MATHEUS FERREIRA SILVA

UTILIZAÇÃO DOS MODELOS *FULL-STACK FRAMEWORK* E *MICRO-FRAMEWORK* PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB ESCALÁVEIS EM LINGUAGEM PHP

BACHARELADO

EM

CIÊNCIA DA COMPUTAÇÃO

FIC – MINAS GERAIS
2016

MATHEUS FERREIRA SILVA

UTILIZAÇÃO DOS MODELOS *FULL-STACK FRAMEWORK* E *MICRO-FRAMEWORK* PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB ESCALÁVEIS EM LINGUAGEM PHP

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 Msc. Glauber Luiz Costa.

FIC – CARATINGA
2016

MATHEUS FERREIRA SILVA

UTILIZAÇÃO DOS MODELOS *FULL-STACK FRAMEWORK* E *MICRO-FRAMEWORK* PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB ESCALÁVEIS EM LINGUAGEM PHP

Monografia submetida à Comissão examinadora designada pelo Curso de Graduação em Ciência da Computação como requisito para obtenção do grau de Bacharel.

Profº. Msc. Glauber Luiz Costa
Faculdades Integradas de Caratinga

Profº. Jonilson Batista Campos
Faculdades Integradas de Caratinga

Profº. Wanderson Miranda Nascimento
Faculdades Integradas de Caratinga

Caratinga, 14 / 12 / 2016

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me concebido forças nesta etapa. Aos meus pais Dair Ferreira e Gerson Santana, pelo incentivo ao estudo e por não terem medido esforços para me proporcionar condições para que eu pudesse prosseguir nesta jornada.

Agradeço a minha irmã Mariana, minha namorada Luana, toda minha família, colegas e amigos que contribuíram de forma direta e indireta e por terem se submetido à minha ausência para que este momento se tornasse possível.

Agradeço também a todos meus professores pelos ensinamentos concebidos nesses anos, em especial ao meu orientador, amigo e patrão Glauber Costa pela disposição em ajudar e ter me guiado durante o processo de desenvolvimento deste trabalho.

Agradeço à todos meus colegas de turma que foram pessoas ao qual tive o prazer de conhecer e compartilhar momentos marcantes que permanecerão a vida toda. Em especial gostaria de agradecer à um grupo que se formou durante a faculdade, "*Black Block Solutions*", ao qual fazem parte além de mim, Edvaldo Rodrigues, Mateus Martins e Jehomaks Philipe, estas pessoas foram cruciais para mudar traços de minha personalidade e me fazer uma pessoa melhor.

Gostaria de agradecer também a Elton Minetto, uma referência na comunidade PHP que se dispôs a responder dúvidas e esclarecimentos além de prestar uma grande ajuda na obtenção de dados para o desenvolvimento deste trabalho.

A todos o meu sincero muito obrigado!

RESUMO

Escalabilidade é uma característica desejável em toda aplicação, se trata da capacidade de um *software* "estar preparado para crescer". Aplicações escaláveis estão dentre as mais complexas, justamente pelas várias considerações a serem feitas ao iniciar o desenvolvimento de uma.

No contexto de desenvolvimento de aplicações *web*, existem dois modelos de *frameworks*: *micro-frameworks* e *full-stack frameworks*. O modelo *full-stack* possuem uma variedade de componentes com o objetivo de revolver distintos problemas no desenvolvimento de uma aplicação. Por outro lado, o modelo *micro* são projetados na resolução de problemas específicos acompanhados de menos componentes embarcados. Partindo-se dessa premissa há uma discussão na comunidade PHP (*Hypertext Preprocessor*) sobre o uso dessa ferramenta para a construção de aplicações *web* complexas, levando como principal argumento o fato de possuírem menos componentes embarcados, auxiliando na autonomia dos envolvidos no desenvolvimento.

Tendo em vista este cenário, o presente trabalho teve como objetivo efetuar uma análise desta hipótese, realizando um comparativo da construção de aplicações *web* utilizando *micro-frameworks* e *full-stack frameworks* em linguagem PHP, afim de se identificar qual dos dois modelos de *frameworks* melhor se adequa ao desenvolvimento de aplicações *web* escaláveis sobre os critérios de manutenibilidade e desempenho.

Foi elaborado e aplicado um questionário tendo como base a pesquisa bibliográfica realizada, buscando identificar características de manutenibilidade na fase de início do desenvolvimento da aplicação, durante sua fase de construção e após seu desenvolvimento. Além do objetivo de identificar índices de desempenho após a aplicação ter sido concluída.

Dessa forma foi possível concluir que sobre o critério de desempenho, o modelo *micro-framework* apresentou melhores índices em relação ao modelo *full-stack*. Sobre o quesito de manutenibilidade, houveram indícios de declínio em aplicações que utilizaram o modelo *full-stack* em relação à aquelas que fizeram uso de *micro-framework*. Porém aplicações que tiveram um planejamento de arquitetura alcançaram altos níveis de flexibilidade e desempenho em ambas os modelos de *frameworks*.

Palavras chaves: Escalabilidade, *Micro-framework*, *Full-stack framework*, Desempenho, Manutenibilidade, Desempenho.

LISTA DE SIGLAS

API - Application Programming Interface (Interface de Programação de Aplicativos)

MVC - Model-View-Controller (Modelo Visão Controlador)

ORM - Object-relational Mapping (Mapeamento objeto-relacional)

PHP - Hypertext Preprocessor (Pré-processador de hipertexto)

REST - Representational State Transfer (Transferência de Estado Representacional)

TI – Tecnologia da Informação

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

A internet deixou de ser apenas um mecanismo científico e acadêmico para se tornar um meio de entretenimento, comunicação, transporte de informações e interação. Ainda em pouco tempo, esta ferramenta passou a ser utilizada para o fomento de atividades rotineiras, empresariais e científicas.

Aplicações *web* tem se tornado o foco principal de empresas quando o assunto é o desenvolvimento de um novo *software*, entre as razões se destacam os requisitos mínimos para acesso, sendo um navegador com acesso à internet. Com o aumento gradativo do uso de internet no mundo, aplicações *web* possuem o desafio de manterem-se estáveis e disponíveis enquanto o número de usuários e requisições aumenta de forma elevada, o que chamamos de escalabilidade (NEUMAN, 1994). Tal objetivo, pode ser alcançado planejando a estrutura do *hardware* usado na disponibilização da aplicação. Atualmente existem dois modelos principais que suprem as necessidades e são comumente utilizados para este fim, escalabilidade horizontal (*scale out*) e escalabilidade vertical (*scale up*).

No modelo horizontal, são vários computadores de pequeno porte conectados em uma única rede onde a aplicação é subdividida, novos computadores são adicionados na medida que aumenta o número de requisições. O modelo vertical, acrescenta recursos como memória, CPUs e disco de armazenamento na mesma máquina na medida que cresce as solicitações dos usuários.

Conforme descrito, esses dois modelos atendem as necessidades atuais; mas existem pontos importantes que se melhor tratados podem resultar em ganho na qualidade do *software*, além de trazer possível redução de custos. Esses pontos estão relacionados à arquitetura do *software* e serão abordados no trabalho.

Para Smith e Williams (2000), decisões de arquitetura de *software*, são as mais importantes que um desenvolvedor irá fazer ao projetar sua aplicação *web*. Isto porque a arquitetura de uma aplicação, impacta diretamente na qualidade e no futuro da mesma, pois, partindo-se do ponto em que requisitos de *software* mudam, uma arquitetura mal planejada pode vir a comprometer e dificultar mudanças necessárias.

Durante o processo de desenvolvimento de uma aplicação *web*, é comum que os envolvidos no projeto utilizem ferramentas para agilizar a construção do *software*, dentre essas

os *frameworks* são habitualmente utilizados. Os pontos flexíveis de um *framework* são 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. Markiewicz e Lucena (2000) destacam que mesmo que um *framework* seja composto por *hot spots*, alguns aspectos são impossíveis de serem flexíveis, assim algumas características não são mutáveis e não podem ser facilmente alteradas, tais pontos são chamados de *frozen spots* e constituem o núcleo (*kernel*) do *framework*.

No cenário de desenvolvimento *web* existem dois modelos de *frameworks*. Os *full-stack frameworks*, que são constituídos por vários componentes buscando tratar diferentes problemas em uma aplicação, e os *micro-frameworks*. Estes são projetados com foco na resolução de problemas específicos e não impõem uma certa estrutura para iniciar o desenvolvimento.

Partindo-se dessa premissa, existe uma discussão na comunidade PHP com a hipótese de que o uso de *micro-frameworks* para a construção de aplicações escaláveis seria mais adequado, levando como principal argumento o fato dos *frameworks* desta categoria incluírem menos componentes embarcados e não conterem uma estrutura base em seu alicerce para iniciar o desenvolvimento de uma aplicação, dessa forma tomam pequenas decisões impactando menos em mudanças futuras, permitindo que os envolvidos utilizem realmente o que é necessário para a aplicação.

Este trabalho tem como objetivo realizar uma análise comparativa sobre pontos de escalabilidade na construção de aplicações *web* que utilizem os modelos *full-stack framework* e *micro-framework* em linguagem PHP, para que se possa chegar à conclusão de qual destes modelos de *framework* melhor se adequa ao desenvolvimento de uma aplicação *web* escalável.

Após o estudo de revisão bibliográfica, foi definido que os pontos de escalabilidade a serem analisados se limitariam a manutenibilidade e desempenho, esses pontos estão dentre as características mais importantes de uma aplicação escalável.

Foi aplicado um questionário com intuito de coletar dados de aplicações que fizessem uso dos modelos de *frameworks* abordados, afim de se realizar uma análise comparativa entre a qualidade das aplicações sobre os pontos de escalabilidade delimitados. Um total de 77 entrevistados participaram da pesquisa.

A pesquisa está estruturada em três capítulos principais. O capítulo 1 contém a descrição dos assuntos abordados, descrevendo o conceito de aplicações *web*, a linguagem PHP, arquitetura de *software* e sua importância, qualidade de *software*, uma concepção sobre

escalabilidade, além da definição de *frameworks* e de seus respectivos modelos abordados na pesquisa. O capítulo 2 apresenta os métodos abordados para alcançar os resultados. O capítulo 3 descreve os resultados obtidos realizando uma associação dos dados coletados afim de extrair melhores informações. Por fim são apresentadas as conclusões geradas pela pesquisa, além de sugestões para futuros trabalhos.

1 REFERENCIAL TEÓRICO

Esta seção tem como objetivo introduzir os conceitos abordados no trabalho para proporcionar uma melhor compreensão da pesquisa realizada. Descrevendo o conceito de aplicações *web* e a linguagem PHP. Além de dissertar sobre a importância da arquitetura no desenvolvimento de *software* e como ela está diretamente relacionada com os requisitos não-funcionais desempenho e manutenibilidade, e a qualidade final do *software*.

A seção também procurou explicar o conceito de escalabilidade e como esta característica é desejável para uma aplicação, principalmente no contexto da *web*, também foram descritos os principais atributos de qualidade ou requisitos não-funcionais que estão relacionados com a escalabilidade. Assim como explicar a definição de *frameworks* e sua relação com o desenvolvimento de aplicações *web*, também foram apresentados os conceitos e diferenças existentes sobre os modelos *full-stack* e *micro-frameworks*.

1.1 APLICAÇÕES WEB

Em (OLIVEIRA et al., 2005; p. 2), a definição de aplicação *web* é descrita como:

> [...] todo o conjunto de programas que implementa um qualquer Sistema de Informação seguindo o paradigma Cliente/Servidor suportado pelo protocolo de comunicação HTTP e cuja camada interativa está escrita em HTML de modo a que a interface com o utilizador seja assegurada pelos browsers tradicionalmente criados para navegação na rede de hiperdocumentos W³.

Segundo Pressman e Maxim (2016), no início da *World Wide Web* (WWW ou W³), por volta de 1990 a 1995, os sites eram formados por apenas conjuntos de hipertextos estáticos e interligados, fornecendo informações aos usuários utilizando recursos limitados. Com o tempo a evolução da linguagem HTML e ferramentas, permitiu aos desenvolvedores fornecerem poder computacional junto com maiores informações.

Para Mazza (2012), a expansão da internet pelo mundo, se explica devido aos grandes avanços da banda larga e diversos outros dispositivos capazes de navegar pela rede, como celulares, *tablets*, televisores e videogames.

Aplicações *web* tem se tornado o foco principal de empresas quando o assunto é o desenvolvimento de um novo *software* por diversas razões, dentre elas a facilidade no acesso se destaca por possuírem como requisito mínimo um navegador instalado no sistema operacional e acesso à internet.

Para Ginige e Murugesan (2001a), embora o desenvolvimento de aplicações *web* possa parecer fácil, muitas vezes é mais complexo e desafiador. De diversas formas, ele é diferente do desenvolvimento de *software* tradicional e acrescenta desafios (GINIGE; MURUGESAN, 2001b).

1.2 PHP

Em Lisboa (2012), PHP (**P**HP: ***H**ypertext **P**reprocessor*) é descrito uma linguagem de programação que possui tipagem fraca e dinâmica, interpretada em tempo de execução e utilizada geralmente para construir aplicações *web*. Segundo uma pesquisa realizada pelo site BuiltWith Trends, em 14/11/2016, PHP está presente na maior parte dos servidores *web*, a Figura 1 exibe a pesquisa.

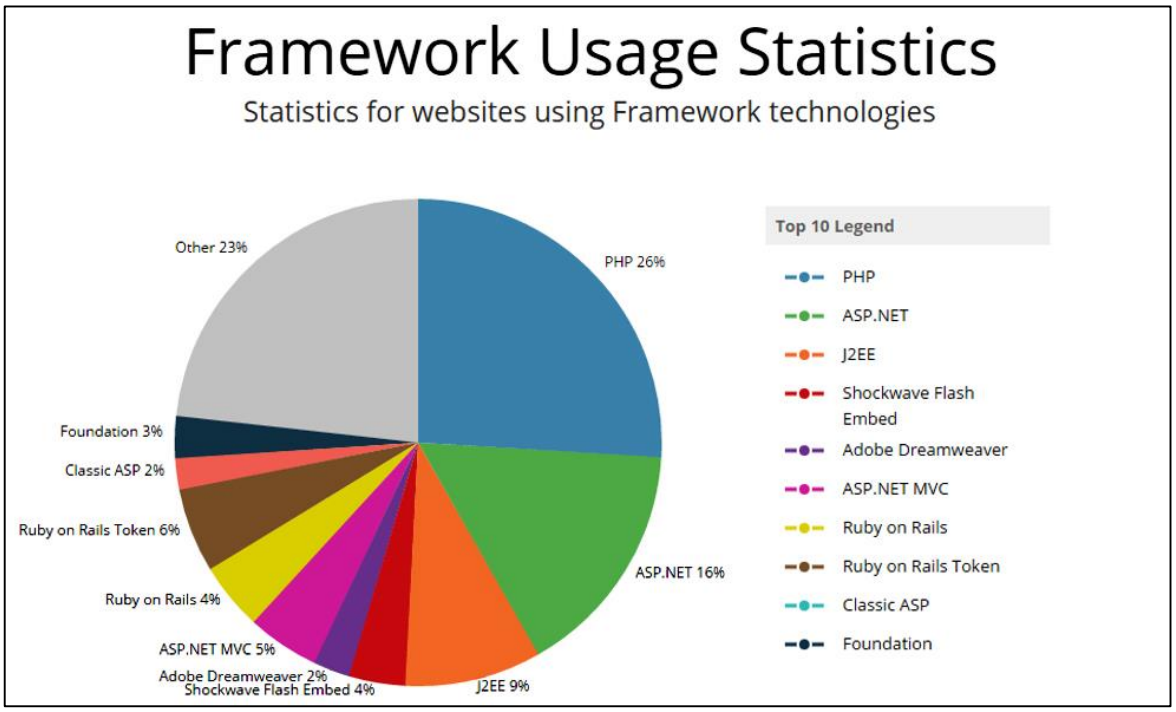

Figura 1 - Estatística de uso de *framework* por *websites*
Fonte: (“BuiltWith Trends”, 2016)

A linguagem PHP é amplamente conhecida pelos desenvolvedores devido sua ampla utilização, flexibilidade e facilidade de uso; é o que destaca Loudon (2010). Segundo sua documentação, a linguagem possui uma enorme portabilidade, podendo ser utilizada em diferentes arquiteturas de computador, possibilitando a utilização de programação estruturada, orientada a objetos, ou ambas.

PHP "é uma ferramenta que possibilita o pré-processamento de páginas HTML" (BENTO; 2014; p. 3). Dessa forma, PHP processa o conteúdo de uma página, antes da mesma ser enviada para o navegador para ser exibida ao usuário, ela também torna possível capturar entrada de dados do usuário para serem processados. Ainda segundo Bento (2014), PHP nasceu para a *web* e sua integração com servidores *web* é simples.

Segundo uma pesquisa de ranking, com as linguagens de programação mais populares do ano, realizada pela RedMonk (2016) em janeiro deste ano, PHP aparece como a terceira linguagem mais utilizada, ficando atrás apenas de JavaScript e Java na sequência; porém como JavaScript é uma linguagem comumente utilizada no *front-end* de aplicações *web* e Java na construção de aplicações desktop, podemos concluir com essa pesquisa que PHP é a linguagem mais utilizada no *back-end* de aplicações *web*. A pesquisa compara movimentações no Github, uma aplicação *web* que possibilita a hospedagem de repositórios Git para auxiliar no desenvolvimento de *software* colaborativo além de funcionar como uma rede social para desenvolvedores (AQUILES; FERREIRA, 2014), e Stack Overflow, um dos principais fóruns de discussão sobre programação.

A Figura 2, exibe as linguagens mais utilizadas até janeiro do ano de 2016, segundo a Redmonk (2016).

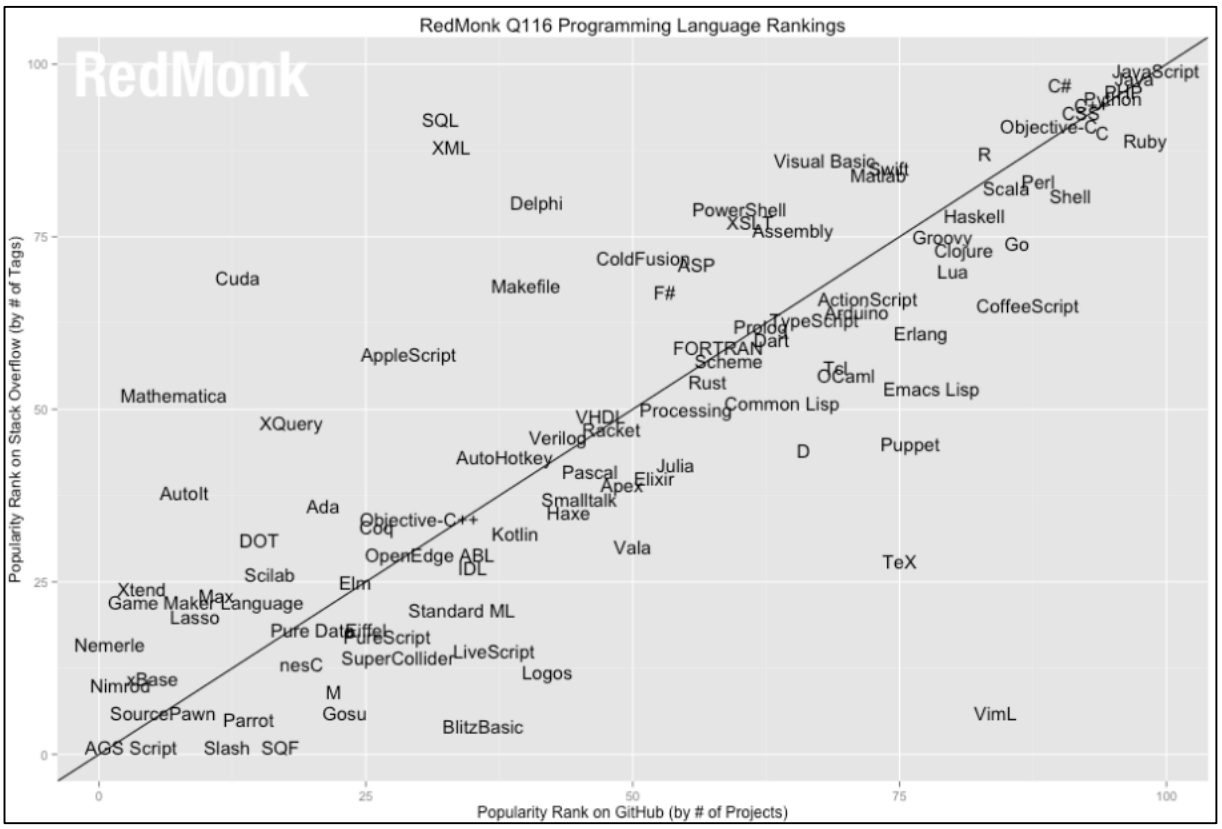

Figura 2 - Ranking das linguagens mais utilizadas no ano
Fonte: (REDMONK, 2016)

Segundo Lisboa (2012), uma forte característica da linguagem PHP é que ela adere a arquitetura *web* ao invés de resistir a ela; também é flexível o bastante para servir a diversos propósitos, pois abrange tanto a área de desenvolvimento quanto de infraestrutura de sistemas, pois além de ser utilizada para construir aplicações, também pode ser usada para automatizar tarefas.

1.3 ARQUITETURA DE SOFTWARE

Na medida que o *software* se torna cada vez mais ubíquo, seu tamanho e complexidade tem aumentado significativamente no mesmo passo. Isto se refere ao nível de organização do mesmo, ou, mais precisamente, o nível de arquitetura do software.

Perry e Wolf (1992) introduziram sua definição para arquitetura de *software* em um artigo seminal, utilizando a seguinte fórmula com a explicação dos termos:

$$Arquitura = \{Elementos, Organização, Decisões\}$$

De acordo com essa definição de arquitetura de *software*, se trata de um conjunto de elementos arquiteturais que possuem alguma organização definidos por decisões tomadas em prol de objetivos e restrições.

Baseando-se em discussões realizadas no *Software Engineering Institute* da *Carnegie Mellon University*, David Garlan e Dewayne Perry definiram arquitetura de *software* como, "a estrutura dos componentes de um programa/sistema, suas inter-relações, princípios e diretrizes que regem sua concepção e evolução ao longo do tempo" (GARLAN; PERRY, 1995; p. 1). Os mesmos destacam que a arquitetura de *software* se faz necessária quando o tamanho e a complexidade dos sistemas crescem.

Jeff Tyree e Art Akerman (2005), ressaltam que arquiteta de *software* é algo mais, ela é milhares de decisões, tanto grandes como pequenas. Pressman e Maxim completam dizendo que "algumas dessas decisões são tomadas no início do projeto e podem ter impacto profundo sobre todas as ações subsequentes" (PRESSMAN; MAXIM, 2016; p. 253).

Em um livro dedicado à arquitetura de *software* Bass, Clements e Kazman (2003) identificam três razões principais do porque a arquitetura de *software* é importante:

- A arquitetura de *software* fornece uma representação que facilita a comunicação e entendimento entre as partes interessadas;
- A arquitetura destaca desde o início as decisões de projeto que terão um grande impacto no trabalho de engenharia de *software* que se segue durante seu desenvolvimento;
- A arquitetura constitui um modelo relativamente pequeno e intelectualmente palpável de como um sistema é estruturado e como seus elementos trabalham juntos;

Segundo Sommerville (2012), os requisitos de *software* são classificados como requisitos funcionais e requisitos não-funcionais. O mesmo descreve como requisitos funcionais, serviços que o sistema/*software* deve fornecer ou fazer, e requisitos não-funcionais como restrições sobre tais serviços, incluindo restrições de tempo, de desenvolvimento, processos e limitações impostas por normas. Uma descrição mais clara e objetiva é encontrada

em (MENDES, 2002; p. 38), onde descreve requisito funcional, como “um requisito de *software*/sistema que especifica uma função que o *software*/sistema ou componente de *software*/sistema deve ser capaz de realizar”, e requisito não-funcional como aquele que descreve não o que o *software* fará, mas sim como ele fará.

