, Ciência da Computação
Rede de Ensino Doctum
leonardowebster15@gmail.com

PRÓXIMA

Nunca envie senhas pelo Formulários Google.

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

*Obrigatório

Identificação do Respondente

As perguntas a seguir procuram coletar informações sobre seus conhecimentos para entender o nível de conhecimento quanto ao tema da pesquisa.

Qual sua formação acadêmica? *

- ◯ Técnico;
- ◯ Graduando/Graduado;
- ◯ Pós-Graduando/Pós-graduado;
- ◯ Mestrando/Mestre;
- ◯ Doutorando/Doutor;
- ◯ Outro: _____

Há quanto tempo você exerce atividades na área computação? *

- ◯ Não atuo;
- ◯ Um ano ou menos;

- ◯ Dois a quatro anos;
- ◯ Cinco a dez anos;
- ◯ Mais de dez anos;

Qual cargo ocupa atua atualmente? *

- ◯ Apoio Estratégico (Diretor, CEO, CIO);
- ◯ Apoio Gerencial (Gerente de projetos, gerente de qualidade, gerente de teste);
- ◯ Apoio Operacional (Desenvolvedor, analista de sistemas, webdesigner, analista de testes);
- ◯ Estudante;
- ◯ Outro;

Você já participou do desenvolvimento de algum projeto em linguagemPHP sendo profissionalmente ou não? *

- ◯ Sim;
- ◯ Não;

VOLTAR PRÓXIMA

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

*Obrigatório

Experiência em projetos PHP

As perguntas a seguir são voltadas para seus conhecimento e experiência em projetos PHP.

Qual a média de idade dos projetos que participou? *

- ◯ Um ano ou menos;
- ◯ Dois a quatro anos;
- ◯ Cinco a dez anos;
- ◯ Mais de dez anos;

Quantas pessoas em média participaram dos projetos que participou? *

- ◯ Apenas eu;
- ◯ De duas a quatro pessoas;
- ◯ De cinco a oito pessoas;

- ◯ Mais de oito pessoas;

Dentre as correções do projeto, qual era a que mais teve o costume de realizar? *

- ◯ Manutenção Corretiva;
- ◯ Manutenção Adaptativa;
- ◯ Manutenção de Aperfeiçoamento;
- ◯ Não realizo manutenção em produtos de software;
- ◯ Outro;

Quais Frameworks, já utilizou em seus projetos? *

- ☐ CakePHP;
- ☐ CodeIgniter;
- ☐ Laravel;
- ☐ Limonade;
- ☐ Lumen;
- ☐ PhalconPHP;
- ☐ Silex;
- ☐ Slim;

- [ ] Symfony;
- [ ] Wave;
- [ ] Zend Explessive;
- [ ] Zend Framework;
- [ ] Não sei opinar;
- [ ] Não usava;
- [ ] Outros;

Você já utilizou o Framework Laravel em algum projeto sendo profissionalmente ou não? *

- ( ) Sim;
- ( ) Não;

VOLTAR PRÓXIMA

Nunca envie senhas pelo Formulários Google.

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

*Obrigatório

Laravel Framework

As perguntas a seguir são relacionadas a sua experiência com a ferramenta Laravel Framework.

Quais motivos o levaram a utilizar o Laravel Framework: *

- [ ] Atual popularidade do Framework;
- [ ] Comunidade maior o que leva a um maior auxilio na utilização;
- [ ] Facilidade de aprendizado na sua utilização;
- [ ] Facilidade no reaproveitamento de código;
- [ ] Fato do framework ser Livre;
- [ ] Organização, estrutura de código e arquivos;
- [ ] Possuir documentação de fácil acesso e entendível;
- [ ] Recomendação de amigos ou empresa;
- [ ] Relacionamento de entidades;

- [ ] Testes mais intuitivos;
- [ ] Utilização de Blades ou Sistema de Templates em um modelo mais flexível;
- [ ] Utilização de arquivo de rotas;
- [ ] Utilização de Eloquent e QueryBuilder;
- [ ] Utilização do Artisan Console;
- [ ] Outro: _____

Há quanto tempo conhece a ferramenta Laravel Framework? *

- ◯ Um ano ou menos;
- ◯ Dois a quatro anos;
- ◯ Cinco a dez anos;
- ◯ Mais de dez anos;

Na sua opinião, em uma escala de 1 a 10, com que frequência utiliza o Laravel Framework? *

	1	2	3	4	5	6	7	8	9	10	
Nunca	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Sempre

Na sua opinião, em uma escala de 1 a 10 qual o nível de dificuldade encontrado no processo de aprendizado do Framework? *

	1	2	3	4	5	6	7	8	9	10	
Muito fácil	○	○	○	○	○	○	○	○	○	○	Muito difícil

Houve algum framework que têm se demonstrado mais adequado em relação aos projetos com os quais têm trabalhado? O qual tenha te motivado a deixar de utilizar o Laravel. *

- ○ Sim;
- ○ Não;

VOLTAR PRÓXIMA

Nunca envie senhas pelo Formulários Google.

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

*Obrigatório

Outro Framework

As perguntas a seguir são relacionadas a sua experiência com o framework citado na última questão.