1.3.1 Requisitos Não-Funcionais e Arquitetura de Software

Segundo Germoglio (2010), um *software* deve apresentar atributos de qualidade que atendam aos seus requisitos. Seu único objetivo é atender a seus requisitos; por sua vez, a arquitetura de *software* contém a descrição de como o mesmo alcança os atributos de qualidade.

Perceba que ambos os tipos de requisitos são importantes para o desenvolvimento de um sistema/*software*; entretanto os requisitos não-funcionais, também chamados de atributos de qualidade, possuem um papel muito importante durante o desenvolvimento de um sistema, atuando como critérios de seleção para composição de uma arquitetura de *software*. Requisitos arquiteturais compreendem os requisitos não-funcionais e atributos de projeto associados à arquitetura do sistema a ser desenvolvido (MENDES, 2002).

Brooks (1987) frisa que desconsiderar uma boa compreensão de tais requisitos é reconhecidamente oneroso e torna difícil a correção uma vez que o sistema foi implementado.

Ao desenvolver um novo sistema assim como projetar sua arquitetura, projetistas ou engenheiros de *software* expõem uma série de atributos de qualidade que o sistema deveria suportar. Exemplos desses atributos são desempenho, portabilidade, manutenibilidade e escalabilidade. Assim a arquitetura do *software* deveria fornecer suporte a tais requisitos não-funcionais. Isso revela a associação presente entre arquitetura de *software* e requisitos não funcionais (MENDES, 2002).

1.4 QUALIDADE DE SOFTWARE

A necessidade de maior qualidade de *software* surgiu a partir do momento que o *software* ficou cada vez mais integrado a todas as atividades de nossas vidas (PRESSMAN; MAXIM, 2016).

Levinson (2001) publicou um artigo com o título “Let's Stop Wasting $78 Billion a Year” (Chega de desperdiçar US$ 78 bilhões por ano), sobre o fato das empresas americanas gastarem bilhões em *softwares* que não faziam o que supostamente deveriam fazer.

A *ComputerWorld* lamenta que:

> *Software* de má qualidade está praticamente em todas as organizações que usam computadores, provocando horas de trabalho perdidas durante o tempo em que a máquina fica parada, dados perdidos ou corrompidos, oportunidades de vendas perdidas, custos de suporte e manutenção de TI elevados e baixa satisfação do cliente. (HILDRETH, 2005; p. 1)

Qualidade de *software* pode ser definida como, “uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que produzem e para aqueles que utilizam” (PRESSMAN; MAXIM, 2016, p. 414). Ainda segundo Pressman e Maxim (2016), esta definição tem por objetivo enfatizar três pontos importantes:

- Uma gestão de qualidade efetiva estabelece uma base para servir de suporte na construção de um *software* de qualidade.
- Um produto útil fornece aquilo que foi solicitado pelo usuário, além de confiabilidade e isenção de erros.
- Agrega valor tanto para o fabricante quanto para o usuário, pois uma vez que um *software* de qualidade foi entregue, o fabricante gastará menos tempo com manutenção, podendo focar no desenvolvimento de novos sistemas e ao usuário uma vez que terá a capacidade de agilizar algum processo do seu negócio

Pressman (2011) salieta que a análise e gestão de riscos no desenvolvimento de *software*, são ações que podem ajudar a equipe a trabalhar com a incerteza durante seu desenvolvimento e contribuir para um *software* com qualidade.

Para Sommerville (2012) há pelo menos seis tipos de riscos que podem ser incluídos no controle de riscos para o desenvolvimento de *software*:

- Riscos de tecnologia: derivam das tecnologias de *software* ou *hardware* que são usados para desenvolver o sistema.
- Riscos de pessoas: estão relacionados com as pessoas da equipe de desenvolvimento.
- Riscos organizacionais: derivam do ambiente organizacional onde o *software* está sendo desenvolvido.
- Riscos das ferramentas: derivam das ferramentas de *software* e outros *softwares* de suporte utilizados para o desenvolvimento do sistema.
- Riscos de requisitos: derivam das alterações dos requisitos do cliente e do processo de gestão da mudança de requisitos.
- Riscos de estimativas: estão diretamente relacionados a estimativas dos recursos necessários para construir o sistema de gestão.

É importante enfatizar o risco de tecnologia, já que se encontra diretamente ligado às escolhas das ferramentas que fazem parte ou que são utilizadas no *software*, demonstrando a importância de realizar escolhas conscientes para que tal escolha não venha a prejudicar a vida futura do sistema.

McCall, Richards e Walters (1977), fizeram uma proposta para categorização de fatores que afetam a qualidade de *software*. Estes fatores de qualidades se concentram em três pontos importantes de um produto de *software*: características operacionais (Fatores da Operação do Produto), capacidade de suportar mudanças (Fatores da Revisão do Produto) e a adaptabilidade a novos ambientes (Fatores da Transição do Produto). Abaixo são descritas as subcategorias que se enquadram em cada fator proposto.

- Fatores da Operação do Produto
    - Correção: o quanto o *software* atende a sua especificação e satisfaz aos objetivos da missão do cliente.
    - Confiabilidade: o quanto o *software* realiza uma função pretendida com a precisão requerida/exigida.
    - Eficiência: quantidade de recursos computacionais e código necessários para um *software* realizar uma função.
    - Integridade: o quanto o acesso ao *software* ou a informações por pessoas não autorizadas pode ser controlado.
    - Usabilidade: esforço necessário para entender, operar, preparar a

entrada de dados e interpretar a saída de um *software*.

- Fatores da Revisão do Produto
    - Manutenibilidade: esforço necessário para localizar e corrigir um erro em um programa.
    - Flexibilidade: esforço necessário para modificar um programa em funcionamento.
    - Testabilidade: esforço necessário para testar um programa para garantir que ele realize sua função pretendida.
- Fatores da Transição do Produto
    - Portabilidade: esforço necessário para transferir um sistema a partir de uma configuração de hardware e/ou *software* de um ambiente para outro.
    - Reusabilidade: o quanto um programa ou parte de um, pode ser usado em outra aplicação – relacionado com o empacotamento e o escopo das funções que o mesmo realiza.
    - Interoperabilidade: esforço necessário para integrar um sistema a outro.

1.4.1 Manutenibilidade

Manutenção de *software* é um termo geralmente empregado quando se refere a modificações realizadas no sistema depois de sua disponibilização para uso. Porém, na realidade o termo manutenibilidade é mais abrangente, pois envolve tanto atividade de reparo de algum erro no *software*, quanto também a atividades de alteração ou evolução de características ou acréscimo de novas que não constavam no projeto inicial (MENDES, 2002). Ainda segundo Mendes (2002), a manutenibilidade é um dos requisitos mais relacionados com a arquitetura de um sistema de *software*.

1.4.2 Desempenho

Segundo Fowler em Fowler (2006), muitas decisões de arquitetura, dizem respeito ao desempenho. Ainda segundo o mesmo, desempenho pode significar tanto *throughtput* quanto tempo de resposta:

- *Throughtput* é a quantidade de coisas que algo pode fazer em uma dada quantidade de tempo. Se for contabilizado o tempo gasto na cópia de um arquivo, o *throughtput* poderia ser medido em *bytes* por segundo.
- Tempo de resposta é a quantidade de tempo que o sistema leva para processar uma solicitação externa. Pode ser uma ação na interface do usuário, como pressionar um botão, ou uma chama de API (*Application Programming Interface*) do servidor.

"Desempenho é um atributo de qualidade importante para sistemas de *software"* (MENDES, 2002, p. 50). Segundo o mesmo, este requisito se trata da velocidade de operação de um sistema de *software* e pode ser visto nos seguintes termos:

- Requisitos de resposta – Especificam o tempo de resposta de um *software* aceitável para usuários.
- Requisitos de processamento (*throughtput*) – Especifica a quantidade de dados que deveriam ser processadas num determinado período.
- Requisitos de temporização – Este requisito especifica quão rapidamente o sistema deveria coletar dados de entrada de sensores antes que próximas leituras de dados de entrada, sobrescrevam os anteriores.
- Requisitos de espaço – São considerados em alguns casos. Pode-se referir à memória principal ou a secundária. Um exemplo seria a memória principal ser considerada um requisito de desempenho para executar uma aplicação, já que está relacionada ao comportamento do sistema em tempo de execução.

1.5 ESCALABILIDADE

Com o crescente número de acesso à web, torna-se essencial desenvolver soluções eficientes e que estejam disponíveis enquanto ocorre o aumento de acessos simultâneos em um sistema web. Nesse contexto se enquadra a escalabilidade, pois é desejável que uma aplicação web seja projetada para suprir a demanda de acesso em grande fluxo.

Escalabilidade é um termo utilizado a bastante tempo para descrever uma propriedade de sistemas. Hill (1990) propôs um desafio a fim de se criar uma definição formal para escalabilidade. Após esse desafio, várias definições surgiram, tanto formais quanto informais, entretanto ainda hoje não há uma definição amplamente aceita.

Um ponto que dificulta uma definição exclusiva para escalabilidade, se explica pelo fato do tópico ser multidimensional, assim como a arquitetura de software (BACHMANN et al., 2002). Não é possível falar de escalabilidade sem considerar outros aspectos como manutenibilidade, desempenho, confiabilidade, segurança, disponibilidade e custo (DUBOC; ROSENBLUM; WICKS, 2007). Uma escassa escalabilidade pode resultar em mau desempenho do sistema, necessitando sua reengenharia ou duplicação (BONDI, 2000).

Bondi (2000) define escalabilidade em dois aspectos: escalabilidade estrutural e escalabilidade de carga. Segundo o mesmo, “escalabilidade estrutural é a habilidade de um sistema expandir-se em uma dimensão escolhida sem grandes modificações em sua arquitetura” (BONDI, 2000, p. 1), algo que vai bem de encontro a manutenibilidade da aplicação. Para escalabilidade de carga, Bondi define como a capacidade de um sistema em manter uma boa performance enquanto o mesmo é exposto a uma crescente massa de tráfego.

Um sistema é dito escalável se ele pode manipular a adição de usuários e recursos sem sofrer uma notável perda de desempenho ou aumento na complexidade administrativa (NEUMAN, 1994). Para Jogalekar, Woodside e Member (2000), escalabilidade significa não apenas a habilidade de operar, mas de operar com eficiência e com qualidade de serviço adequada, dentro de uma faixa de possíveis configurações.

Dessa forma, escalabilidade pode ser definida como a facilidade com que um sistema ou componente pode ser modificado para atender a área de um problema. Desempenho e escalabilidade são conceitos diferentes, um sistema pode ter um alto desempenho e não escalar, assim como também pode-se ocorrer o inverso. (HENDERSON, 2006).

No que se diz respeito à *hardware*, escalabilidade pode ser alcançada planejando a estrutura do *hardware* utilizado na disponibilização da aplicação. Atualmente existem dois modelos principais que suprem as necessidades e são comumente utilizados para este fim; escalabilidade horizontal (*scale out*) e escalabilidade vertical (*scale up*).

No modelo horizontal, são vários computadores de pequeno porte conectados em uma única rede onde a aplicação é subdividida; novos computadores são adicionados na medida que aumenta o número de requisições. O modelo vertical, acrescenta recursos como memória, CPUs e disco de armazenamento na mesma máquina na medida que cresce as solicitações dos usuários (PORTO, 2009).

1.6 FRAMEWORKS

Durante o processo de desenvolvimento de uma aplicação, é comum que os envolvidos no projeto utilizem ferramentas para agilizar a construção do *software*, dentre essas os *frameworks* são habitualmente utilizados. Segundo Buschmann (1996), Pree e Sikora (1997) e Pinto (2000), um *framework* é definido como um *software* parcialmente completo projetado para ser instanciado. O *framework* define uma arquitetura para uma família de subsistemas e oferece os construtores básicos para cria-los.

Para Fayad, Schmidt e Johnson (1999), um *framework* é um conjunto de classes que constitui um projeto abstrato para solução de uma família de problemas. Um *framework* também pode ser descrito sendo algo 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 *softwares*” (MINETTO, 2007, p 17).

Russel (2016), destaca alguns benefícios na utilização de *frameworks*:

- Uma estrutura definida que se torna familiar em todos os seus sites e aplicações;
- A contribuição da comunidade para o melhoramento do código do *framework*;
- Um conjunto de funcionalidades pré-definidas, sem a necessidade de reinventar para cada aplicação;
- Módulos, bibliotecas disponíveis para adicionar á aplicação;
- Melhor testabilidade;

- Integração com ORMs (*Object-relational mapping*);
- Uso pré-estabelecido de padrões de projeto;
- Contribui para reusabilidade e manutenibilidade do código;

Os pontos flexíveis de um *framework* são chamadas de *hot spots*, tal flexibilidade se refere a capacidade desses tais pontos/componentes serem adeptos a sofrerem modificações; ou em outras palavras, são pontos em que os envolvidos no desenvolvimento do *software* possuem grande autonomia. Markiewicz; Lucena (2000) salientam que embora um *framework* seja composto por *hot spots*, algumas características não são mutáveis e não podem ser facilmente alteradas, tais pontos são chamados de *frozen spots* e constituem o núcleo do *framework*.

Para Bustamante (2008) *frameworks* no domínio de desenvolvimento de aplicações *web*, são ferramentas que contém uma diversidade de funcionalidades prontas para serem usadas, essas funcionalidades geralmente tratam de problemas comuns que aplicações web possuem.

O desenvolvimento de aplicações *web* utilizando *frameworks* se expandiu de tal forma que se tornou imprescindível o uso de *frameworks* para construção de sistemas voltados para internet, sejam pequenos ou grandes sistemas (SOUZA, 2010).

No contexto de desenvolvimento de aplicações *web*, existem duas terminologias utilizadas para classificar modelos de *frameworks*, são elas *full-stack frameworks* e *micro-frameworks*. A classificação está relacionada principalmente a quantidade de componentes/funções que o *framework* fornece para iniciar o desenvolvimento de alguma aplicação. Carece de uma definição formal ou acadêmica para as duas terminologias, o conceito/definição empregada para distinguir as duas terminologias é um consenso na comunidade. No entanto, é de extrema importância que o leitor tenha conhecimento de tais conceitos, pois se tratam de pontos importantes na pesquisa.

1.5.1 Full-Stack Frameworks

O modelo *full-stack*, se enquadra nos conceitos de *frameworks* já descritos na pesquisa, principalmente na definição apresentada por Bustamante (2008), se tratam de *frameworks* que

fornecem diferentes componentes para solucionar diferentes problemas, porém comuns durante o desenvolvimento de uma aplicação web.

Geralmente são acompanhados de componentes que buscam realizar o básico de uma aplicação web, como tratamento de requisições e envio de respostas HTTP (*Hypertext Transfer Protocol*), além daqueles (componentes) que buscam realizar a comunicação com banco de dados, gerenciamento de sessão do usuário, autenticação e autorização, validação de formulários e geração de código HTML. Laravel, Symfony, Zend Framework e CakePHP são alguns exemplos de *framework* populares na comunidade deste modelo, em linguagem PHP. (RUSSELL, 2016)

Além disso, os *frameworks* desta categoria fornecem uma determinada estrutura para iniciar o desenvolvimento de aplicações com os mesmos, comumente essa estrutura segue um modelo de arquitetura MVC.

Segundo Turini (2015), a arquitetura MVC divide as aplicações em três camadas, *Model, View* e *Controller*:

- O *Model* é onde ficam as regras de negócio, as entidade e classes de acesso ao banco de dados.
- A camada *View* é a responsável por apresentar as páginas de resultados e interagir com o usuário, exibe o conteúdo processado que a aplicação envia para o navegador.
- *Controller* é o responsável por receber as requisições web e decidir o que fazer com elas. Nessa camada definimos quais *models* devem ser executados para determinada ação e para qual *view* será encaminhada a resposta/conteúdo. Em outras palavras, essa camada realiza a comunicação entre as outras camadas.

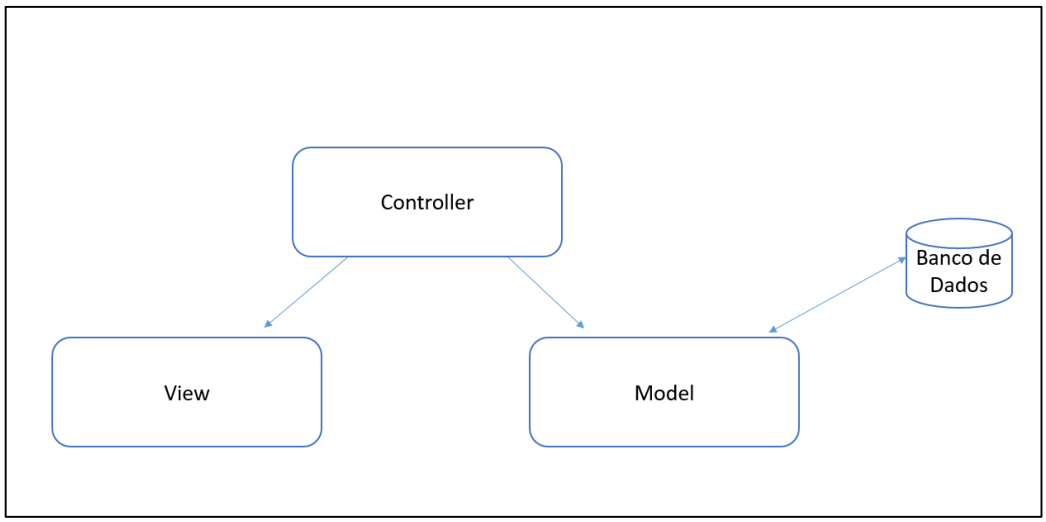

Figura 3: Fluxo do MVC
Fonte: Adaptado de (TURINI, 2015)

Porém, essa estrutura que geralmente acompanha os *frameworks* deste modelo, pode dificultar quando é necessário modifica-la, mesmo entre *frameworks* modulares, pois eles ainda definem a estrutura de módulos e classes que se deve seguir para desenvolver uma aplicação utilizando o *framework*. Por um lado isso é bom, já que auxilia na implantação de padrões durante o desenvolvimento, no entanto, em determinados cenários em que a arquitetura da aplicação não se beneficia com a estrutura do *framework*, isso pode não ajudar muito.

1.5.2 Micro-Frameworks

Micro-frameworks existem como alternativa para quando se deseja a estrutura e velocidade de desenvolvimento fornecidos por um *framework*, porém com menos componentes e menor *overhead* de um *full-stack framework*.(RUSSELL, 2016).

"Um *micro-framework* é um *framework* que procura fornecer apenas os componentes que são absolutamente necessários para que um desenvolvedor construa uma aplicação" (GIANNINI, 2015, p. 6). A quantidade de componentes fornecidas por esse modelo, é consideravelmente menor que de um *full-stack framework*, isso se explica pelo fato de serem construídos com foco na resolução de problemas específicos e em sua grande maioria simples. Exemplos de *frameworks* que se incluem nesta categoria em linguagem PHP são, Slim, Silex, Lumen e Zend Expressive.

Na própria documentação de *frameworks* desta categoria, conforme se observa na de Silex, Slim e Lumen, são descritos como ferramentas para auxílio no desenvolvimento de aplicações simples, APIs, *web services* em arquitetura *REST* (*Representational State Transfer*)[1], ou aplicações de um único arquivo. Este leque de opções para uso destas ferramentas, se explica pelo fato de não haver uma estrutura base vinculada ao *framework* para ser utilizado, conforme nota-se em Russell(2016), desta forma são independentes de arquitetura, o que auxilia na adequação às necessidades da aplicação a ser desenvolvida, permitindo aos envolvidos no projeto uma grande autonomia quanto a esse ponto.

Giannini(2015) e Russell(2016) também reforçam a ideia de que *micro-frameworks* tendem a se adequarem para o desenvolvimento de aplicações simples ou de pequeno porte, e mantêm a sugestão da utilização destes para o desenvolvimento de APIs. No entanto, como Russell(2016) mesmo destaca, nada impede de utilizá-los para a construção de aplicações de porte maior.

Apoiando-se nesta premissa, existe uma discussão ("Post The MicroPHP Manifesto", 2012) que se tornou um manifesto ("The MicroPHP Manifesto", 2012) na comunidade PHP, sobre a utilização de *micro-frameworks* para a construção de aplicações web. A discussão baseia-se no fato de que nem sempre é preciso do aglomerado de opções que um *full-stack framework* fornece, dos quais por muitas vezes não se fazem necessários para o domínio da aplicação, sendo fundamental algo mais adepto as necessidades/domínio da aplicação em desenvolvimento.

Nesse contexto, surgiu a hipótese de que a utilização de *micro-frameworks* para a construção de aplicações web escaláveis em linguagem PHP, torna-se mais factível em relação ao emprego de *full-stack frameworks* para este fim, já que o ambiente de desenvolvimento de aplicações escaláveis se faz dinâmico e que requisitos não funcionais importantes como manutenibilidade e desempenho seriam favorecidos com o emprego da ferramenta. Esta hipótese é o que busca responder este trabalho, utilizando os passos do próximo capítulo.

---

[1] *Rest* é um estilo de arquitetura usado principalmente para construir *web services*. Um serviço baseado em Rest é chamado de RestFul. Restful utiliza o protocolo HTTP como subjacente. (VAQQAS, 2014)

2 METODOLOGIA

O presente trabalho teve como objetivo analisar pontos de escalabilidade no desenvolvimento de aplicações *web*, utilizando os modelos de *framework micro-framework* e *full-stack framework*.

Após a realização do estudo através de revisão bibliográfica, decidiu-se que a análise do trabalho se limitaria a dois requisitos não-funcionais que são de suma importância a respeito de escalabilidade de *software*; são eles, manutenibilidade e desempenho.

Durante o desenvolvimento do trabalho para a realização de seu objetivo, uma das abordagens iniciadas foi o desenvolvimento de uma aplicação exemplo utilizando as duas terminologias de *frameworks*, os *frameworks* de cada categoria foram escolhidos com base em níveis de popularidade na comunidade PHP, são eles, Laravel e Slim dos modelos *full-stack* e *micro* em sequência. Porém, essa abordagem foi descartada após a percepção do autor e do orientador de que os resultados gerados através desta metodologia ficariam sujeitos a experiência do autor. Diante desse fato, a abordagem empregada no trabalho foi a utilização de um questionário, dessa forma, acredita-se que os resultados gerados sejam de maior imparcialidade já que o método abrange pessoas com diferentes níveis de conhecimento e experiências sobre aplicações escaláveis e as terminologias. O questionário se encontra anexado no final do trabalho como APÊNDICE 1 – QUESTIONÁRIO.

2.1 PÚBLICO ALVO DO QUESTIONÁRIO

O público alvo do questionário, são pessoas com o perfil que apresentam conhecimento e experiência no desenvolvimento de aplicações web escaláveis, assim como também experiência sobre as terminologias de *frameworks* abordadas no trabalho.

O questionário foi divulgado em grupos de discussão nas redes sociais Facebook e Google Plus, relacionados a programação para a web em geral, assim como também relacionados a linguagem PHP e grupos de *frameworks* específicos. O mesmo também foi enviado por e-mail diretamente para entrevistados específicos, ao qual a participação destes foi

considerada importante. Além disso a pesquisa também foi disponibilizada em uma *newsletter* semanal de um dos participantes ao qual foi enviado diretamente; nessa newsletter constam tópicos e *links* relacionados a linguagem PHP e foi de extrema relevância para encontrar entrevistados com o perfil buscado.

2.2 ELABORAÇÃO DO QUESTIONÁRIO

As questões contidas no questionário foram elaboradas com base no objetivo do trabalho e no conhecimento adquirido através das leituras realizadas.