Qual o Framework utilizado? *

- ◯ CakePHP;
- ◯ CodeIgniter;
- ◯ Laravel;
- ◯ Limonade;
- ◯ Lumen;
- ◯ Phalcon;
- ◯ Silex;
- ◯ Slim;
- ◯ Symfony;
- ◯ Wave;

- ◯ Zend Explessive;
- ◯ Zend Framework;
- ◯ Outro:

Quais motivos o levaram a utilizar do Framework: *

- ☐ Popularidade do Framework;
- ☐ Comunidade maior que pode auxilia na utilização;
- ☐ Facilidade de aprendizado na sua utilização;
- ☐ Facilidade no reaproveitamento de código
- ☐ Fato do framework ser Livre;
- ☐ Organização, estrutura de código e arquivos;
- ☐ Possuir documentação de fácil acesso e entendível;
- ☐ Recomendação de amigos ou empresa;
- ☐ Relacionamento de entidades;
- ☐ Framework com suporte para teste;
- ☐ Utilização de Sistema de Templates;
- ☐ Utilização de arquivo de rotas;
- ☐ Utilização do QueryBuilder;

- [ ] Utilização de interações via linha de comando;
- [ ] Outro: _____

Há quanto tempo conhece o Framework? *

- ◯ Um ano ou menos;
- ◯ Dois a quatro anos;
- ◯ Cinco a dez anos;
- ◯ Mais de dez anos;

Ne uma escala de 1 a 10, com que frequência utiliza este Framework? *

	1	2	3	4	5	6	7	8	9	10	
Nunca	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Sempre

Na sua opinião, qual o nível de dificuldade encontrado no processo de aprendizado do framework? *

	1	2	3	4	5	6	7	8	9	10	
Nunca	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Sempre

Segundo Sommervile (2011) um dos atributos essenciais para um software é a Manutenibilidade, que é a capacidade do produto de ser modificado para atender as necessidades de seus clientes. Na sua opinião em uma escala de 1 a 10 quanto o este Framework auxilia na obtenção deste atributo? *

	1	2	3	4	5	6	7	8	9	10	
Nunca	○	○	○	○	○	○	○	○	○	○	Sempre

VOLTAR PRÓXIMA

Nunca envie senhas pelo Formulários Google.

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

*Obrigatório

Manutenibilidade de Software

As perguntas a seguir são relacionadas aos projetos que desenvolveu envolvendo Laravel Framework no quesito manutenibilidade de software.

De modo geral, como você considera seu nível de conhecimento acerca de manutenção de software? *

	1	2	3	4	5	6	7	8	9	10	
Pouco experiente	○	○	○	○	○	○	○	○	○	○	Muito experiente

Com que frequência é considerada a preocupação do desenvolvimento de um sistema com analisabilidade? *

- ○ Em todos os casos;
- ○ Na maioria dos casos;
- ○ Eventualmente;
- ○ Raramente;

- ◯ Nunca;
- ◯ Outro: _____

Com que frequência é considerada a preocupação do desenvolvimento de um sistema com modificabilidade? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Nunca;
- ◯ Outro: _____

Com que frequência é considerada a preocupação do desenvolvimento de um sistema com estabilidade? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Nunca;
- ◯ Outro: _____

Com que frequência é considerada a preocupação do desenvolvimento de um sistema com testabilidade? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Nunca;
- ◯ Outro: _____

Com que frequência é considerada a preocupação do desenvolvimento de um sistema com conformidade relacionada à manutenibilidade? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Nunca;
- ◯ Outro: _____

VOLTAR PRÓXIMA

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

*Obrigatório

Laravel Framework e Manutenibilidade

As perguntas a seguir relacionam informações sobre suas experiências com Laravel e manutenibilidade de software.

O fato do Laravel ter recursos como o Artisan Console que buscam auxiliar no desenvolvimento de sistemas, estabelecendo uma padronização e organização na criação de arquivos, pode se dizer que ele auxilia na analisabilidade de código fonte? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Não sei opinar;
- ◯ Nunca;
- ◯ Outro:

A utilização de recursos como arquivo de rotas e templates ou blades no Laravel auxilia no processo de reutilização de código fonte? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Não sei opinar;
- ◯ Nunca;
- ◯ Outro: _____

O fato do Laravel ser construído com suporte para testes com o PHPUnit, auxilia na estabilidade e testabilidade de recursos do sistema evitando assim efeitos inesperados decorrentes de modificações no software? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Não sei opinar;
- ◯ Nunca;

- ◯ Outro: _____

Pode se dizer que a utilização do Eloquent no Laravel auxilia no processo de implementação e manutenção de código fonte? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Não sei opinar;
- ◯ Nunca;
- ◯ Outro: _____

Pode se dizer que a utilização das Migrations, auxilia na construção, modificação e controle de versão do banco de dados? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Não sei opinar;
- ◯ Nunca;

- ◯ Outro: _____

25.Segundo Sommervile (2011) um dos atributos essenciais para um software é a Manutenibilidade, a capacidade do produto de ser modificado para atender as necessidades de seus clientes. Na sua opinião o Laravel Framework auxilia na obtenção deste atributo? *

- ◯ Em todos os casos;
- ◯ Na maioria dos casos;
- ◯ Eventualmente;
- ◯ Raramente;
- ◯ Nunca;
- ◯ Outro: _____

VOLTAR PRÓXIMA

Estudo dos benefícios trazidos na utilização de Laravel Framework em manutenibilidade de software

Agradecimentos

Obrigado por participar desta pesquisa, o conhecimento compartilhado comigo será de grande ajuda.

O conhecimento adquirido é como uma semente lançada em solo fértil: até pode demorar algum tempo, mas acaba sempre por dar fruto. (Autor Desconhecido)

VOLTAR ENVIAR

Nunca envie senhas pelo Formulários Google.