Ao todo foram feitas trinta e cinco questões, sendo divididas em cinco grupos. Para tornar o processo de coleta de dados menos cansativo, o formulário foi dividido em oito páginas, de forma que a próxima página estava condicionada as respostas das anteriores, assim os participantes não precisavam responder a todas as perguntas e sim somente aquelas que se enquadravam ao seu perfil.

O formulário apresentava em seu cabeçalho uma explicação sobre a finalidade do mesmo e informava que os resultados da pesquisa seriam disponibilizados na comunidade para que desenvolvedores, arquitetos e quem mais tiver interesse possa fazer uso.

A maioria das questões que eram adequadas ao perfil do participante, eram obrigatórias e o mesmo só conseguia enviar o formulário após todas devidamente preenchidas/respondidas.

2.3 O QUESTIONÁRIO

Cada grupo de questões continha o objetivo de captar dados sobre um tipo ou conjunto de informações.

O primeiro grupo de questões busca coletar informações a respeito do participante, como e-mail (caso o mesmo queria receber a conclusão do trabalho), país, estado e idade para identificar o mesmo; formação acadêmica, cargo atual e tempo de atuação para saber sobre sua experiência profissional. O grupo ainda consta nas duas últimas perguntas, o objetivo de saber

o nível de conhecimento do participante sobre aplicações web escaláveis e se o mesmo já teve experiência em participar do desenvolvimento ou projeto de alguma aplicação web escalável, desenvolvida em linguagem PHP.

O segundo grupo é mostrado ao entrevistado caso o mesmo tenha participado do desenvolvimento de alguma aplicação web escalável desenvolvida em PHP. Nesta seção o participante é indagado sobre quantos projetos escaláveis teve contato e quantas pessoas em média foram envolvidas em cada projeto.

No terceiro grupo, o entrevistado é questionado se das aplicações que o mesmo participou do desenvolvimento, alguma fez uso de *micro-framework*. Se caso afirmativo, o restante das questões do grupo é exibido na próxima página. Tais questões subsequentes, visam coletar dados da aplicação desenvolvida relacionados a pontos de qualidade, como, a aptidão da arquitetura do *framework* utilizado para se adequar as necessidades da aplicação, a flexibilidade da arquitetura e desempenho da aplicação propriamente desenvolvida, além de qual *framework* e metodologia de desenvolvimento utilizados na construção da mesma. Caso o participante não tenha experiência com *micro-framework*, o mesmo é direcionado para o próximo grupo de questões.

No quarto grupo, se repete as mesmas perguntas, processo e técnica do grupo três, porém as perguntas são relacionadas a terminologia *full-stack*.

No quinto e último grupo, o participante é indagado baseado em sua experiência no desenvolvimento; com perguntas como, se o mesmo fosse realizar o redesenvolvimento da aplicação preparando-a para evolução, qual tipo de terminologia de *framework* utilizaria; qual o motivo da escolha e qual o *framework* seria escolhido.

2.4 COLETA DOS DADOS

O questionário foi elaborado criando um formulário utilizando a ferramenta Google Forms. Esta ferramenta possibilita a criação de pesquisas de forma online que podem ser facilmente compartilhadas e acessadas através de qualquer dispositivo conectado à internet.

O questionário foi disponibilizado a partir do dia 14 de setembro de 2016 às 02h00min e permaneceu disponível até o dia 14 do mês de outubro de 2016 às 02h00min, completando

um mês. Ao todo foram coletadas 77 respostas.

2.5 TRATAMENTO DE DADOS

A respostas coletadas pelo formulário foram armazenadas automaticamente em uma planilha criada pela própria ferramenta Google Forms.

Embora a ferramenta também disponibilize gráficos do resumo das respostas para que o pesquisador possa ter uma noção da trajetória que os resultados de sua pesquisa estão direcionando, não foram o suficiente para encontrar as respostas que o trabalho busca, era preciso analisar com mais detalhes os resultados.

Dessa forma, para uma melhor apuração dos dados, foi utilizada a ferramenta Power BI, desenvolvida pela *Microsoft* para análise de dados. Com ela foi de maior facilidade realizar a análise das respostas, permitindo coadunar as perguntas e gerar gráficos das associações, proporcionando uma melhor compreensão dos dados.

3 ANÁLISE DOS RESULTADOS

Nesta seção serão apresentados os resultados obtidos por meio da análise das respostas dos 77 participantes do questionário. Durante o processo de análise das respostas captadas, foi identificado que uma pessoa respondeu o questionário de forma aleatória, essa resposta foi desconsiderada para a análise dos dados a seguir. Portanto, todos os itens analisados, não contêm a resposta do participante em questão. Dessa forma, os resultados apresentados são referentes aos 76 participantes restantes.

3.1 ANÁLISE DAS RESPOSTAS COLETADAS

Os resultados exibidos a seguir serão organizados de acordo com os 5 grupos do questionário, cada um em um tópico, sendo: Perfil e experiência do participante, Aplicações web escaláveis em linguagem PHP, Desenvolvimento com *micro-frameworks*, Desenvolvimento com *full-stack frameworks* e Redesenvolvimento da aplicação buscando escalabilidade.

3.1.1 Perfil e Experiência Do Participante

Entre os 76 participantes 93,42% declararam viver no Brasil, 3,95% vivem na Alemanha, 1,32% nos EUA e 1,32% no México. A maioria dos participantes formam um público adulto, sendo 50% com idade entre 26 e 35 anos, 38,16% com idade entre 16 e 25 anos e 11,84% com idade entre 36 e 45 anos.

Sobre a formação acadêmica dos entrevistados, 59,21%, precisamente 45 pessoas, afirmaram ter cursado ou estarem cursando o ensino superior completo, 27,63% exatamente 21 pessoas terem pós-graduação, 7 pessoas cerca de 9,21%, afirmam terem concluído apenas o ensino médio, 2,63% exatamente 2 pessoas, cursaram o ensino técnico e apenas um participante

concluiu o mestrado com 1,32%.

Com relação ao tempo de atuação dos entrevistados na área de TI, notou-se um bom nível de experiência dos entrevistados, a maioria, exatamente 23 pessoas correspondendo a 30,26%, tem cerca de 7 a 10 anos de atuação, entrevistados com mais de 10 anos e com 4 a 6 anos de atuação em TI, se igualaram com 26,32% totalizando 20 participantes para cada grupo, participantes com 1 a 3 anos de experiência totalizaram 10 e correspondem a 13,16%, com experiência de 0 a 12 meses são 3 pessoas e correspondem a 3,95%.

Em relação a ocupação atual dos participantes, 51,32%, exatamente 39 pessoas, afirmaram ser Desenvolvedor/Programador, 14 pessoas relativo a 18,42% afirmaram desempenhar funções de Analista de Sistemas, 14,45% (11 participantes), se consideram Arquitetos de *Software*, 3,95% (3 pessoas) são Estagiários, 2,63% exatamente 2 pessoas são Gerente de Projetos, 9,23% exatamente 7 pessoas declararam desempenhar outras funções dentro da área de TI (Tecnologia da Informação).

Grande maioria dos entrevistados, cerca de 42,11% trabalham em empresas com 2 a 20 funcionários, 26,32% em empresas com mais de 100 funcionários, 14,47% em empresas com 21 a 40 funcionários, 9,21% são autônomos, 5,26% em empresas com 41 a 60 funcionários, 1 participante (1,32%) em uma empresa com 81 a 100 funcionários e 1 entrevistado declarou trabalhar em uma empresa com 61 a 80 funcionários.

Os participantes também foram indagados sobre qual o nível de conhecimento possuem sobre aplicações escaláveis. As opções de respostas foram dadas em escala de 0 a 10, sendo que quanto mais próximo de 10, maior conhecimento o participante apresenta sobre o assunto em questão e quanto mais próximo de 0 menor conhecimento.

O Gráfico 1 mostra o nível de conhecimento que os participantes da pesquisa apresentam sobre aplicações escaláveis.

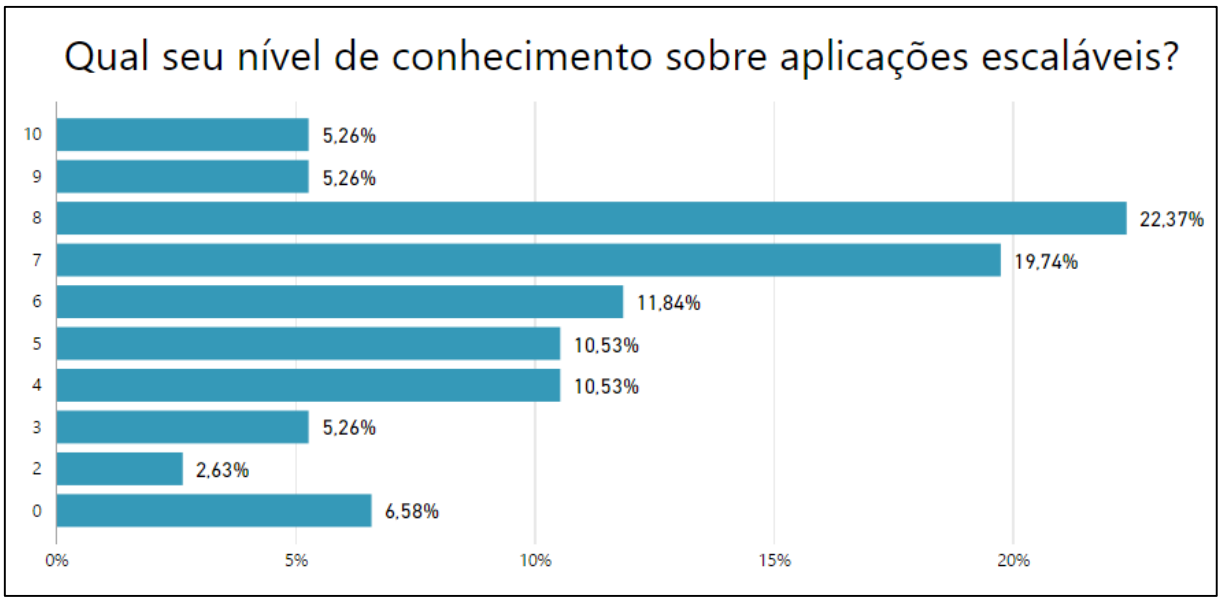

Gráfico 1 - Qual seu nível de conhecimento sobre aplicações escaláveis?
Fonte: Próprio autor

A maioria dos entrevistados, cerca de 22,37% declararam ter um conhecimento de nível 8 sobre o assunto, em seguida 19,74% com nível 7, 11,84% apresentam conhecimento de nível 6, nível 4 e nível 5 com 10,53%, 6,58% não apresentam nenhum conhecimento sobre aplicações escaláveis, nível 3, 9 e 10 ficaram com 5,26% dos participantes cada, e 2,63% apresentam conhecimento de nível 2.

Associando o nível de conhecimento sobre aplicações escaláveis com o cargo dos participantes, notou-se que a maioria dos que apresentam conhecimento mais profundo, acima de nível 7, são arquitetos de *software*, como mostra o Gráfico 2.

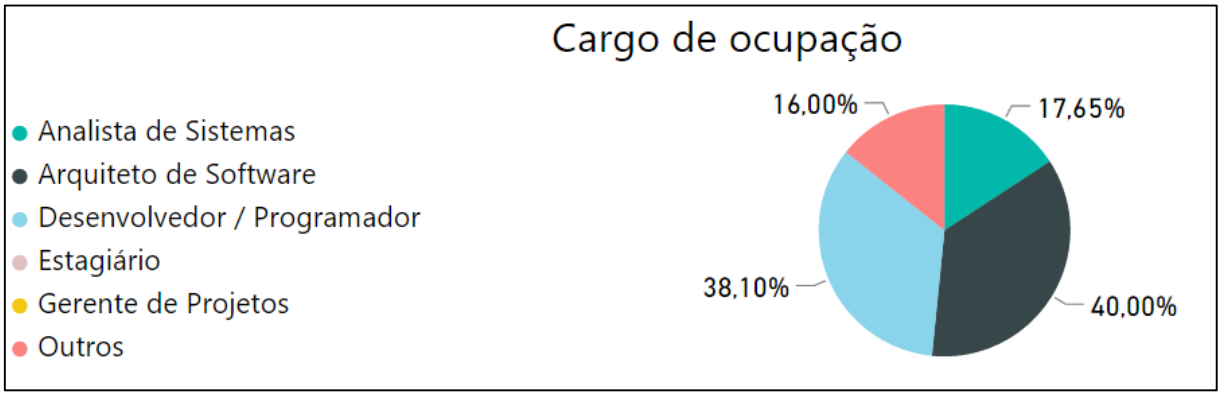

Gráfico 2 - Cargo dos Participantes com nível de conhecimento acima de 7
Fonte: Próprio autor

O percentual de participantes que se declaram arquitetos de *software*, aumenta em

relação aos outros cargos na medida em que o nível de conhecimento chega próximo a 10. O tempo de atuação na área de TI dos profissionais que apresentam tal nível de conhecimento também aumenta na mesma proporção.

O grupo avaliava por último, se o entrevistado já participou do desenvolvimento de alguma aplicação escalável desenvolvida em linguagem PHP. O Gráfico 3 exibe a participaram dos participantes no desenvolvimento de aplicações escaláveis em linguagem PHP.

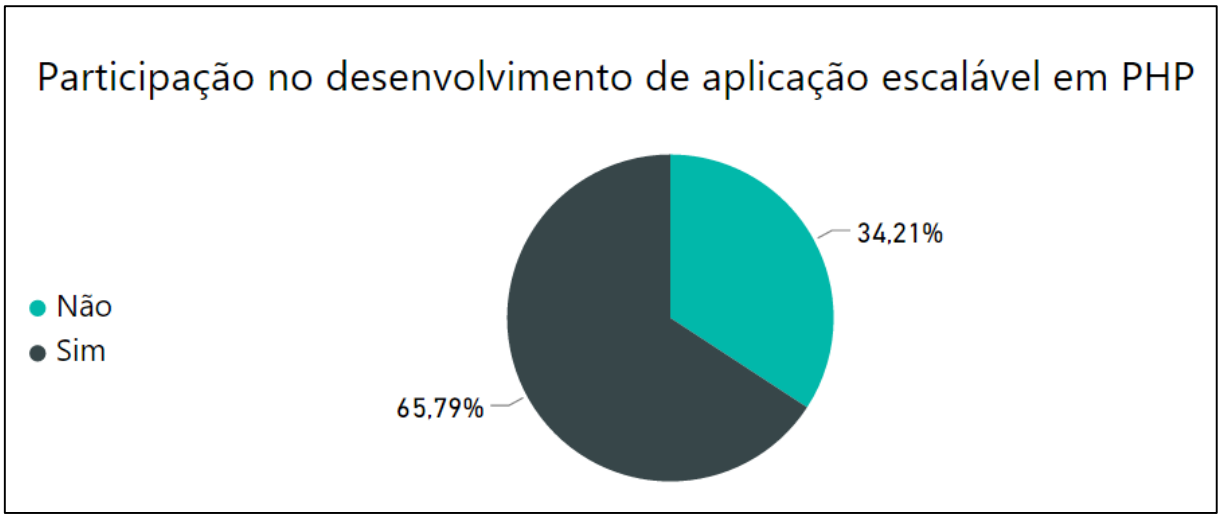

Gráfico 3 -Você já trabalhou em algum projeto ou no desenvolvimento de alguma aplicação escalável em linguagem PHP?
Fonte: Próprio autor

Aproximadamente 65,79% das pessoas participaram no desenvolvimento de alguma aplicação escalável em linguagem PHP e 34,21% nunca participaram.

3.1.2 Aplicações Web Escaláveis em Linguagem PHP

Essa seção contém perguntas feitas a pessoas que declararam participar do desenvolvimento de alguma aplicação escalável em linguagem PHP, seu objetivo é saber mais sobre a experiência do participante com o desenvolvimento de aplicação escalável em linguagem PHP.

Uma das perguntas era sobre quantos projetos escaláveis o participante teve contato. O Gráfico 4 mostra a participação dos entrevistados em projetos escaláveis.

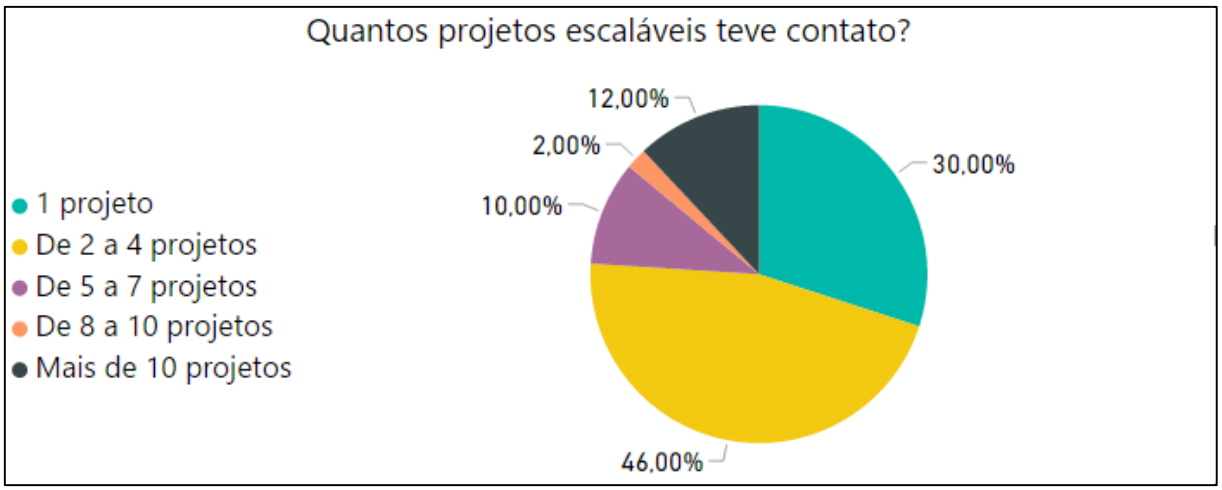

Gráfico 4 - Quantos projetos escaláveis teve contato?
Fonte: Próprio autor

Aproximadamente 46% dos entrevistados participaram de 2 a 4 projetos de aplicações escaláveis, logo em seguida 30% declaram ter participado de apenas um projeto, 12% participaram de mais de 10 projetos, 10% de 5 a 7 projetos e 2% disseram ter participado de 8 a 10 projetos escaláveis.

Associando a pergunta atual com o nível de conhecimento o participante declarou ter e sua ocupação, podemos notar que quanto mais projetos o entrevistado teve contato, mais profundo é seu conhecimento sobre aplicações escaláveis e a maioria dos participantes relata ocupar o cargo de arquiteto de *software*.

O Gráfico 5 mostra um exemplo dessa associação filtrando participantes que tiveram contato com mais de 10 projetos escaláveis.

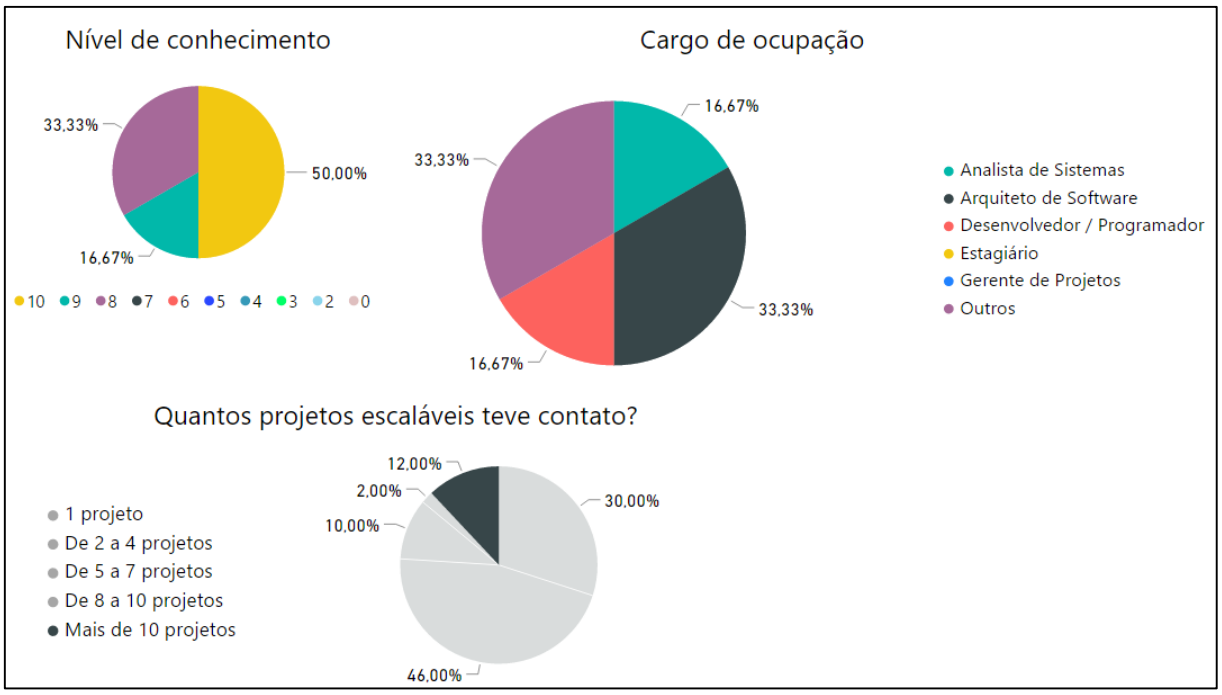

Gráfico 5 - Associando quantidade de projetos com nível de conhecimento e função desempenhada
Fonte: Próprio autor

Foi perguntado também ao participante, sobre quantas pessoas em média estavam envolvidas nos projetos que colaborou. O Gráfico 6 mostra quantas pessoas em média estavam envolvidas nos projetos que o entrevistado esteve envolvido.

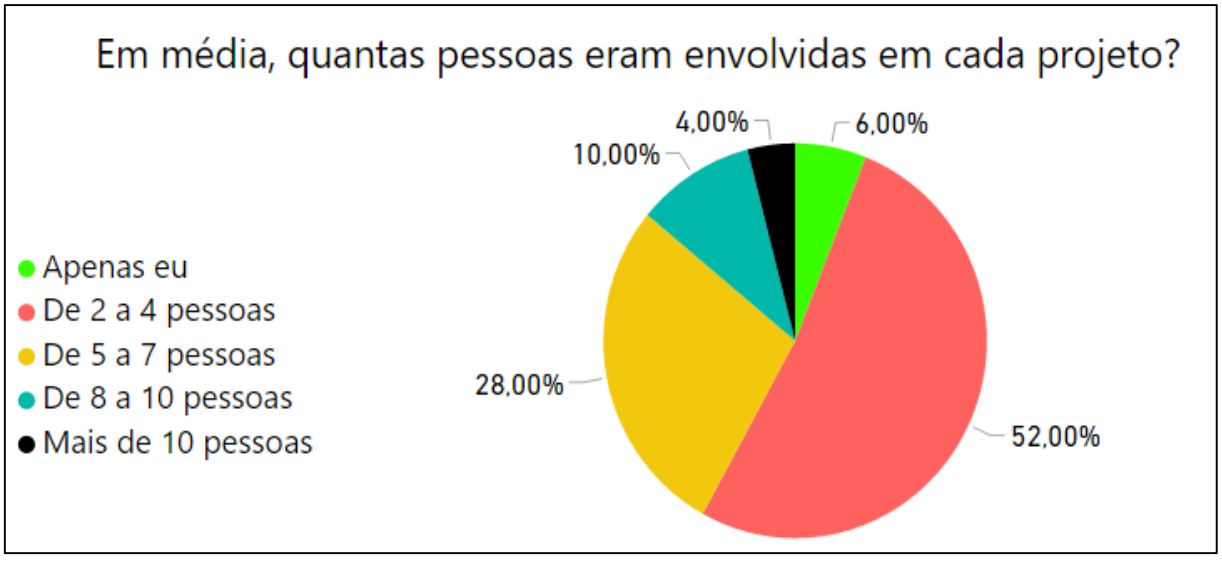

Gráfico 6 - Em média, quantas pessoas eram envolvidas em cada projeto?
Fonte: Próprio autor

Segundo os participantes da pesquisa, aproximadamente 52% destes, estavam inseridos em projetos escaláveis com participação de 2 a 4 pessoas, 28% continham de 5 a 7 pessoas, 10% relataram ter de 8 a 10 pessoas, 6% relataram estarem inseridos sozinhos no projeto e 4%

relataram estarem inseridos em projetos com mais de 10 pessoas.

3.1.3 Desenvolvimento com Micro-Frameworks

Nesta seção de perguntas, o objetivo era saber sobre a experiência acerca da utilização de *micro-frameworks* pelos participantes, principalmente daqueles que declararam trabalhar no desenvolvimento de aplicações escaláveis.

O grupo inicia perguntando se das aplicações que o entrevistado colaborou no desenvolvimento, alguma fez uso de *micro-framework*. O Gráfico 7 demostra o percentual de participantes que utilizaram e não utilizaram *micro-frameworks* no desenvolvimento de alguma aplicação que tenha colaborado.

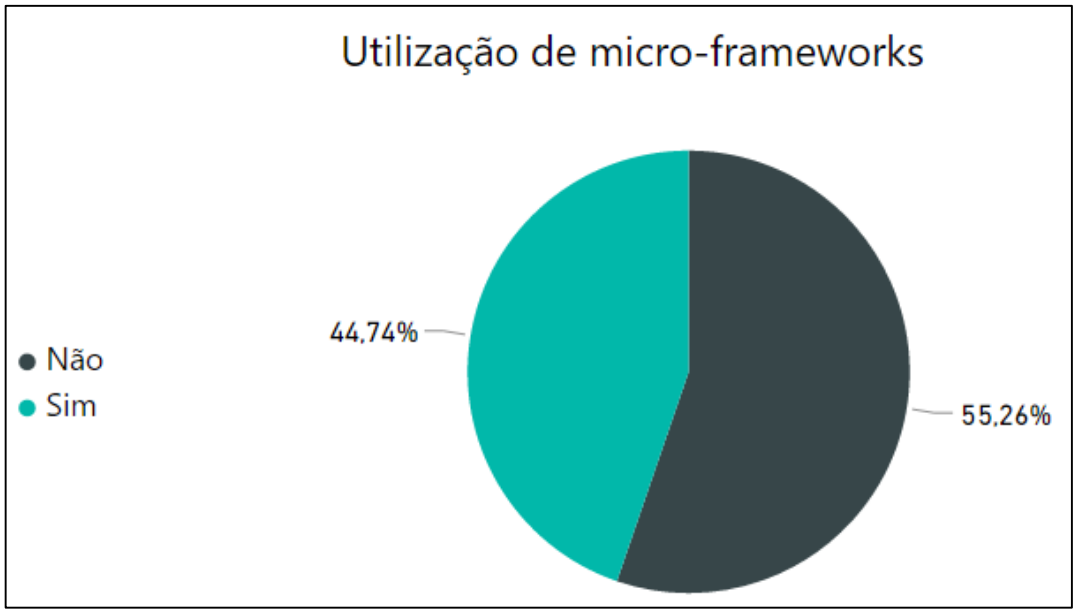

Gráfico 7 - Das aplicações que você participou do desenvolvimento, foi utilizado algum micro-framework?
Fonte: Próprio autor

Um total de 34 participantes, aproximadamente 44,74%, declararam já terem desenvolvido alguma aplicação que fizesse uso de *micro-framework* e 55,26%, exatamente 42 pessoas declararam não terem desenvolvido utilizando *micro-framework*.

Se caso o participante tenha declarado que já utilizou algum *micro-framework* no desenvolvimento de alguma aplicação, o restante das perguntas pertinentes a esse assunto era exibido logo em seguida com intuito de saber mais sobre a experiência do mesmo com o uso

desta terminologia de *framework*, por tanto, para análise das próximas perguntas, foram consideradas 34 respostas.

Na questão seguinte, fora perguntado ao participante sobre qual o principal *micro-framework* utilizado nas aplicações que o mesmo colaborou no desenvolvimento. As opções de *micro-frameworks* pré-definidas para escolha na pergunta, eram todas opões de *frameworks* construídos em linguagem PHP.

O Gráfico 8 mostra os *micro-frameworks* que os participantes declaram utilizar.

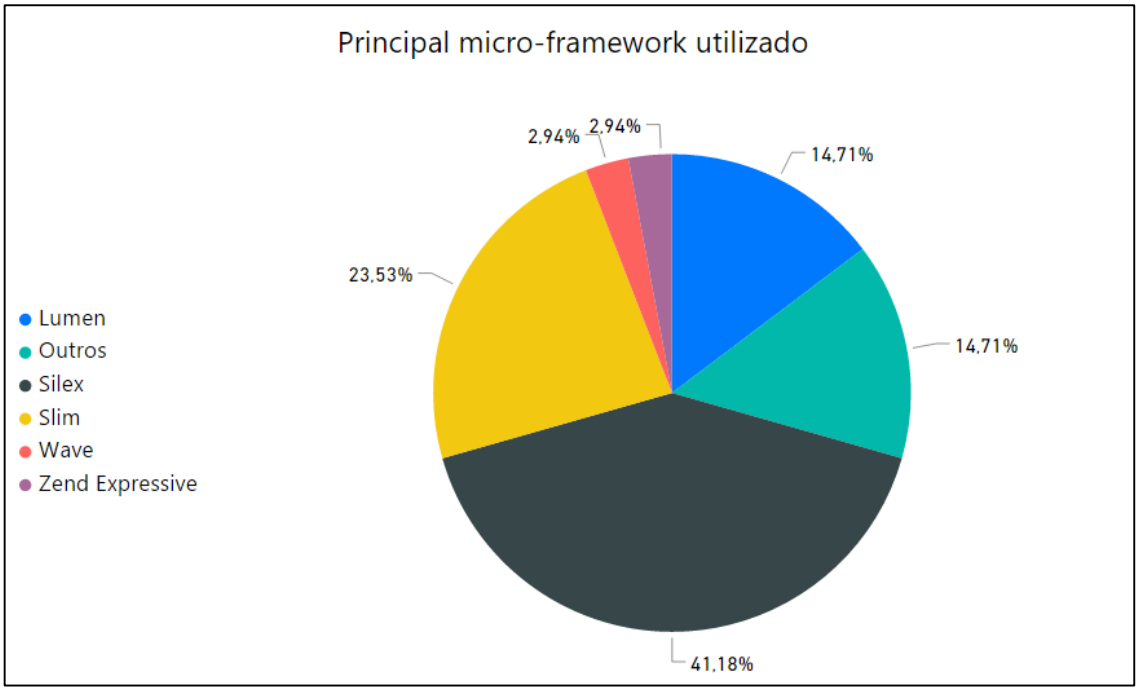

Gráfico 8 - Qual o principal micro-framework utilizado?
Fonte: Próprio autor

Observa-se que Silex é o *micro-framework* mais utilizado, já que a maioria dos participantes, correspondendo a 41,18%, declararam tê-lo como o mais utilizado entre as aplicações que o mesmo colaborou. Logo em seguida, aparece Slim, com 23,53% dos participantes tendo-o como principal *micro-framework* utilizado; e em terceiro aparece Lumen, com 14,71%. Esses *frameworks* foram os mais utilizados dentre todos os participantes que declararam terem desenvolvido alguma aplicação que fizesse uso de *micro-framework*, todos em linguagem PHP. Um total aproximado de 14,71% dos participantes, declararam terem outro *micro-framework* como o principal utilizado

As próximas questões tiveram como objetivo, saber sobre a qualidade das aplicações desenvolvidas utilizando a terminologia *micro-framework*. Fora perguntado ao participante, como ele avalia a flexibilidade da arquitetura do *framework* utilizado, para se adequar as

necessidades da aplicação e permitir sua evolução.

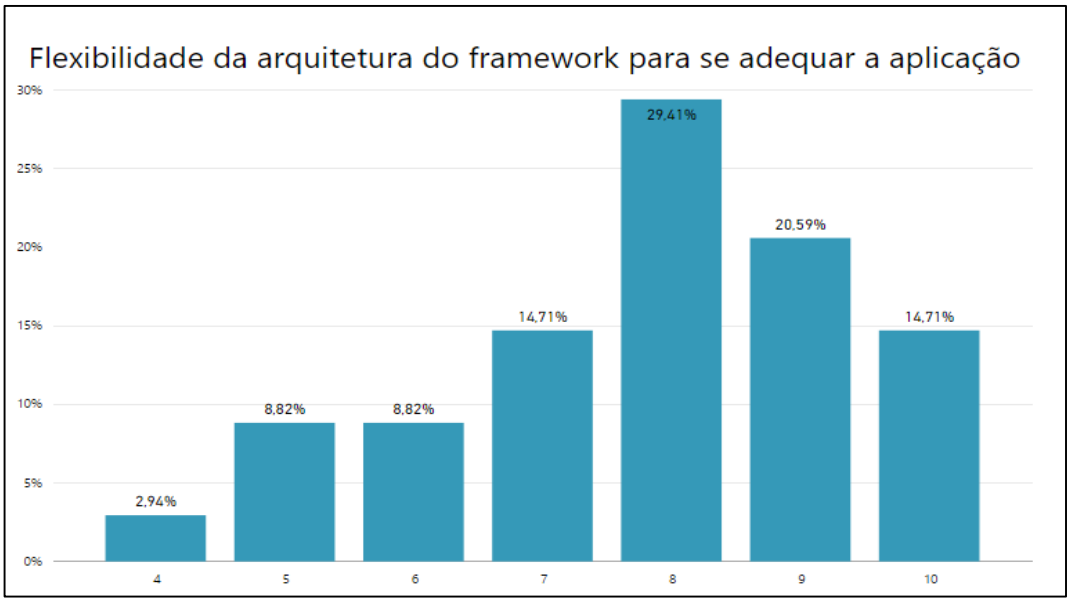

Gráfico 9 - Como você avalia o nível de flexibilidade da arquitetura do framework utilizado, para se adequar as necessidades da aplicação e permitir a evolução da mesma?
Fonte: Próprio autor

Nota-se que a maioria dos entrevistados em um âmbito geral, classificou a flexibilidade da arquitetura do *framework* para se adequar as necessidades da aplicação, com níveis altos, como 8, 9, 10 e 7 em sequência. Sendo que os *frameworks* desta terminologia que foram classificados com o nível mais alto (10), foram os mesmos apontados como os mais utilizados, Slim, Silex e Lumen. Nesta associação, Slim aparece com 60% dos participantes classificando-o com tal nível de aptidão nesta característica, Lumen e Silex aparecem empatados com 20% dos entrevistados.

Após essa questão, uma próxima pergunta era feita com o objetivo de saber se a característica, apontada pelos participantes na questão anterior, alterava-se durante o desenvolvimento da aplicação que fez uso do *framework*, afim de saber se o suporte a flexibilidade continuava nesta etapa.

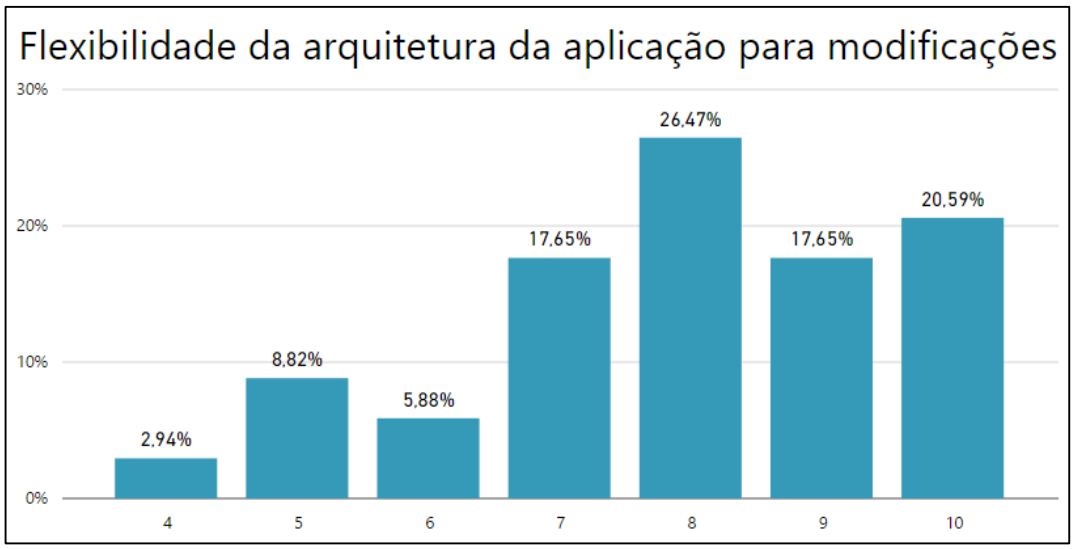

Gráfico 10 - Como você avalia o nível de flexibilidade da arquitetura dessa aplicação, ao sofrer modificações durante seu processo de desenvolvimento?
Fonte: Próprio autor

Realizou-se uma associação entre a questão atual e a anterior, levando em consideração os *frameworks* apontados como os mais utilizados, sendo Slim, Silex e Lumen.

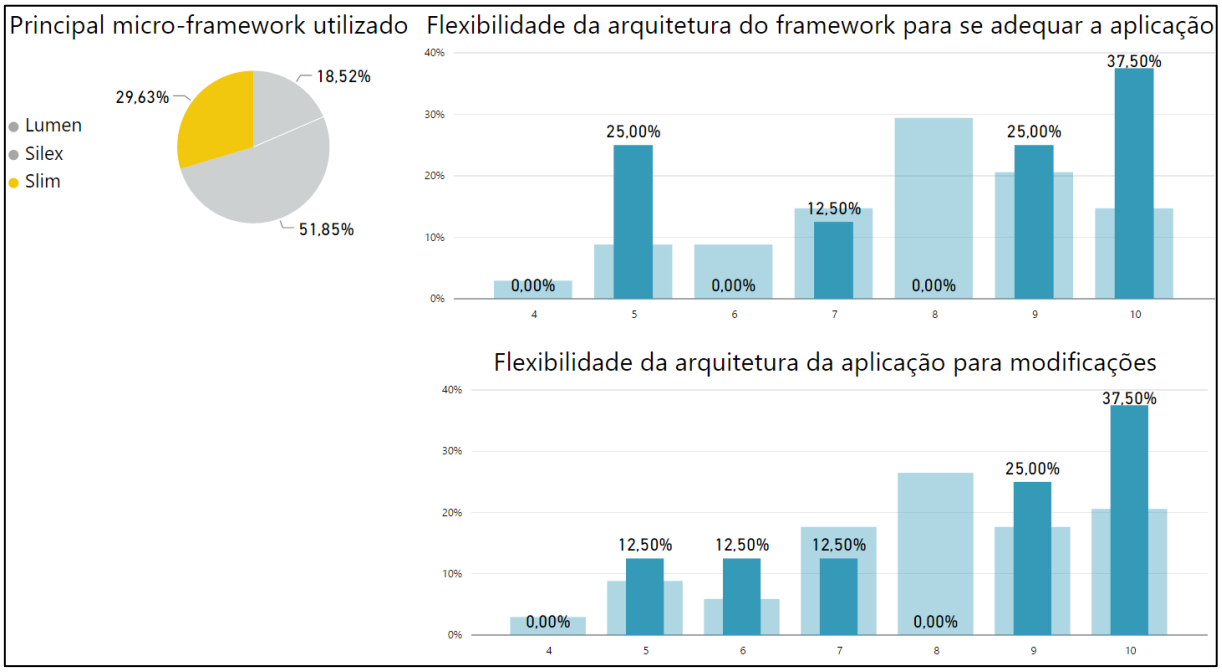

Gráfico 11 - Associação entre a questão 15 e 16 entre participantes que utilizaram *framework* Slim
Fonte: Próprio autor

Em relação ao *micro-framework* Slim, nota-se que mantem a mesma proporção de níveis apontados por participantes anteriormente, sendo a única diferença a porcentagem entre participantes que classificaram com nível 5 e 6, na pergunta anterior a porcentagem com nível 5, era de 25,00% e de nível 6 era de 0%, na pergunta atual, nível 5 aparece com apenas 12,50%

e nível 6 com 12,50%.

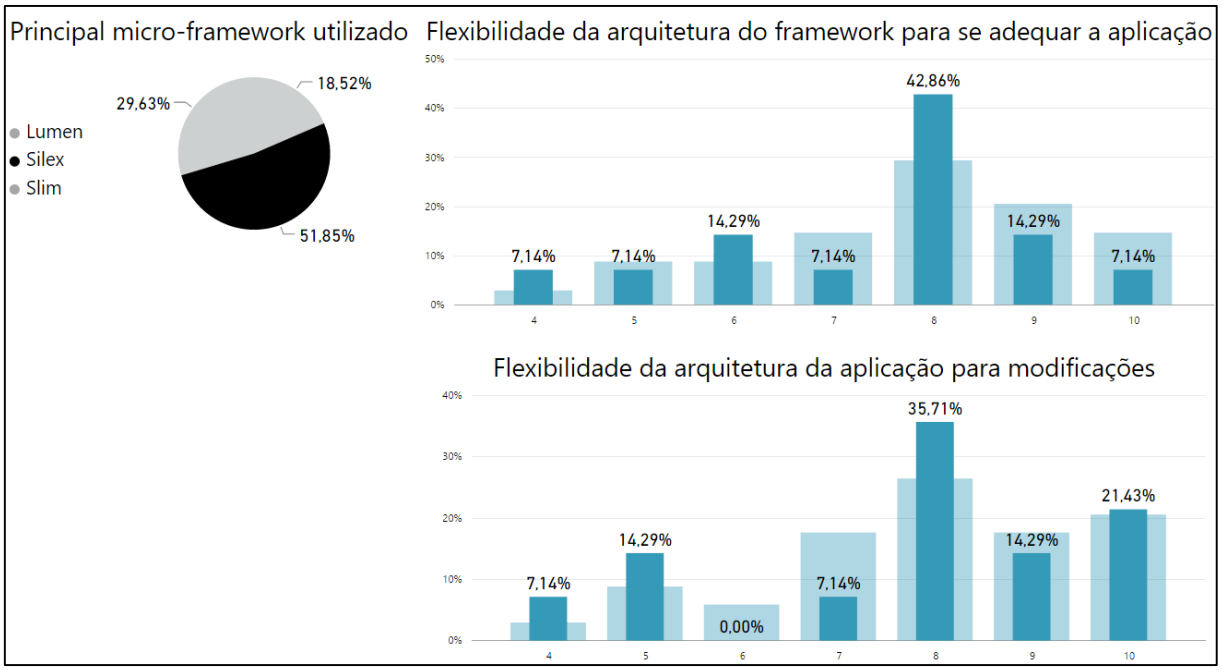

Gráfico 12 - Associação entre a questão 15 e 16 entre participantes que utilizaram framework Silex
Fonte: Próprio autor

Em relação ao framework Silex**,** houve diferença entre os níveis de classificação, porém, a mudança mais perceptível foi a porcentagem de participantes que classificaram com nível 10 na questão anterior, sendo 7,14% e com nível 10 na pergunta atual correspondendo a 21,43% dos participantes.

Ao realizar a análise da associação entre entrevistados que declararam utilizar o *framework* Lumen, notou-se uma contrapartida em relação aos outros dois *frameworks* analisados anteriormente, pois a porcentagem de classificação em níveis mais altos diminuiu e a de níveis mais baixos aumentou, se antes a porcentagem no nível 8 era a maioria correspondendo a 40,00%, na pergunta atual houve uma mudança significativa passando para 0% e nível 7 passando de 20,00% para 60,00%.

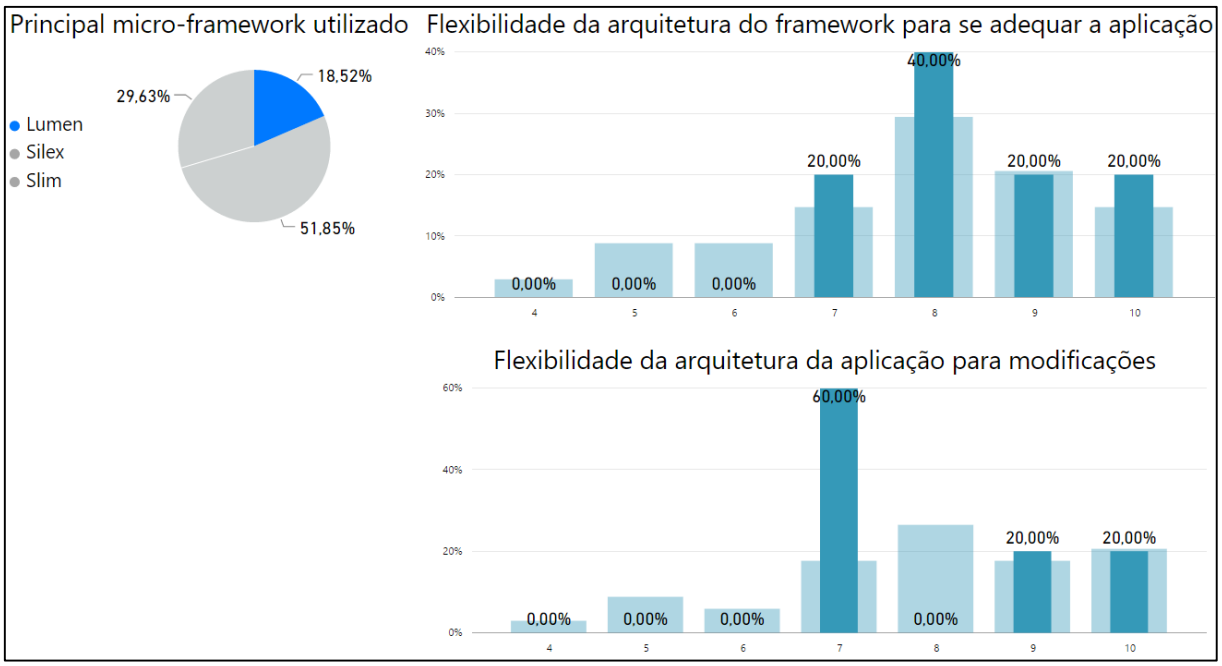

Gráfico 13- Associação entre a questão 15 e 16 entre participantes que utilizaram framework Lumen
Fonte: Próprio autor

Fora perguntado ao participante sobre a dificuldade ao adicionar novas funcionalidades à aplicação após a conclusão do seu desenvolvimento, sem interferir em outras funcionalidades já desenvolvidas.

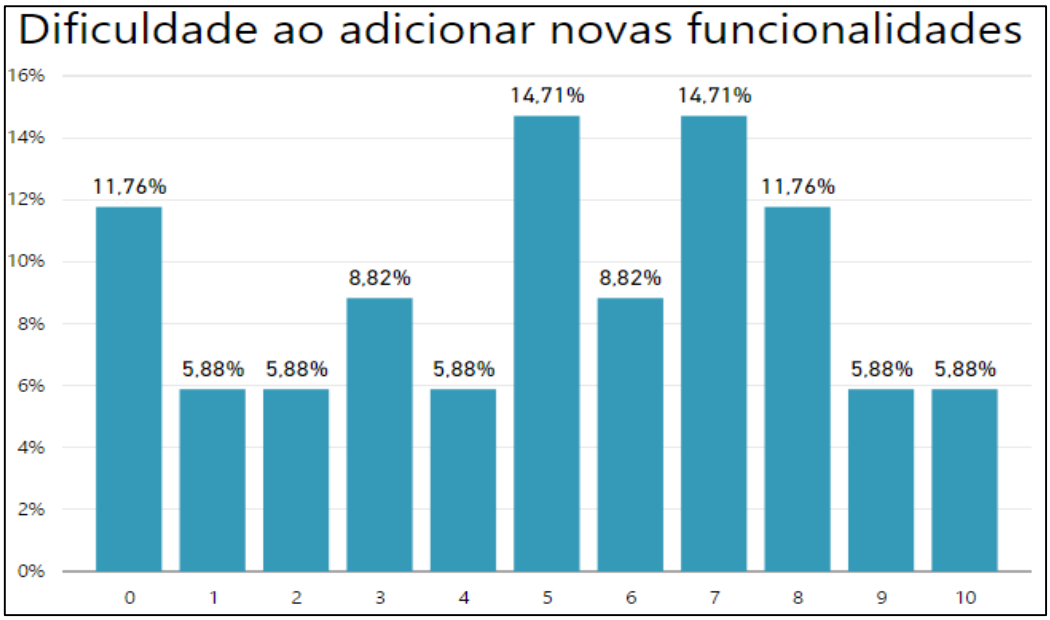

Gráfico 14 - Qual sua classificação em termos de dificuldade, ao adicionar novas funcionalidades à aplicação após sua conclusão, sem interferir em funcionalidades já desenvolvidas?
Fonte: Próprio autor

Observa-se que os níveis de dificuldades declarados pelos participantes sobre esse quesito, apresenta uma classificação distribuída entre diferentes níveis, por isso, afim de saber

um pouco mais sobre o desenvolvimento da aplicação e ter uma melhor conclusão, essa questão foi relacionada com uma das próximas, que questionava ao entrevistado se antes de dar início ao desenvolvimento da aplicação, foi realizado algum projeto arquitetural para a mesma.

Realizando a associação, percebe-se que o percentual de participantes que declaram níveis de dificuldade mais alto, grande maioria também declarou que não houve um projeto arquitetural antes de iniciar o desenvolvimento da aplicação. Isso reforça a ideia do quão importante é planejar uma arquitetura para um *software*, pois "a facilidade de fazer alteração no sistema existente, seja adicionando ou modificando alguma funcionalidade, depende muito da arquitetura deste"(MENDES, 2002, p 47). Cerca de 58,82% dos participantes declararam terem feito um projeto arquitetural para a aplicação antes de iniciar seu desenvolvimento e 41,18% não terem feito.

Também fora perguntado, sobre como o entrevistado classifica o nível de desempenho da aplicação depois de desenvolvida. O Gráfico 15 mostra como os participantes classificaram o desempenho da aplicação após seu desenvolvimento.

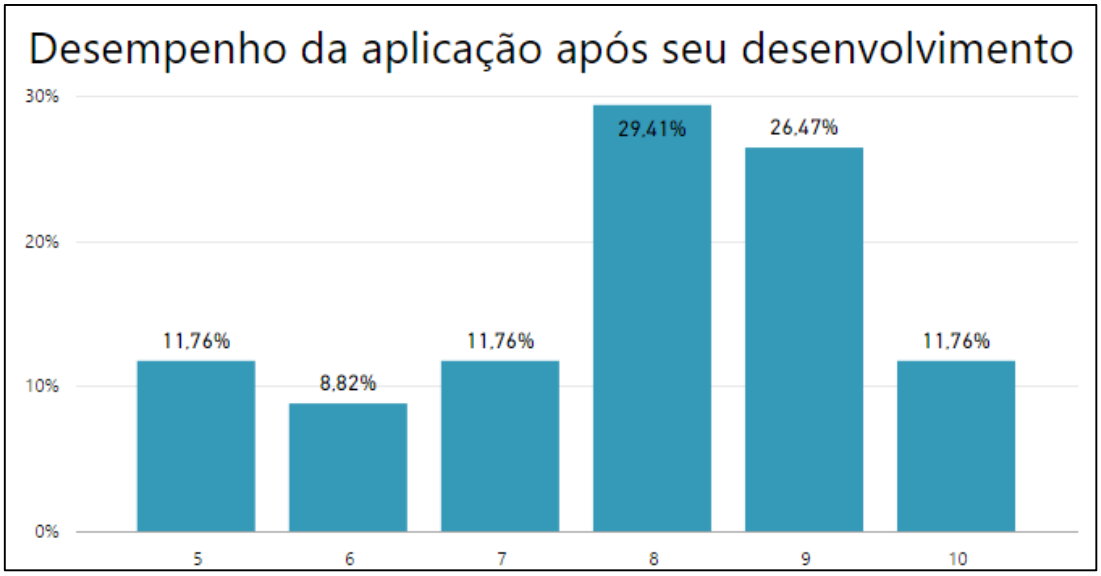

Gráfico 15 - Como você avalia o nível de desempenho dessa aplicação depois de desenvolvida?
Fonte: Próprio autor

Boa parte dos participantes atribuíram um bom nível de desempenho para aplicação após a fase de desenvolvimento da mesma ter sido concluída, sendo o nível 8 o mais apontado, com 29,41% dos entrevistados.

Foi realizado novamente uma associação com a pergunta em que questionava ao participante se houve um projeto arquitetural para a aplicação, assim como também, com a pergunta sobre a flexibilidade da aplicação durante seu desenvolvimento e sobre o nível de

dificuldade para se adicionar novas funcionalidades sem interferir em outras já concluídas; isto levando em consideração o nível mais alto de desempenho indicado pelos participantes, 10.

Realizada a associação, notou-se que 100% dos participantes que declararam o nível de desempenho mais alto, realizaram um projeto arquitetural para a aplicação antes de iniciar seu desenvolvimento; os mesmos também declararam um alto nível de flexibilidade da arquitetura da aplicação para modificações durante sua construção e grande maioria, 75%, ainda classificou como nível 0 em termos de dificuldade para se adicionar novas funcionalidades a aplicação sem interferir em funcionalidades concluídas. O Gráfico 16 exibe essa associação.

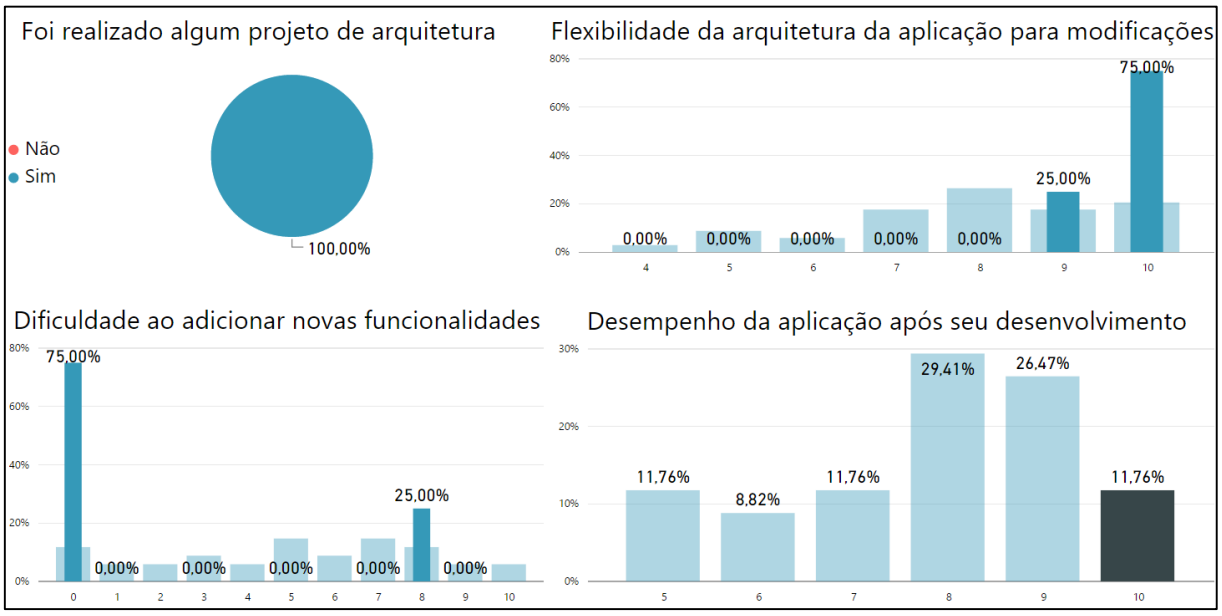

Gráfico 16 - Associação entre a questão 19, 16, 17 e 18
Fonte: Próprio autor

Nesta seção, o entrevistado ainda foi indagado sobre o motivo que influenciou na escolha por um *micro-framework* para o desenvolvimento da aplicação. O Gráfico 17 apresenta os principais motivos declarados pelos entrevistados pela escolha de um *micro-framework* para o desenvolvimento da aplicação.

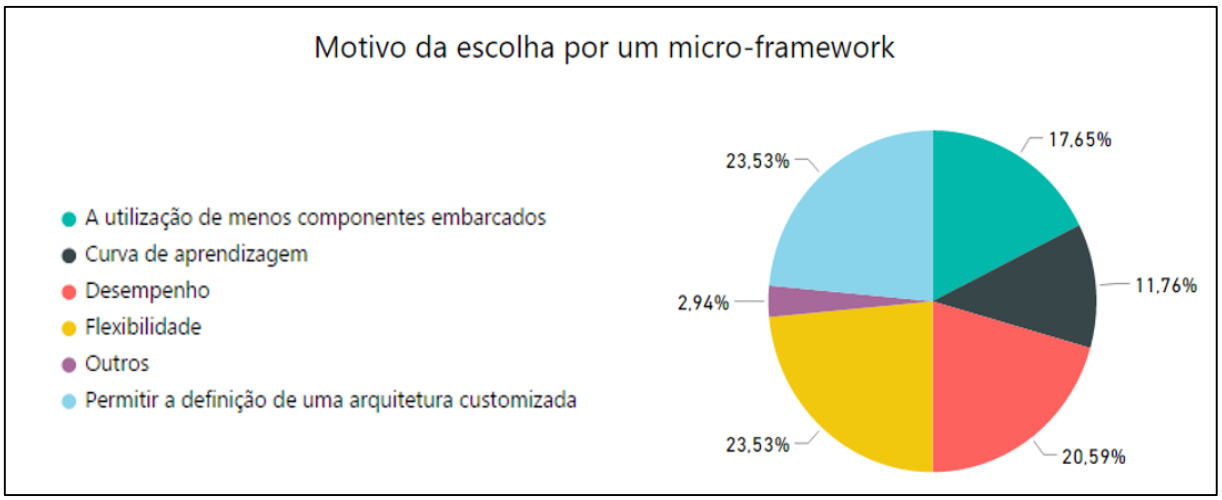

Gráfico 17 - Qual o principal motivo influenciou na escolha de um micro-framework para o desenvolvimento?
Fonte: Próprio autor

Como destaque, temos empatado, dois motivos principais que levaram aos participantes a optar por um *micro-framework* para o desenvolvimento da aplicação, sendo, a flexibilidade de modo geral presente no *framework* e o fato de permitir a definição de uma arquitetura customizada, ambos com 23,53%. Em seguida tem-se desempenho como segundo fator mais apontado, com cerca de 20,59% dos participantes.

Essa seção também procurou saber em qual cenário de arquitetura houve o emprego da utilização do *framework*, entre as opções pré-definidas estavam: Arquitetura *RestFul* e Arquitetura MVC. O Gráfico 18 apresenta os cenários utilizados apontados pelos entrevistados.

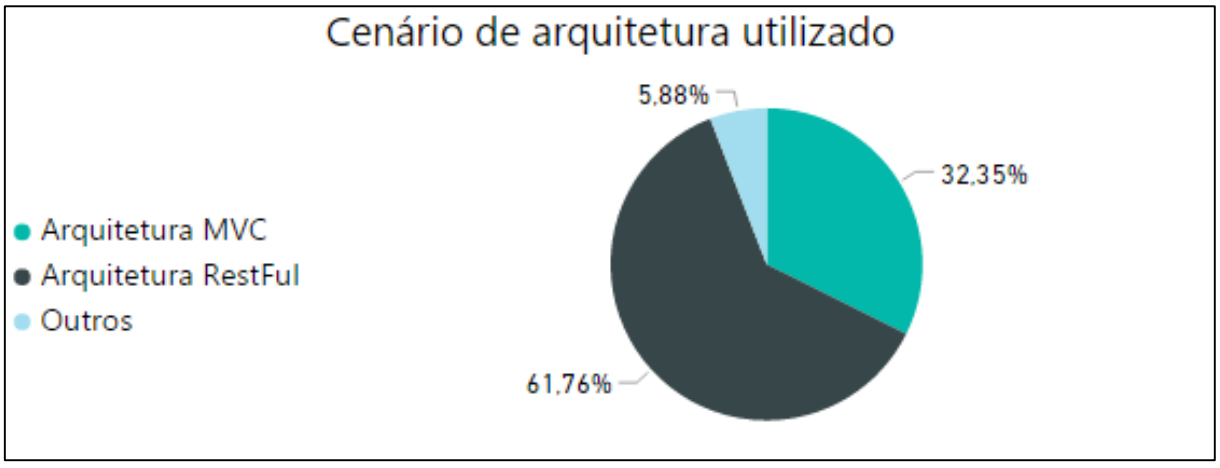

Gráfico 18 - Em qual cenário de arquitetura foi utilizado o framework?
Fonte: Próprio autor

Percebe-se que a maior parte dos participantes, aproximadamente 61,76% declararam utilizar uma "Arquitetura *RestFul*", em seguida 32,35% uma "Arquitetura MVC" e um total correspondente a 5,88% declarou utilizar em outro cenário de arquitetura.

Em seguida perguntou-se ao entrevistado quais metodologias de desenvolvimento foram utilizadas durante a construção da aplicação. Foi permitido ao participante que marcasse mais de uma opção nesta interrogação, pois como Gomes(2013) destaca, nenhum método é completo e nenhum é perfeito, portanto é comum que se encontre equipes utilizando junções de acordo com a cultura da organização, do contexto do projeto ou objetivo em questão.

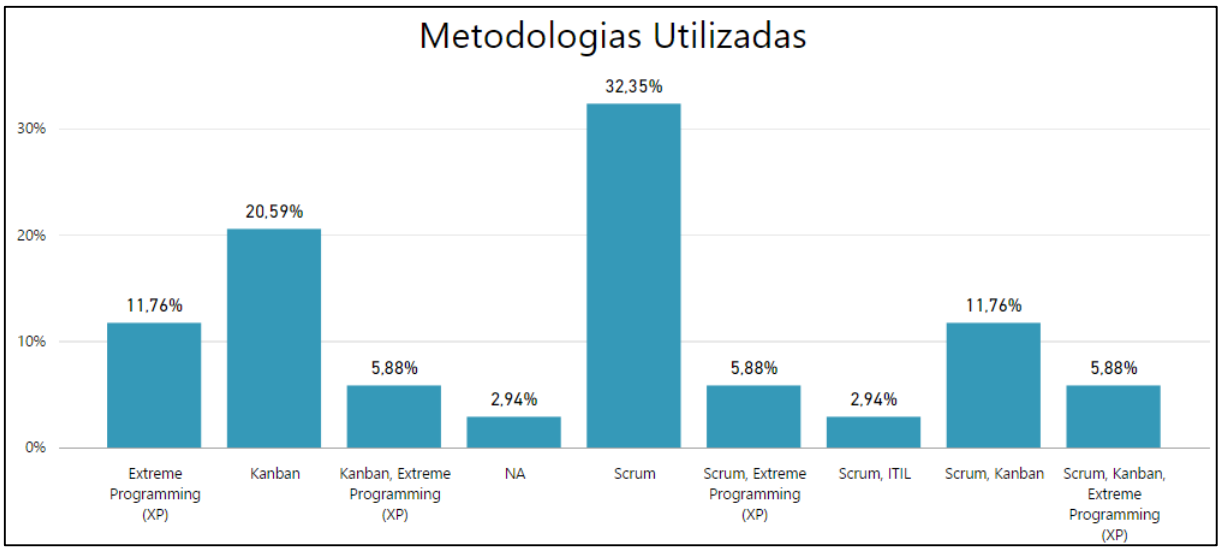

Gráfico 19 - Quais metodologias de desenvolvimento foram utilizadas durante a construção da aplicação?
Fonte: Próprio autor.

Como pode-se notar, Scrum é o método de desenvolvimento mais utilizado de forma geral, porém, nota-se também que os participantes declararam utilizar o método Kanban aliado com outros métodos ágeis, isso se deve ao fato de que o método Kanban incentiva a adaptação baseada em contexto (GOMES, 2013).

3.1.4 Desenvolvimento com Full-Stack Frameworks

Essa seção contém as mesmas perguntas da seção anterior, porém o objetivo era saber informações acerca de entrevistados que utilizaram *full-stack frameworks* no desenvolvimento de aplicações, principalmente aqueles que declararam participar da construção de aplicações escaláveis em linguagem PHP.

A primeira pergunta interrogava ao entrevistado se o mesmo já desenvolveu alguma aplicação que fazia uso de um *full-stack framework*. O Gráfico 20 mostra o percentual de entrevistados que já desenvolveram e que nunca desenvolveram alguma aplicação que fizesse

uso de algum *full-stack framework*.

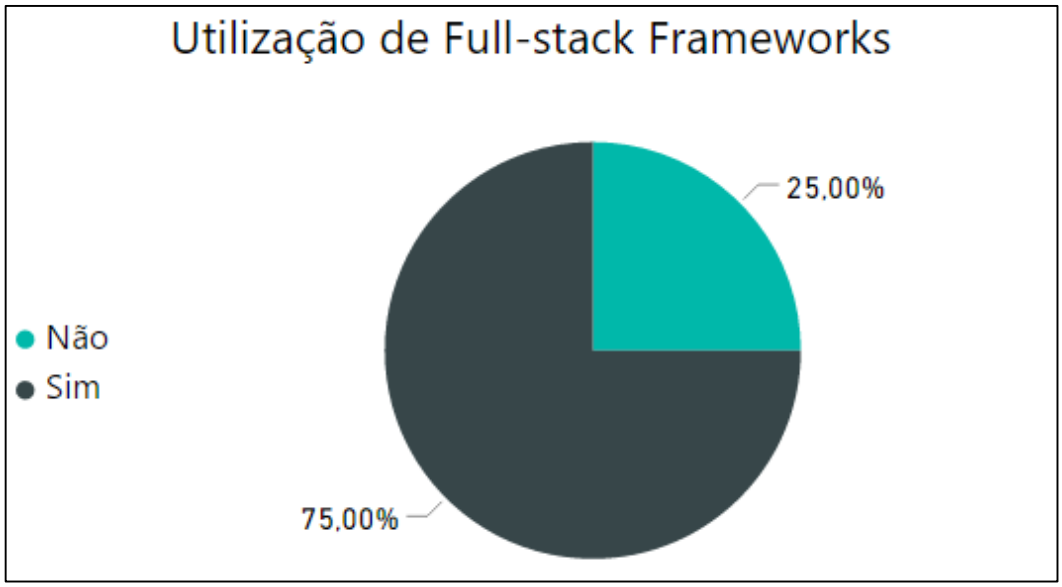

Gráfico 20 - Das aplicações que você participou do desenvolvimento, foi utilizado algum framework full-stack?
Fonte: Próprio autor

Cerca de 75% dos participantes, exatamente 57 pessoas, declararam terem participado do desenvolvimento de alguma aplicação que fizesse uso de *full-stack framework*, 19 entrevistados, aproximadamente 25% declarou nunca ter desenvolvido utilizando *framework* deste modelo.

Caso o entrevistado declarasse já ter participado da construção de uma aplicação que utilizasse *full-stack framework*, o restante das perguntas do grupo era exibido em seguida. Portanto a análise do restante das perguntas desse grupo, foram feitas levando em consideração todos os 57 entrevistados que confirmaram tal quesito.

Fora perguntado ao entrevistado, qual o principal *framework* desta terminologia foi utilizado nas aplicações que o mesmo colaborou no desenvolvimento. As opções pré-definidas eram *frameworks* escritos em linguagem PHP e foram selecionadas com base nos níveis de popularidade na comunidade PHP. O Gráfico 21 mostra os *full-stack frameworks* apontados como os mais utilizados no desenvolvimento de aplicações segundo os entrevistados.

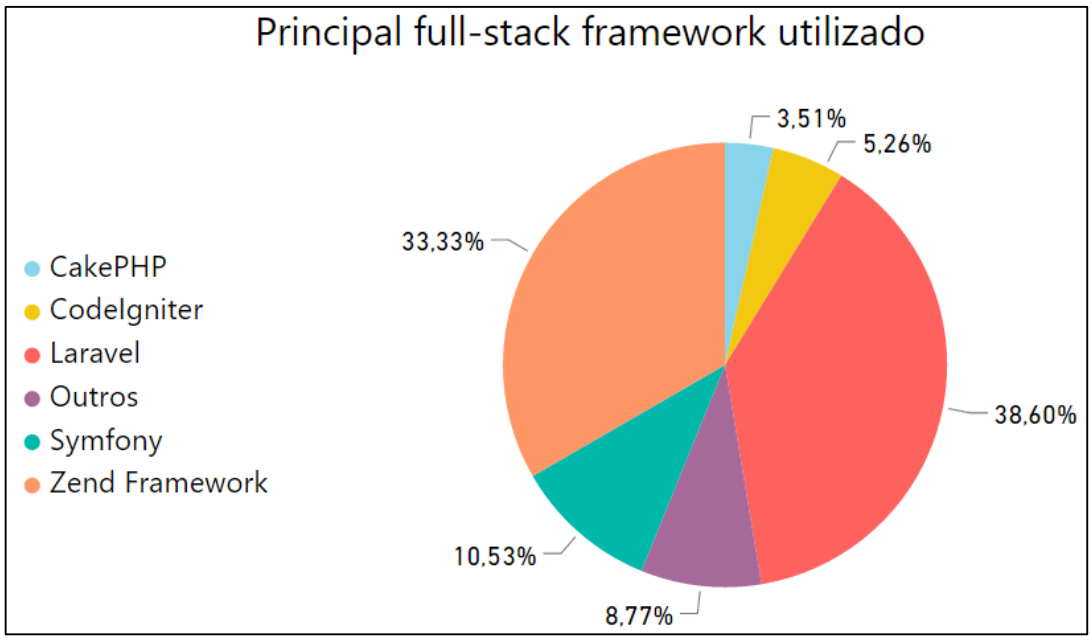

Gráfico 21 - Qual o principal framework full-stack utilizado?
Fonte: Próprio autor

Observa-se que Laravel é apontado como o mais utilizado dentre essa categoria, com cerca de 38,60% dos participantes, logo em seguida Zend Framework com cerca de 33,33% dos participantes e Symfony em terceiro com 10,53%, esses foram os mais utilizados segundo os entrevistados.

As próximas questões, tiveram como objetivo saber um pouco mais sobre a qualidade das aplicações desenvolvidas utilizando a terminologia *full-stack*.

Perguntou-se ao participante, como ele classifica a flexibilidade da arquitetura do *framework* utilizado para se adaptar às necessidades da aplicação no início do desenvolvimento. O Gráfico 22 exibe a classificação geral dos participantes em relação a esse quesito.

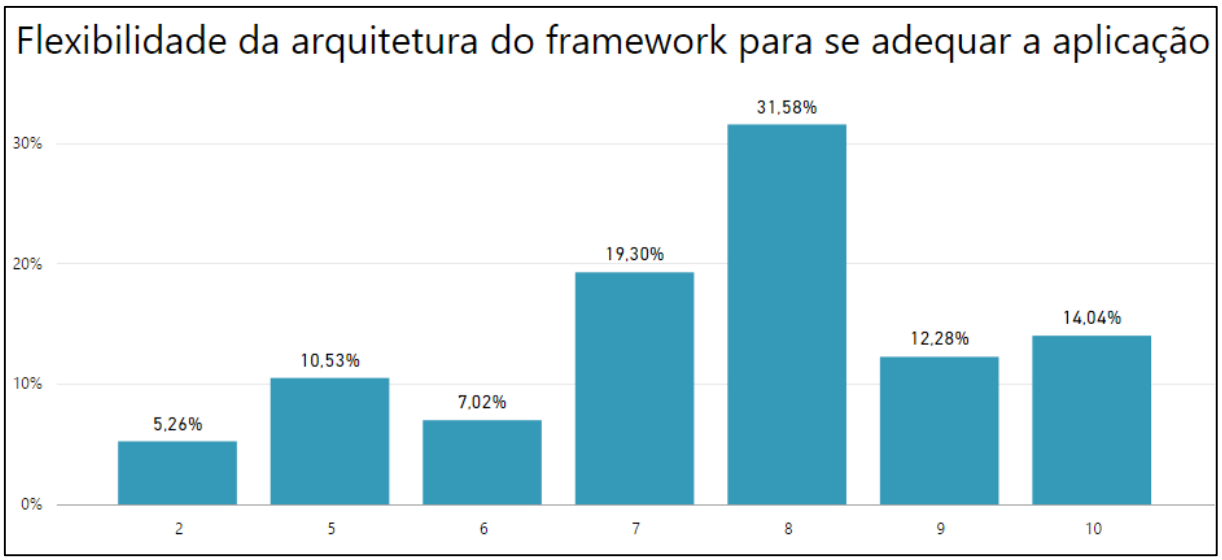

Gráfico 22 - Como você avalia o nível de flexibilidade da arquitetura do framework utilizado, para se adequar as necessidades da aplicação e permitir a evolução da mesma?
Fonte: Próprio autor

A classificação declarada pela maioria dos participantes ficou com nível 8, com um total de 31,58% dos entrevistados. Logo após 19,30% com nível 7 e em terceiro, nível 10 com 14,04%. Os *frameworks* que foram classificados com o nível mais alto, nível 10, foram, Laravel, sendo o que mais se destacou com 75% dos participantes, logo após Symfony e CodeIgniter empatados com 25% cada.

Logo em seguida o entrevistado era interrogado sobre a flexibilidade da arquitetura da aplicação em desenvolvimento, afim de saber se a característica apontada pelo mesmo na pergunta anterior, sofria alteração durante o ciclo de desenvolvimento da aplicação.

O Gráfico 23 mostra a classificação geral dos participantes, sobre o quesito de flexibilidade da arquitetura da aplicação em ser adepta a modificações durante seu desenvolvimento.

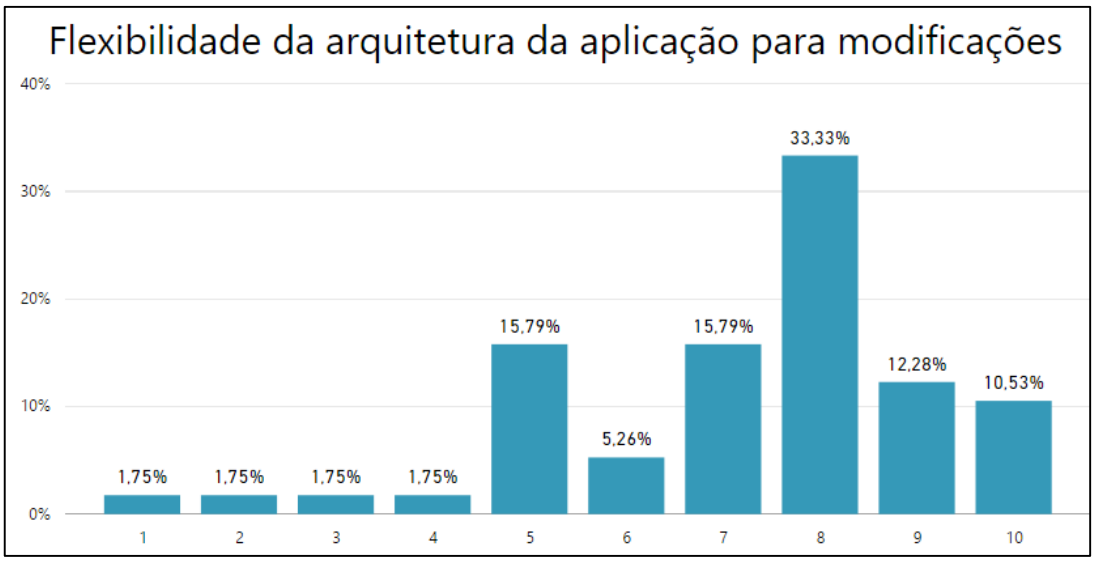

Gráfico 23 - Como você avalia o nível de flexibilidade da arquitetura dessa aplicação, ao sofrer modificações durante seu processo de desenvolvimento?
Fonte: Próprio autor

Percebe-se que grande maioria, aproximadamente 33,33%, classificou como nível 8 o suporte da arquitetura da aplicação para modificações durante a fase de desenvolvimento da mesma. Logo em seguida nível 7 e 5 empatados com 15,79%.

Foi realizada uma associação da questão atual com a pergunta anterior, levando em consideração os *frameworks* apontados como os mais utilizados, são eles: Laravel, Zend Framework e Symfony.

Em relação ao *framework* Laravel, entrevistados que classificaram o quesito de adaptação às necessidades da aplicação como nível 8, se mantiveram na mesma proporção com 31,82% quando questionados sobre o nível de aptidão da arquitetura da aplicação para sofrer modificações durante seu desenvolvimento, porém participantes que classificaram como nível 10 na pergunta anterior, diminuíram de 27,27% para 18,18% e entrevistados que classificaram com nível 9 aumentaram de 9,09% para 18,18%. Tal associação pode ser vista no Gráfico 24.

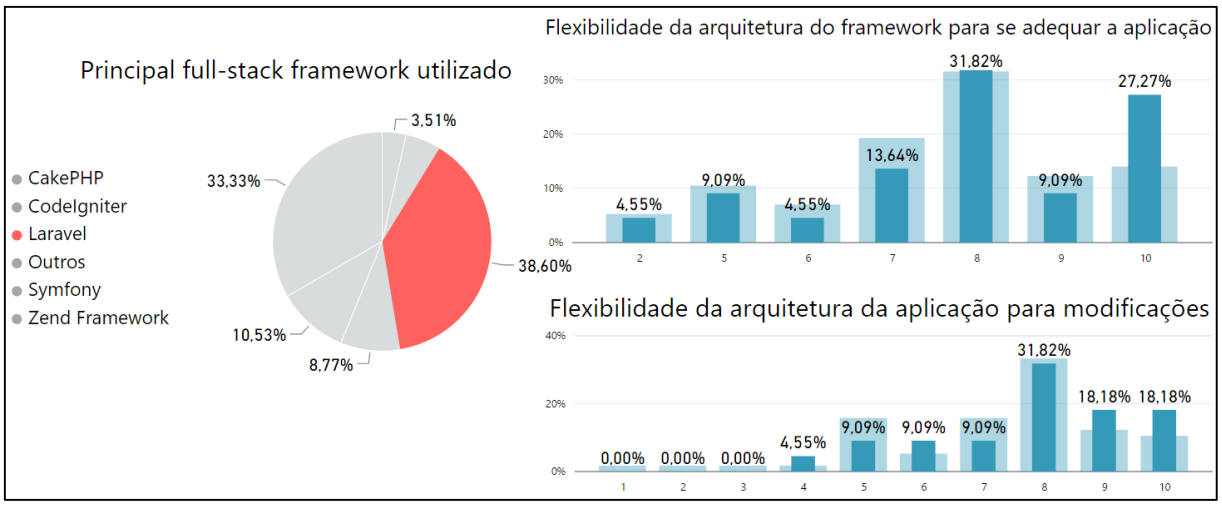

Gráfico 24- Associação entre a questão 25 e 26 entre participantes que utilizaram framework Laravel
Fonte: Próprio autor

Referente ao *framework* Zend Framework, houve uma queda em relação a participantes que o classificaram com nível 8 de 42,11% para 31,58% na pergunta atual. O percentual que atribuiu nível 6 também caiu em relação à pergunta atual, antes com 10,53% e agora com 5,26%; o percentual de entrevistados que atribuiu nível 5 na pergunta atual é de 21,05%, na pergunta anterior o percentual em relação ao mesmo nível era de apenas 10,53%. De forma geral, nota-se uma queda na classificação da flexibilidade por parte dos entrevistados que declararam utilizar o *framework* Zend Framework antes de iniciar o desenvolvimento e durante o desenvolvimento da aplicação.

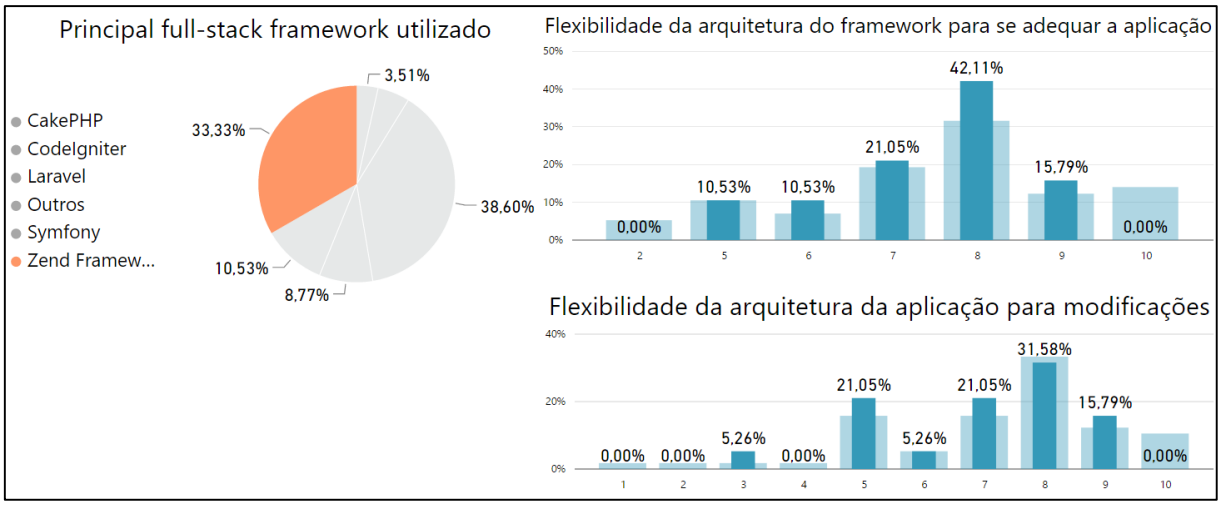

Gráfico 25 - Associação entre a questão 25 e 26 entre participantes que utilizaram Zend Framework
Fonte: Próprio autor

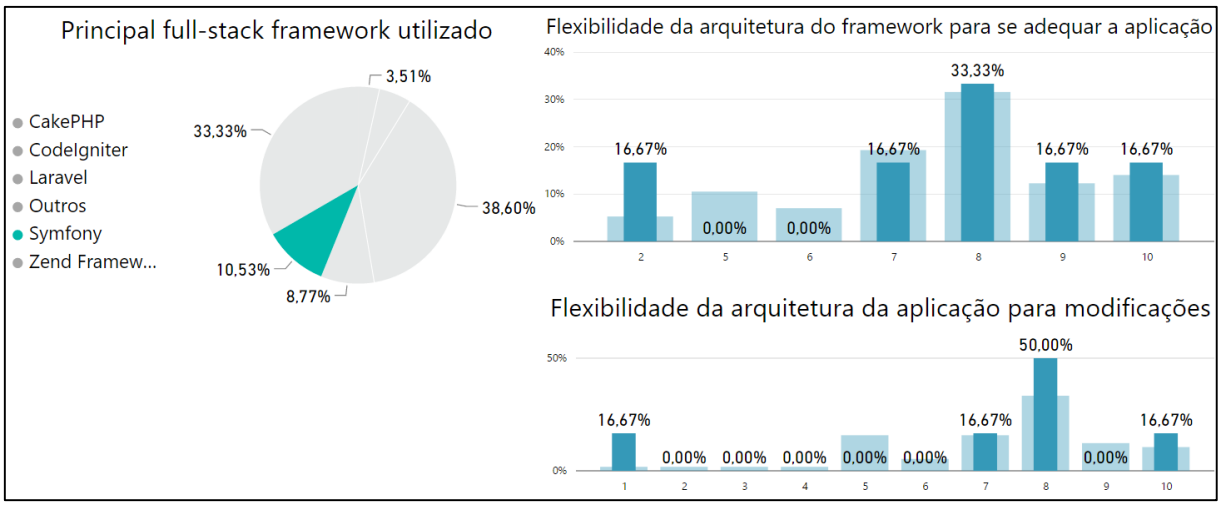

Gráfico 26- Associação entre a questão 25 e 26 entre participantes que utilizaram framework Symfony
Fonte: Próprio autor

Com relação ao *full-stack framework* Symfony, houve uma queda de porcentagem em relação ao nível de classificação 9. Antes, na pergunta anterior representava 16,67% e na atual pergunta 0% dos participantes; também houve queda no percentual de classificação do nível 2, passando de 16,67% para 0%. O nível 1 que antes não aparecia na classificação, na pergunta atual conteve 16,67% dos entrevistados. Observa-se então, que entrevistados que relataram ter o *framework* Symfony como principal, elencaram também uma baixa da aplicação no suporte a aptidão para modificações.

Interrogou-se ao participante sobre a dificuldade ao adicionar novas funcionalidades à aplicação após a conclusão do seu desenvolvimento, sem interferir em outras funcionalidades já desenvolvidas. O Gráfico 27 exibe a classificação dos entrevistados a respeito da dificuldade em se adicionar novas funcionalidades à aplicação após a fase de desenvolvimento.

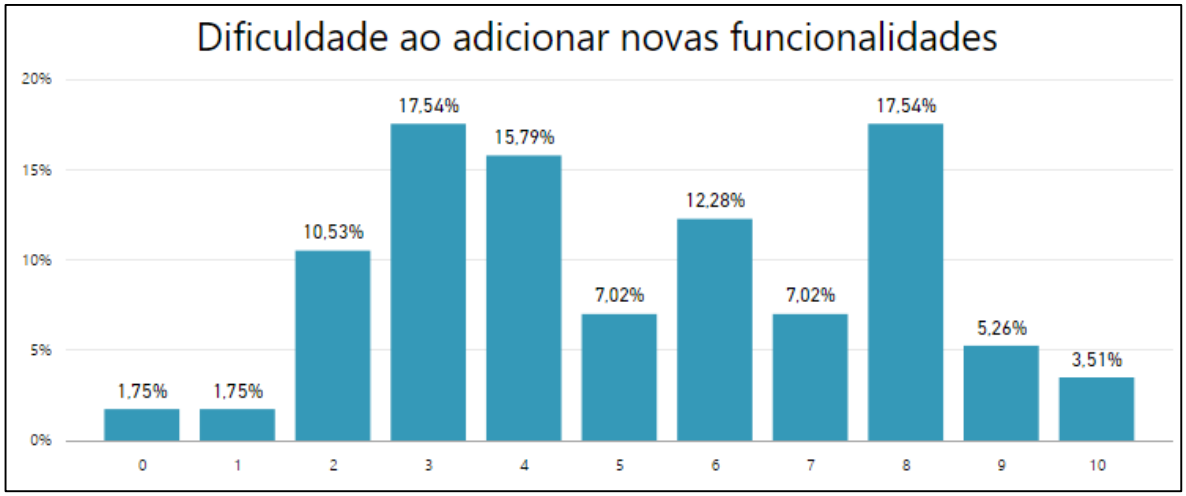

Gráfico 27 - Qual sua classificação em termos de dificuldade, ao adicionar novas funcionalidades à aplicação após sua conclusão, sem interferir em funcionalidades já desenvolvidas?

Fonte: Próprio autor

Percebe-se que a classificação dos entrevistados de forma geral, remete a um nível de dificuldade mediano. No entanto, para extrair uma melhor informação dos dados, a questão foi associada com uma das próximas, que interrogava ao participante se antes de iniciar o desenvolvimento da aplicação, foi realizada algum projeto arquitetural.

Realizada a associação, percebeu-se que participantes que declararam não terem realizado um projeto arquitetural para a aplicação, relataram também um nível de dificuldade um pouco acima da média de classificação apontada; no entanto, entre participantes que afirmaram terem realizado um projeto arquitetural, os valores do percentual de classificação de dificuldade caíram, embora tenham continuado próximos ao nível 5. Aproximadamente 61,40% dos participantes declararam terem realizado um projeto arquitetural antes de iniciar o desenvolvimento da aplicação, cerca de 38,60% afirmaram não terem realizado tal ato.

Ainda se tratando de características após o desenvolvimento da aplicação, fora perguntado ao entrevistado como o mesmo classifica o seu nível de desempenho.

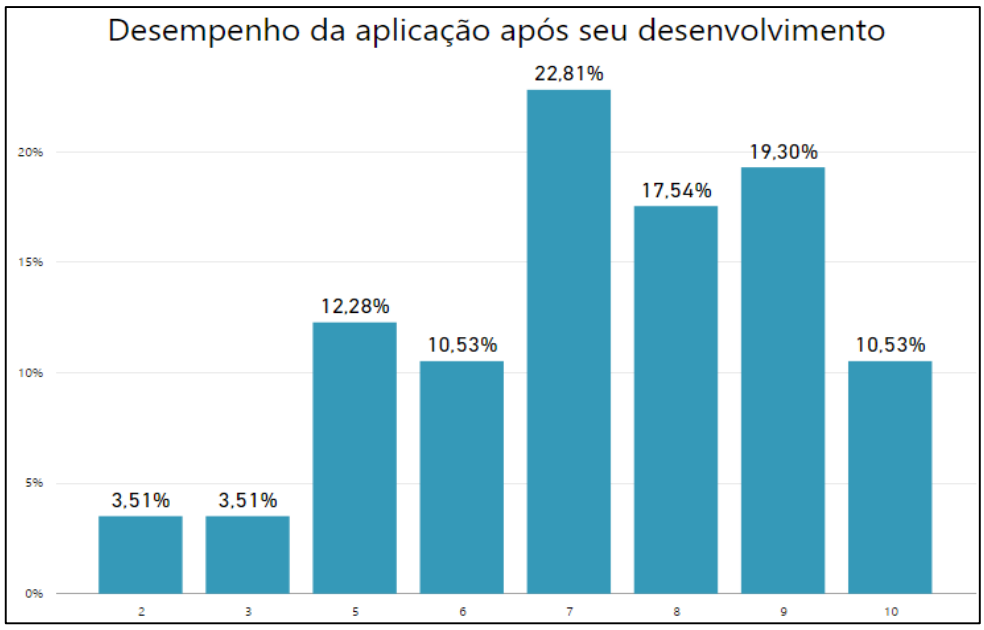

Gráfico 28 - Como você avalia o nível de desempenho dessa aplicação depois de desenvolvida?
Fonte: Próprio autor

Boa parte dos entrevistados atribuíram um bom nível de desempenho a aplicação após finalizado sua fase de desenvolvimento. Tendo como nível 7 o mais apontado com 28,81%, seguido do nível 9 com 19,30% e nível 8 como em terceiro com 17,54%.

A pergunta também foi associada com a qual interrogava ao participante se houve projeto arquitetural para a aplicação, assim como também, com a pergunta sobre a flexibilidade

da aplicação durante seu desenvolvimento e sobre o nível de dificuldade os entrevistados declararam ter, no que diz respeito ao se adicionar novas funcionalidades após a aplicação ser desenvolvida sem interferir em outras funcionalidades; isto mediante ao nível de desempenho mais alto apontado pelos participantes (10).

Realizada a associação, notou-se que a maioria dos participantes que atingiram desempenho com este nível, realizaram projeto arquitetural para aplicação antes de começar seu desenvolvimento, apenas um participante relatou ter conseguido o mesmo resultado, sem realizar projeto de arquitetura para a aplicação. A maior parte do grupo, também relatou ter menor dificuldade no que diz respeito ao acréscimo de funcionalidades à aplicação, sem intervir em outras funcionalidades.

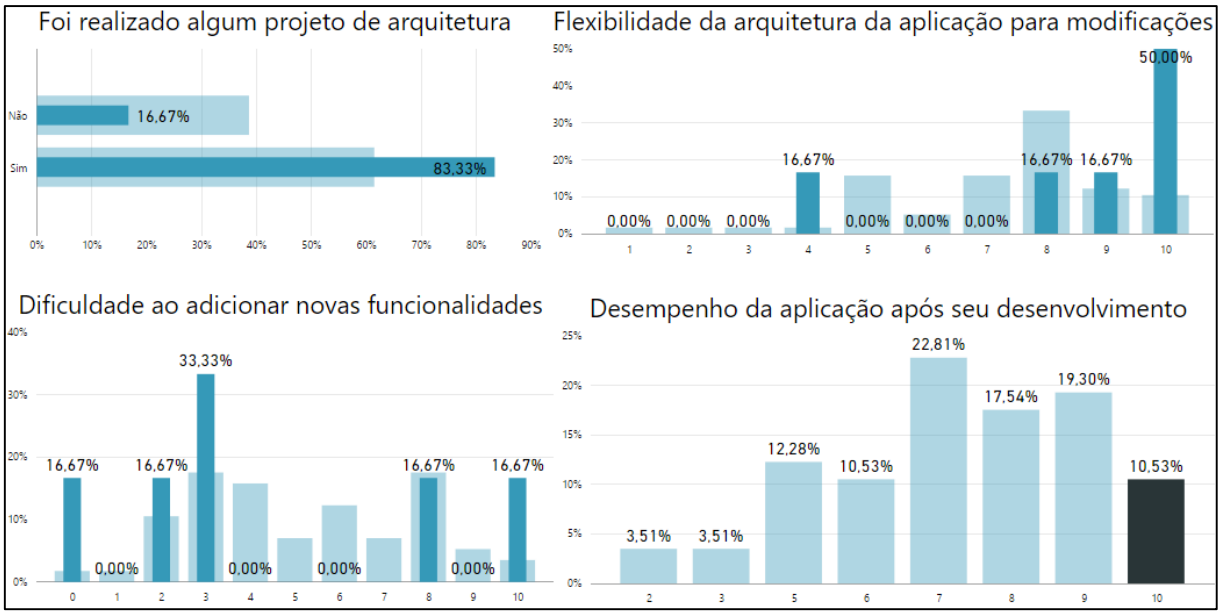

Gráfico 29- Associação ente as perguntas 29, 26, 27 e 28
Fonte: Próprio autor

O entrevistado ainda foi indagado sobre o motivo que influenciou na escolha por um *full-stack framework* para a construção da aplicação. O Gráfico 30 mostra os principais motivos declarados pelos entrevistados ao escolher um *full-stack framework* para o desenvolvimento da aplicação.

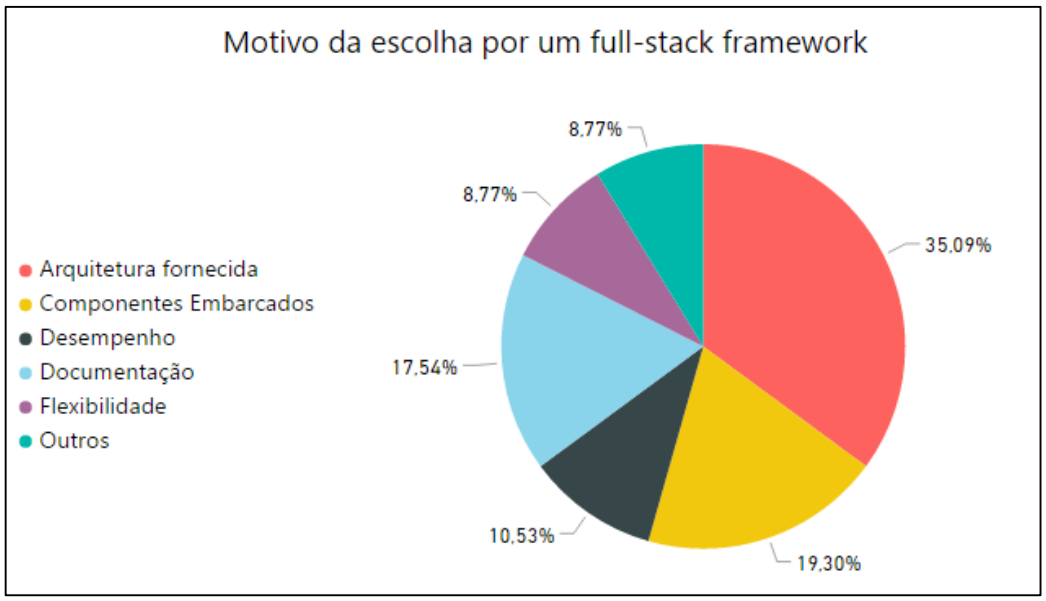

Gráfico 30- Qual o principal motivo influenciou na escolha de um framework full-stack para o desenvolvimento?
Próprio autor

O principal motivo declarado pelos participantes por utilizarem um *full-stack* para o desenvolvimento da aplicação, foi a "Arquitetura fornecida", indicado por 35,09% dos entrevistados, logo após "Componentes embarcados" é o segundo motivo mais indicado com 19,30% e "Documentação" com 17,54%. Esses foram os três motivos mais declarados pela escolha de um *full-stack framework* para o desenvolvimento de uma aplicação segundo os participantes.

O entrevistado foi perguntado sobre qual cenário de arquitetura houve o emprego da utilização do *framework*. O Gráfico 31 mostra as arquiteturas utilizadas pelos participantes.

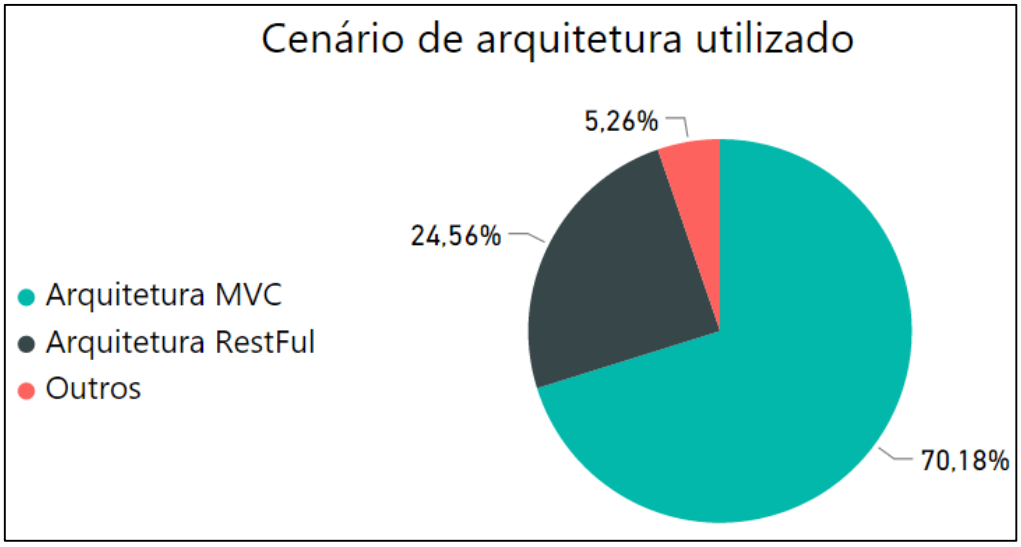

Gráfico 31 - Em qual cenário de arquitetura foi utilizado o framework?
Fonte: Próprio autor

A grande maioria dos entrevistados que utilizaram algum *full-stack framework*, 70,18% destes, afirmaram terem utilizado em uma arquitetura MVC, 24,56% declararam terem utilizado em uma arquitetura *RestFul* e 5,26% afirmaram terem utilizado em uma outra arquitetura. Pode-se inferir que um dos indícios da arquitetura MVC ser a mais utilizada, seja pelo fato de que grande parte dos *frameworks* desta terminologia são acompanhados desta arquitetura em seu alicerce. Associando dados dessa questão com o motivo de utilização declarado pelos entrevistados, estes indícios são reforçados, pois percebe-se que a maioria dos participantes que declararam utilizar o *framework* em um cenário de arquitetura MVC, teve como principal motivo a "Arquitetura fornecida". A associação dos dados pode ser vista no Gráfico 32.

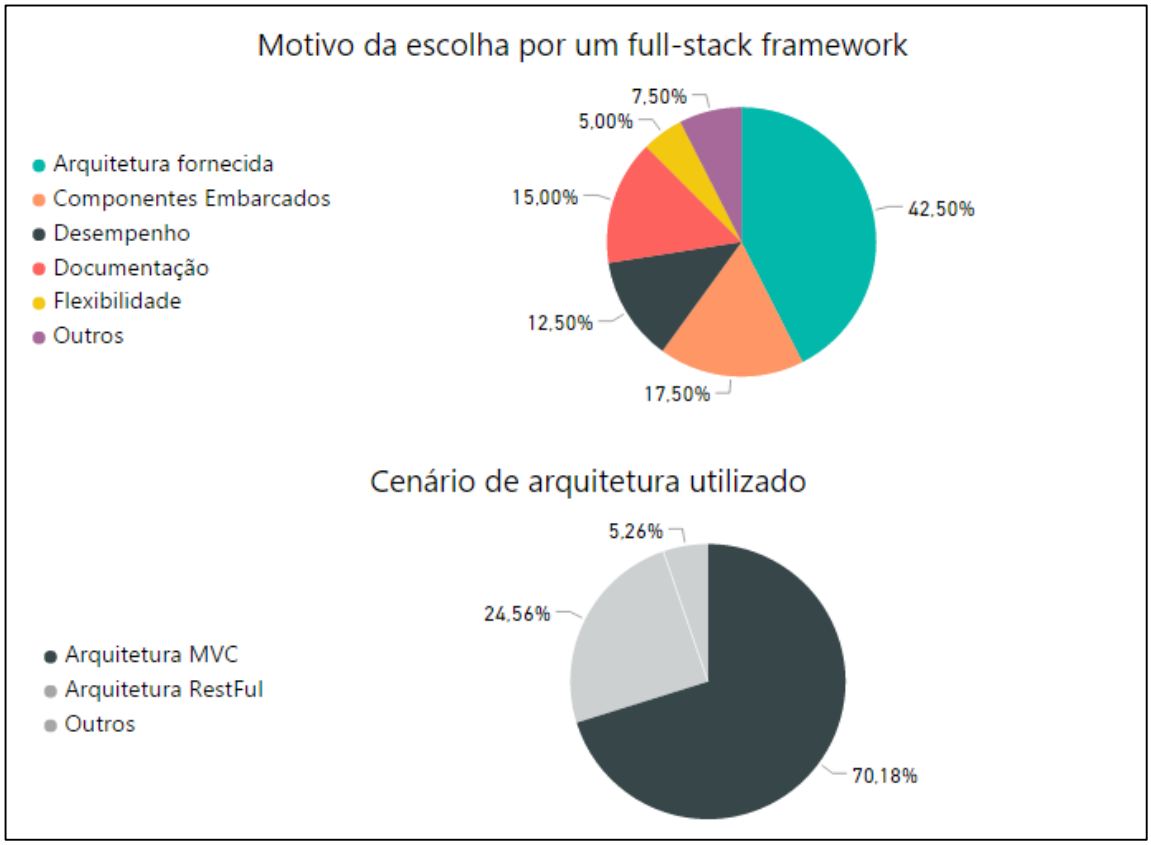

Gráfico 32 - Associa cenário de arquitetura MVC com motivos de utilização do framework
Fonte: Próprio autor

O participante também foi questionado sobre as metodologias de desenvolvimento utilizadas durante a fase de construção da aplicação. Foi permitido ao entrevistado que marcasse mais de uma opção nesta interrogação, pois novamente segundo Gomes(2013), nenhum método é completo e nenhum é perfeito, portanto é comum que se encontre equipes utilizando junções

de acordo com a cultura da organização, do contexto do projeto ou objetivo em questão.

O Gráfico 33 mostra as metodologias mais utilizadas declaradas pelos entrevistados, inclusive suas junções.

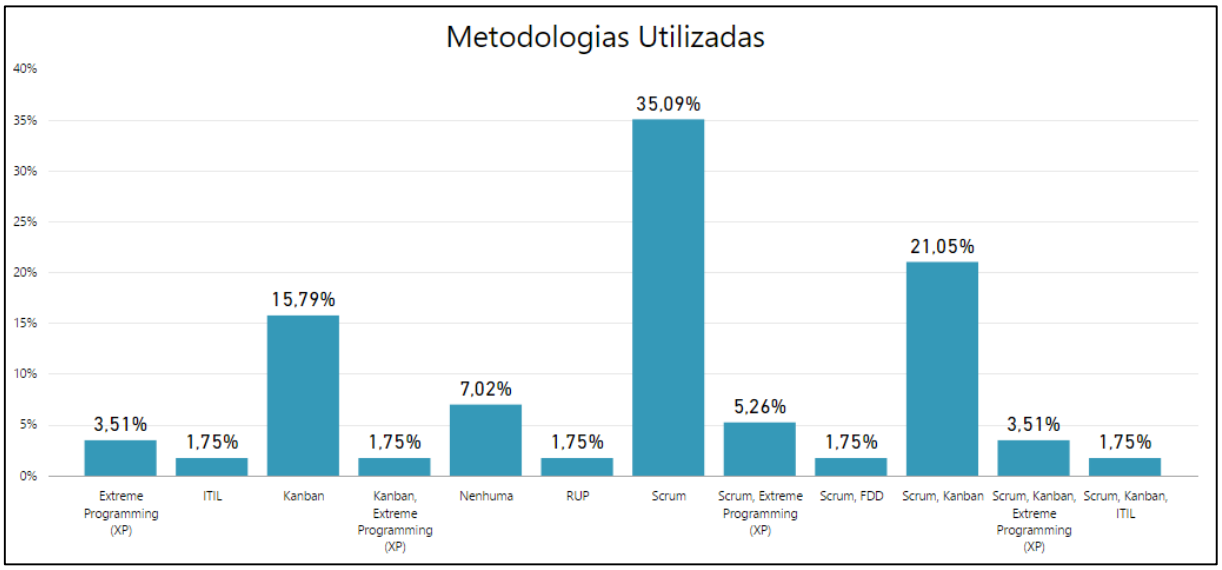

Gráfico 33- Quais metodologias de desenvolvimento foram utilizadas durante a construção da aplicação?
Fonte: Próprio autor

Grande parte dos participantes, relativo a 35,09%, declararam utilizar Scrum como principal metodologia de desenvolvimento, 21,05% utilizaram Scrum e Kanban durante a construção da aplicação, 15,79% utilizaram apenas Kanban, 7,02% dos entrevistados declararam não terem utilizado nenhuma metodologia para o desenvolvimento, o restante dos entrevistados declararam utilizar outra metodologia de desenvolvimento assim como também junções entre elas.

3.1.5 Redesenvolvimento da Aplicação Buscando Escalabilidade

Depois de captar experiência do entrevistado, sobre aplicações escaláveis, desenvolvimento com *micro-frameworks* e *full-stack frameworks*, esta seção teve como objetivo saber com base na experiência do entrevistado, quais decisões ou preferências teria ao realizar o redesenvolvimento de uma aplicação preparando-a para a evolução ou alcançar um bom nível de escalabilidade. Por anto, as próximas três questões analisadas, terão dados apenas de entrevistados que declararam terem alguma experiência com o desenvolvimento utilizando

alguma das terminologias abordadas (*micro-framework*, *full-stack framework*).

Fora perguntado então, qual o tipo de *framework* o entrevistado escolheria se fosse realizar o redesenvolvimento da aplicação preparando-a para evolução. O Gráfico 34 apresenta as terminologias escolhidas pelos entrevistados para o redesenvolvimento da aplicação, com o objetivo de atingir um bom nível de escalabilidade.

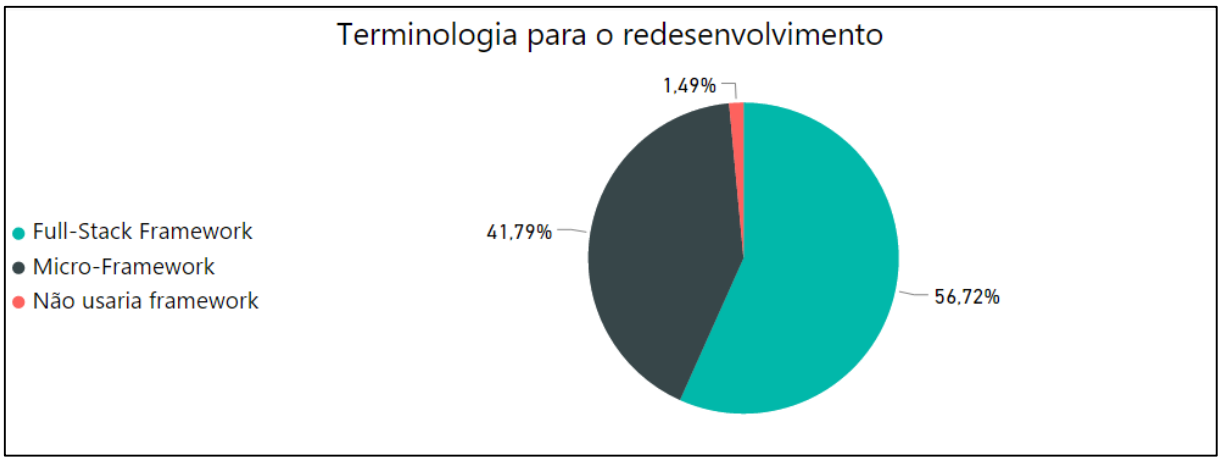

Gráfico 34 - Se você fosse realizar o redesenvolvimento dessa aplicação, preparando-a para evolução; qual tipo de framework escolheria?
Fonte: Próprio autor

Grande parte dos entrevistados afirmaram que optariam por escolher um *full-stack framework* se fossem realizar a reconstrução de uma aplicação preparando-a para escalar. Entretanto, como o principal perfil buscado eram de profissionais que apresentaram um conhecimento mais profundo sobre aplicações escaláveis, realizando um filtro das respostas obtidas desta questão, levando em consideração entrevistados que declararam um alto nível de conhecimento sobre o assunto, percebe-se que gradativamente a medida que aumenta o nível de conhecimento, os entrevistados afirmam optar pela terminologia *micro-framework* se fossem realizar o procedimento.

O Gráfico 35 mostra a associação entre participantes que declararam níveis de conhecimento mais profundo, com quantidade de projetos escaláveis afirmaram ter participado e a terminologia de *framework* escolhida por estes se fossem realizar o redesenvolvimento da aplicação em busca de escalabilidade. Apenas participantes com níveis de conhecimento 9 e 10 foram levados em consideração no filtro.

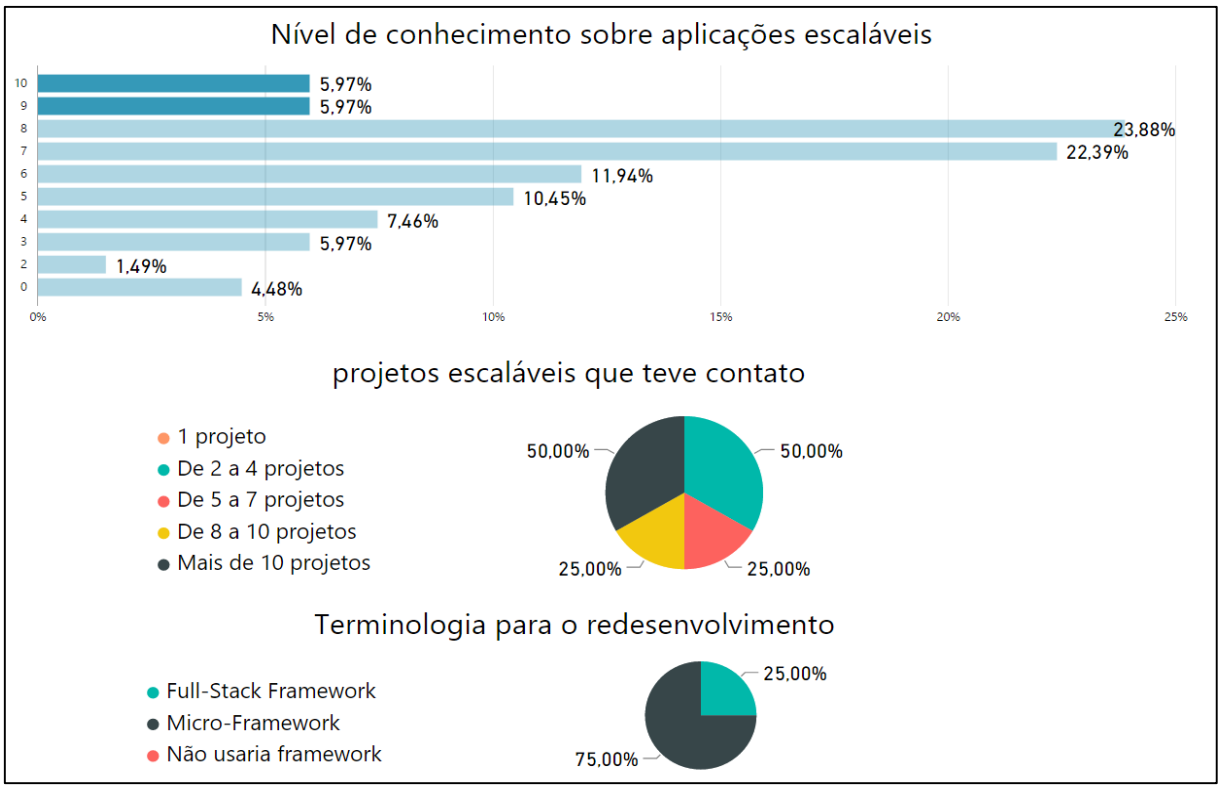

Gráfico 35 - Maiores níveis de conhecimento, quantidade de projetos que teve contato e terminologia escolhida para redesenvolvimento
Fonte: Próprio autor

Nota-se que participantes que declaram níveis de conhecimento mais profundo sobre o assunto, também participaram de mais projetos escaláveis e grande maioria afirmou optar pela terminologia *micro-framework* se fosse realizar o redesenvolvimento da aplicação em busca de escalabilidade.

O participante foi indagado sobre o motivo da escolha ao optar pela terminologia indicada na questão anterior. O Gráfico 36 mostra os principais motivos indicados pelos participantes.

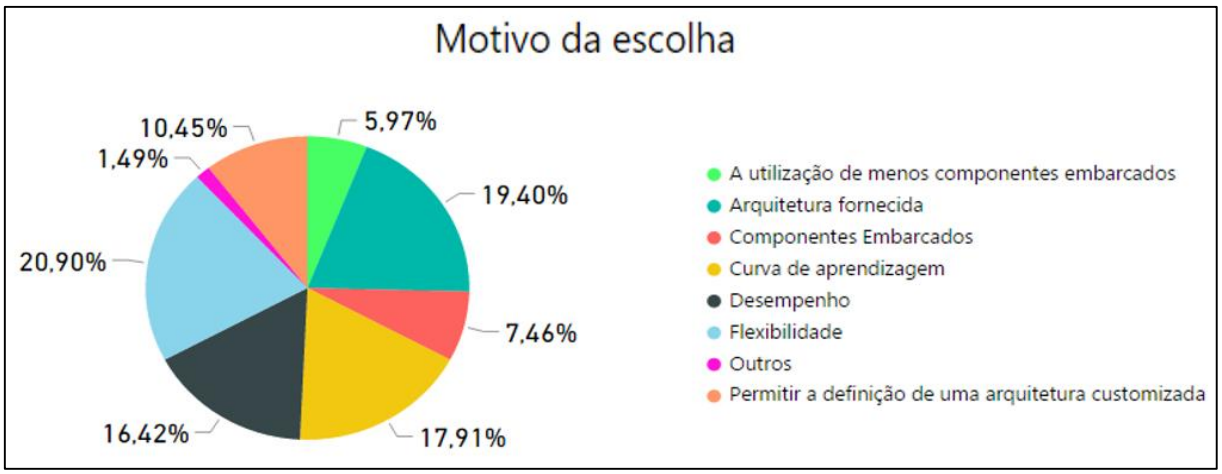

Gráfico 36 - Baseando-se na resposta anterior, qual o principal motivo da escolha?
Fonte: Próprio autor

O principal motivo indicado pelos entrevistados sobre a escolha na pergunta anterior, foi a "Flexibilidade" de modo geral que a terminologia oferece, com 20,90%. Como segundo motivo mais indicado tem-se "Arquitetura Fornecida" com 19,40%, "Curva de aprendizagem" em terceiro com 17,91% dos participantes, e "Desempenho" com 16,42% dos entrevistados.

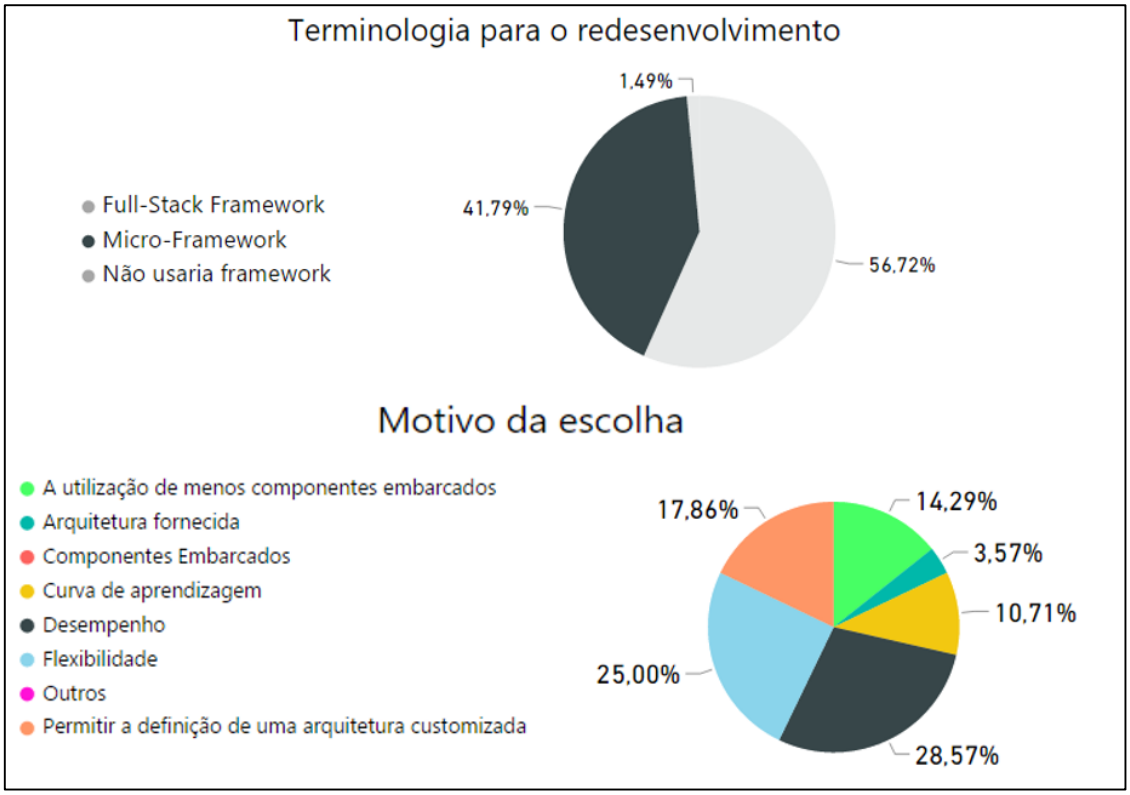

Gráfico 37 - Entrevistados que optaram por *micro-frameworks* e motivos da escolha apontados
Fonte: Próprio autor

Realizando uma associação entre o motivo da escolha indicado pelo participante, com a terminologia escolhida pelo mesmo, observa-se que a maioria que optou pela terminologia

micro-framework para desenvolver uma aplicação buscando alcançar tais objetivos, realizou a escolha pelo desempenho que o mesmo oferece; flexibilidade foi o segundo motivo mais indicado.

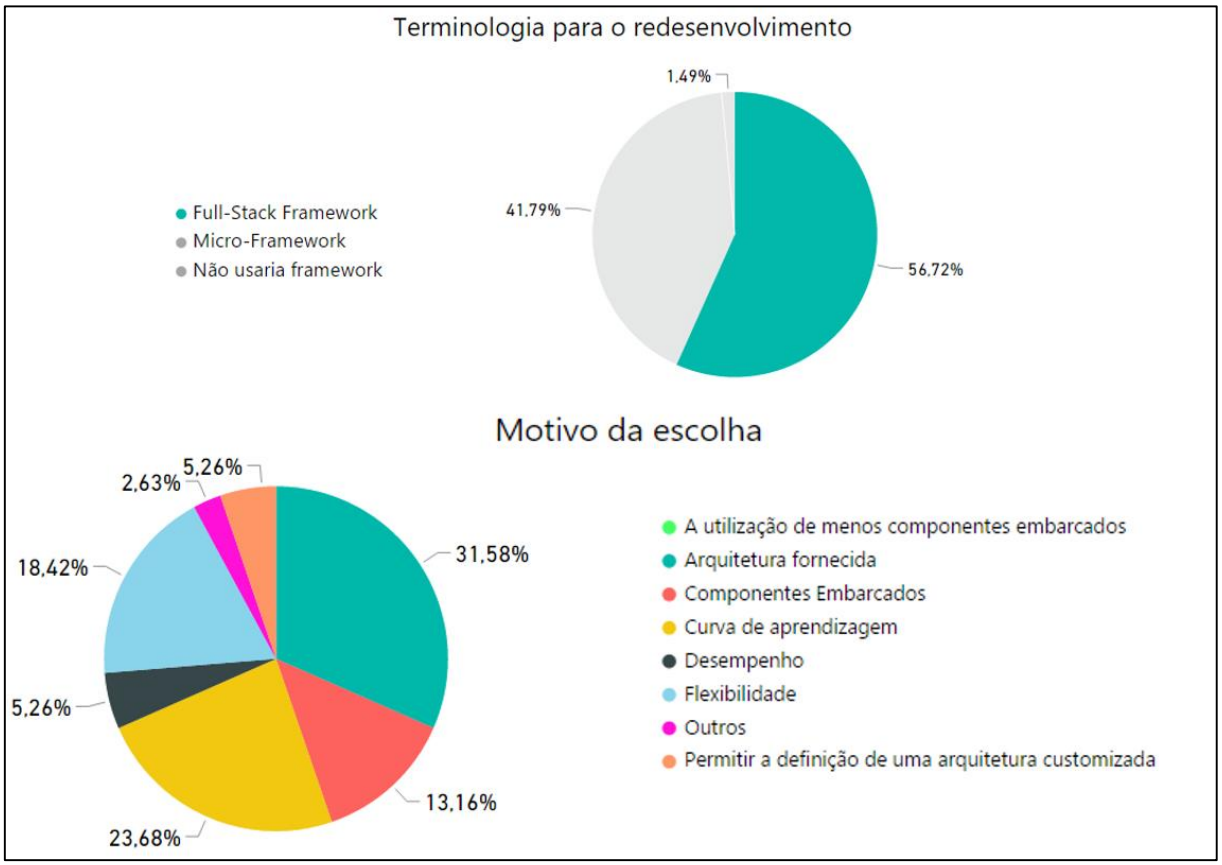

Gráfico 38 - Entrevistados que optaram por *full-stack frameworks* e motivos da escolha apontados
Fonte: Próprio autor

Como motivo mais apontado por entrevistados que optaram por um *full-stack framework*, tem-se a arquitetura fornecida para iniciar o desenvolvimento, em segundo tem-se curva de aprendizagem sobre a terminologia, e como terceiro motivo mais apontado pelos entrevistados tem-se a flexibilidade.

Após as perguntas acima discutidas, o entrevistado foi questionado sobre o possível *framework* que escolheria para o redesenvolvimento de alguma aplicação afim de que a mesma atinja um bom nível de escalabilidade.

Entre participantes que optaram pela terminologia *micro-framework*, Lumen com 28,57% foi o mais apontado pelos entrevistados como escolha para desempenhar o processo, em seguida tem-se Silex com 25% dos entrevistados.

Dentre os *frameworks* de terminologia *full-stack*, o mais indicado foi Laravel com 65,79% dos participantes e em seguida Zend Framework com 18,42%.

CONCLUSÃO

O presente trabalho, teve como objetivo analisar os pontos de escalabilidade: desempenho e manutenibilidade no desenvolvimento de aplicações web utilizando *full-stack framework* e *micro-framework* em linguagem PHP.

Com a análise dos resultados captados, sobre o critério de manutenibilidade:

- Na fase inicial do desenvolvimento, sobre a adaptação da arquitetura do *framework* às necessidades da aplicação, ambas as terminologias obtiveram níveis altos de flexibilidade neste quesito;
- Sobre a flexibilidade da arquitetura da aplicação para modificações durante a fase de desenvolvimento, não se pode certificar que o *framework* utilizado seja somente o responsável pela perda ou melhoria de qualidade neste quesito, no entanto, pode-se inferir que houve um declínio de flexibilidade em aplicações que utilizaram *full-stack framework*, porém em aplicações que fizeram uso de *micro-framework* houveram relatos que esta característica se elevou durante essa etapa.
- Sobre a dificuldade ao se adicionar novas funcionalidades à aplicação após a etapa de desenvolvimento. Em aplicações que utilizaram *full-stack framework* o nível de dificuldade apontado foi um pouco maior do que em relação a aplicações que fizeram uso de *micro-framework*.

Com relação ao critério de desempenho, aplicações que foram desenvolvidas utilizando *micro-framework* apresentaram maior desempenho em relação à aplicações que utilizaram *full-stack framework.*

No entanto é importante frisar que a realização de um projeto arquitetural para a aplicação foi de suma importância e decisiva na qualidade final dos atributos analisados, pois para ambos os modelos utilizados em aplicações que houveram projeto arquitetural, apresentaram maior grau de flexibilidade na arquitetura, menor dificuldade em adicionar novas funcionalidades e maior desempenho.

Sobre a reconstrução das aplicações em busca de escalabilidade, entrevistados com níveis de conhecimento elevado optaram por utilizar um *micro-framework* para alcançar este objetivo, o motivo mais indicado se trata da curva de aprendizagem. Isso se explica pelo fato

de que entrevistados que apresentaram nível de conhecimento elevado, também possuem maior experiência com o desenvolvimento de aplicações escaláveis, dessa forma podem tomar melhores decisões pois possuem um maior conhecimento, e pelo que se observa *micro-frameworks* seriam ferramentas que possibilitariam autonomia para estas decisões.

Com este trabalho foi possível demonstrar que para o desenvolvimento de aplicações web escaláveis em linguagem PHP, sobre os critérios de manutenibilidade e desempenho, os modelos *micro-framework* e *full-stack* atingiram bons níveis de qualidade sobre a garantia de um projeto arquitetural, ou seja um planejamento para a arquitetura da aplicação foi o fator decisivo para alcançar tais atributos de qualidade. Dessa forma a pesquisa alcançou seu objetivo servindo de auxílio para o desenvolvimento de aplicações deste nicho, destacando a importância do planejamento da arquitetura de um *software* para a qualidade final do mesmo.

TRABALHOS FUTUROS

Como sugestões para trabalhos futuros, seria interessante realizar uma estudo de quais boas práticas seguir durante o desenvolvimento de uma aplicação web escalável, para alcançar um bom nível de escalabilidade.

Outra sugestão, seria a análise de mais pontos de escalabilidade voltados ao código do *software*. Mais pontos analisados servirão de auxílio para que desenvolvedores e arquitetos de *software* possam realizar uma melhor escolha sobre qual modelo de *framework* mais se adequa às necessidades de seu *software* a ser desenvolvido.

Um outra sugestão, seria analisar a predisposição para escalabilidade nos modelos de *frameworks* abordados em outras linguagens. Além de analisar qual estilo de arquitetura supri melhor as necessidades de uma aplicação web escalável.

REFERÊNCIAS

AQUILES, A.; FERREIRA, R. Controlando Versões com o Git e GitHub. **Casa do Código**, p. 204, 2014.

BACHMANN, F. et al. **Documenting Software Architecture: Documenting BehaviourCarnegie Mellon University Technical Note**.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture In Practice, Second Edition**. 2. ed. Addison Wesley, 2003.

BENTO, E. J. **Desenvolvimento web com PHP e MySQL**. Casa do Código, 2014.

BONDI, A. Characteristics of Scalability and Their Impact on Performance. **Proceedings of the 2nd international workshop on Software and performance**, p. 195–203, 2000.

BROOKS, F. P. No Silver Bullet — Essence and Accident in Software Engineering. **University of North Carolina at Chapel Hill**, p. 1–16, 1987.

BuiltWith Trends. Disponível em: <https://trends.builtwith.com/framework>. Acesso em: 15 nov. 2016.

BUSCHMANN, F. et al. **Pattern Oriented Software Architecture Volume 1: A System of Patterns**. v. Vol. 1

BUSTAMANTE, T. D. R. DE. CRUX: UM ARCABOUÇO DE SOFTWARE PARA DESENVOLVIMENTO DE APLICAÇÕES WEB. **Universidade Federal de Minas Gerais**, 2008.

DUBOC, L.; ROSENBLUM, D.; WICKS, T. A framework for characterization and analysis of software system scalability. **Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering**, p. 375–384, 2007.

FAYAD, M.; SCHMIDT, D. C.; JOHNSON, R. E. **Building Application Frameworks: Object-Oriented Foundations of Framework Design**. Wiley, 1999.

FOWLER, M. **Padrões de arquitetura de aplicações corporativas.pdf**. Porto Alegre: .

GARLAN, D.; PERRY, D. Introduction to the Special Issue on Software Architecture. **Practice**, p. 1–6, 1995.

GERMOGLIO, G. **Arquitetura de Software**. Houston, Texas: .

GIANNINI, N. J. **Vulnerable Web Application Framework**. University of Rhode Island, 2015.

GINIGE, A.; MURUGESAN, S. Web Engineering: A Methodology for Developing Scalable, Maintainable Web Applications. **Cutter IT Journal**, v. 14, n. 7, p. 24–35, 2001a.

GINIGE, A.; MURUGESAN, S. Web engineering: an introduction. **IEEE Multimedia**, v. 8,

n. 1, p. 14–18, 2001b.

GOMES, A. F. **Agile: Desenvolvimento de Software com entregas frequentes e foco no valor de negócio**. São Paulo: Casa do Código, 2013.

HENDERSON, C. **Building Scalable Web Sites**.

HILDRETH, S. **Buggy Software: Up From a Low-Quality Quagmire**. Disponível em: <http://www.computerworld.com/article/2557403/app-development/buggy-software--up-from-a-low-quality-quagmire.html>. Acesso em: 15 out. 2016.

HILL, M. D. What is scalability? **ACM SIGARCH Computer Architecture News**, v. 18, n. 4, p. 18–21, 1990.

JOGALEKAR, P.; WOODSIDE, M.; MEMBER, S. Evaluating the Scalability of Managed Distributed Systems. **IEEE Transactions on Parallel and Distributed Systems**, v. 11, n. 6, p. 589–603, 2000.

LEVINSON, M. **Let's Stop Wasting $78 billion a Yer**. Disponível em: <http://www.cio.com/article/2441228/enterprise-software/software-development---let-s-stop-wasting--78-billion-a-year.html>. Acesso em: 15 out. 2016.

LISBOA, F. **Criando Aplicações PHP com Zend Framework e Dojo**. 2. ed. São Paulo: .

LOUDON, K. **Desenvolvimento de Grandes Aplicações Web**. São Paulo: Novatec Editora Ltda., 2010.

MARKIEWICZ, M. E.; LUCENA, C. Understanding Object-Oriented Framework Engineering. **PUC-RioInf.MCC38**, p. 11, 2000.

MAZZA, L. **HTML5 e CSS3 - Domine a web do futuro**. 1ª Ed ed. São Paulo: . v. 53

MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. Factors in Software Quality. Volume-III. Preliminary Handbook on Software Quality for an Acquisiton Manager. v. III, n. November, p. 42, 1977.

MENDES, A. **Arquitetura de Software: Desenvolvimento Orientado para Arquitetura**. 1. ed. Rio de Janeiro: Editora Campus, 2002.

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

NEUMAN, B. C. Scale in Distributed Systems. **Readings in Distributed Computing Systems**, 1994.

OLIVEIRA, E. et al. Compreensão de Aplicações Web : O Processo e as Ferramentas Resumo. **6ª Conferência da Associação Portuguesa de Sistemas de Informação**, 2005.

PERRY, D. E.; WOLF, A. L. Foundations for the study of software architecture. **ACM SIGSOFT Software Engineering Notes**, v. 17, n. 4, p. 40–52, 1992.

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

PORTO, I. O. PADRÕES E DIRETRIZES ARQUITETURAIS PARA ESCALABILIDADE DE SISTEMAS. **UNIVERSIDADE FEDERAL DE UBERLÂNDIA FACULDADE DE CIÊNCIA DA COMPUTAÇÃO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**, 2009.

Post The MicroPHP Manifesto. Disponível em: <https://funkatron.com/posts/the-microphp-manifesto.html>.

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. **Roger S.Pressman - Engenharia de Software 7° Edição - Uma Abordagem Profissional.pdf**, 2011.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software uma Abordagem Profissional**. 8. ed. São Paulo: AMGH Editora Ltda, 2016.

REDMONK. **The RedMonk Programming Language Rankings: January 2016**. Disponível em: <http://redmonk.com/sograd y/2016/02/19/language-rankings-1-16/>. Acesso em: 15 out. 2016.

RUSSELL, C. **PHP Development Tool Essentials**. Berkeley, CA: Apress, 2016.

SMITH, C.; WILLIAMS, L. Building responsive and scalable web applications. **Cmg-Conference-**, p. 1–11, 2000.

SOMMERVILLE, I. **ENGENHARIA DE SOFTWARE**. 9. ed. São Paulo: PEARSON BRASIL, 2012.

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: 16 nov. 2016.

The MicroPHP Manifesto. Disponível em: <http://microphp.org/>.

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

TYREE, J.; AKERMAN, A. Architecture decisions: Demystifying architecture. **IEEE Software**, v. 22, n. 2, p. 19–27, 2005.

VAQQAS, M. **RESTful Web Services: A Tutorial**. Disponível em: <http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069>. Acesso em: 27 nov. 2016.

APÊNDICE 1 – QUESTIONÁRIO

Desenvolvimento de Aplicações Escaláveis

Este trabalho tem como objetivo analisar alguns pontos com relação a escalabilidade de aplicações desenvolvidas com os paradigmas de: Micro-frameworks e Frameworks Full-Stack.

Os dados coletados são confidenciais e serão utilizados para fins acadêmicos (auxiliar na elaboração de meu trabalho de conclusão de curso de Ciência da Computação). Os resultados serão divulgados na comunidade para que desenvolvedores, arquitetos e quem mais tiver interesse, possa fazer uso.

Sua colaboração é muito importante pois dará auxilio na minha trajetória em busca de conhecimento.

Desde já, obrigado pela sua contribuição!

*Obrigatório

Perfil do Participante

As seguintes questões têm como objetivo conhecer um pouco do seu perfil.

1. **Caso queria receber a conclusão do trabalho informe seu e-mail:**

   _____

2. **País:** *

   _____

3. **Estado:** *

   _____

4. **Idade** *
   Marcar apenas uma oval.

   - ◯ Entre 16 e 25
   - ◯ Entre 26 e 35
   - ◯ Entre 36 e 45
   - ◯ Entre 46 e 55
   - ◯ Acima de 56

5. **Qual sua formação acadêmica?** *
Marcar apenas uma oval.

- ◯ Ensino Médio Completo
- ◯ Técnico
- ◯ Ensino Superior Completo ou Cursando
- ◯ Pós-Graduação
- ◯ Mestrado
- ◯ Doutorado
- ◯ Outro: _____

6. **A quanto tempo atua na área de TI?** *
Marcar apenas uma oval.

- ◯ De 0 a 12 meses
- ◯ De 1 a 3 anos
- ◯ De 4 a 6 anos
- ◯ De 7 a 10 anos
- ◯ Mais de 10 anos
- ◯ Não atuo na área

7. **Atualmente, qual cargo você ocupa?** *
Marcar apenas uma oval.

- ◯ Desenvolvedor / Programador
- ◯ Analista de Sistemas
- ◯ Arquiteto de Software
- ◯ Gerente de Projetos
- ◯ Estagiário
- ◯ Outro:

8. **Quantos funcionários aproximadamente atuam na empresa que você presta serviços?** *
Marcar apenas uma oval.

- ◯ Apenas 1 funcionário
- ◯ De 2 a 20 funcionários
- ◯ De 21 a 40 funcionários
- ◯ De 41 a 60 funcionários
- ◯ De 61 a 80 funcionários
- ◯ De 81 a 100 funcionários
- ◯ Mais de 100 funcionários

Desenvolvimento de Aplicações Escaláveis

As perguntas a seguir tem como objetivo definir o grau de experiência com relação a aplicações escaláveis.

9. **Qual seu nível de conhecimento sobre aplicações escaláveis?** *

Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
Nenhum	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Profundo

10. **Você já trabalhou em algum projeto ou no desenvolvimento de alguma aplicação escalável em linguagem PHP?** *

Marcar apenas uma oval.

- ◯ Sim *Ir para a pergunta 11.*
- ◯ Não *Ir para a pergunta 13.*

Sobre as Aplicações Escaláveis ao qual teve contato

11. **Quantos projetos escaláveis teve contato?** *

Marcar apenas uma oval.

- ◯ 1 projeto
- ◯ De 2 a 4 projetos
- ◯ De 5 a 7 projetos
- ◯ De 8 a 10 projetos
- ◯ Mais de 10 projetos

12. **Em média, quantas pessoas eram envolvidas em cada projeto?** *

Marcar apenas uma oval.

- ◯ Apenas eu
- ◯ De 2 a 4 pessoas
- ◯ De 5 a 7 pessoas
- ◯ De 8 a 10 pessoas
- ◯ Mais de 10 pessoas

13. **Das aplicações que você participou do desenvolvimento, foi utilizado algum micro-framework?** *

Marcar apenas uma oval.

- ◯ Sim *Ir para a pergunta 14.*
- ◯ Não *Ir para a pergunta 23.*

Sobre o uso de micro-frameworks

14. **Qual o principal micro-framework utilizado?** *

Marcar apenas uma oval.

- ◯ Slim
- ◯ Silex
- ◯ Lumen
- ◯ Zend Expressive
- ◯ Limonade
- ◯ Wave
- ◯ Outro: _____

15. **Como você avalia o nível de flexibilidade da arquitetura do framework utilizado, para se adequar as necessidades da aplicação e permitir a evolução da mesma?** *

Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
Nenhum	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

16. **Como você avalia o nível de flexibilidade da arquitetura dessa aplicação, ao sofrer modificações durante seu processo de desenvolvimento?** *

Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
Nenhum	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

17. **Qual sua classificação em termos de dificuldade, ao adicionar novas funcionalidades à aplicação após sua conclusão, sem interferir em funcionalidades já desenvolvidas?** *

Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
Nenhuma	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

18. **Como você avalia o nível de desempenho dessa aplicação depois de desenvolvida?** *

Marcar apenas uma oval.

	1	2	3	4	5	6	7	8	9	10	
Muito Baixo	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

19. **Antes de dar início ao desenvolvimento dessa aplicação, foi realizado algum projeto arquitetural para a mesma?**

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

20. **Qual o principal motivo influenciou na escolha de um micro-framework para o desenvolvimento?** *

Marcar apenas uma oval.

- ◯ Flexibilidade
- ◯ Permitir a definição de uma arquitetura customizada
- ◯ A utilização de menos componentes embarcados
- ◯ Desempenho
- ◯ Curva de aprendizagem
- ◯ Outro: _____

21. **Em qual cenário de arquitetura foi utilizado o framework?** *

Marcar apenas uma oval.

- ◯ Arquitetura RestFul
- ◯ Arquitetura MVC
- ◯ Outro:

22. **Quais metodologias de desenvolvimento foram utilizadas durante a construção da aplicação?** *

Marque todas que se aplicam.

- ☐ Scrum
- ☐ Kanban
- ☐ RUP
- ☐ Extreme Programming (XP)
- ☐ FDD
- ☐ ITIL
- ☐ Outro: _____

23. **Das aplicações que você participou do desenvolvimento, foi utilizado algum framework full-stack** *

Marcar apenas uma oval.

- ◯ Sim *Ir para a pergunta 24.*
- ◯ Não *Ir para a pergunta 33.*

Sobre o uso dos frameworks full-stack

24. **Qual o principal framework full-stack utilizado?** *

Marcar apenas uma oval.

- ◯ Laravel
- ◯ Symfony
- ◯ Zend Framework
- ◯ CodeIgniter
- ◯ PhalconPHP
- ◯ CakePHP
- ◯ Outro: _____

25. **Como você avalia o nível de flexibilidade da arquitetura do framework utilizado, para se adequar as necessidades da aplicação e permitir a evolução da mesma?** *
Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
Nenhum	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

26. **Como você avalia o nível de flexibilidade da arquitetura dessa aplicação, ao sofrer modificações durante seu processo de desenvolvimento?** *
Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
Nenhum	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

27. **Qual sua classificação em termos de dificuldade, ao adicionar novas funcionalidades à aplicação após sua conclusão, sem interferir em funcionalidades já desenvolvidas?** *
Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
Nenhuma	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

28. **Como você avalia o nível de desempenho dessa aplicação depois de desenvolvida?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	8	9	10	
Muito Baixo	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Muito Alto

29. **Antes de dar início ao desenvolvimento dessa aplicação, foi realizado algum projeto arquitetural para a mesma?** *
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

30. **Qual o principal motivo influenciou na escolha de um framework full-stack para o desenvolvimento?** *
Marcar apenas uma oval.

- ◯ Arquitetura fornecida
- ◯ Componentes Embarcados
- ◯ Desempenho
- ◯ Documentação
- ◯ Flexibilidade
- ◯ Outro: _____

31. **Em qual cenário de arquitetura foi utilizado o framework?** *

Marcar apenas uma oval.

- ◯ Arquitetura RestFul
- ◯ Arquitetura MVC
- ◯ Outro: _____

32. **Quais metodologias de desenvolvimento foram utilizadas durante a construção da aplicação?** *

Marque todas que se aplicam.

- ☐ Scrum
- ☐ Kanban
- ☐ RUP
- ☐ Extreme Programming (XP)
- ☐ FDD
- ☐ ITIL
- ☐ Outro: _____

33. **Se você fosse realizar o redesenvolvimento dessa aplicação, preparando-a para evolução; qual tipo de framework escolheria?** *

Marcar apenas uma oval.

- ◯ Micro-Framework
- ◯ Full-Stack Framework
- ◯ Não usaria framework

34. **Baseando-se na resposta anterior, qual o principal motivo da escolha?** *

Marcar apenas uma oval.

- ◯ Arquitetura fornecida
- ◯ Componentes Embarcados
- ◯ Flexibilidade
- ◯ A utilização de menos componentes embarcados
- ◯ Desempenho
- ◯ Permitir a definição de uma arquitetura customizada
- ◯ Curva de aprendizagem
- ◯ Outro: _____

35. Qual framework seria utilizado? *

Marcar apenas uma oval.

- ◯ Laravel
- ◯ Slim
- ◯ Zend Framework
- ◯ Silex
- ◯ Symfony
- ◯ Lumen
- ◯ CodeIgniter
- ◯ Zend Expressive
- ◯ CakePHP
- ◯ Limonade
- ◯ PhalconPHP
- ◯ Wave
- ◯ Não sei opinar
- ◯ Não usaria framework
- ◯ Outro: _____