INSTITUTO ENSINAR BRASIL
FACULDADES INTEGRADAS DE CARATINGA

TALES SATHLER NOVAIS

MODULARIZAÇÃO DE PROJETOS COMO SOLUÇÃO À COMPLEXIDADE E MUTABILIDADE NA CONSTRUÇÃO DE UM SOFTWARE EM ARQUITETURA MVC

CARATINGA
2017

TALES SATHLER NOVAIS
FACULDADES INTEGRADAS DE CARATINGA

MODULARIZAÇÃO DE PROJETOS COMO SOLUÇÃO À COMPLEXIDADE E MUTABILIDADE NA CONSTRUÇÃO DE UM SOFTWARE EM ARQUITETURA MVC

Monografia apresentada ao Curso de Ciência da Computação das Faculdades Integradas de Caratinga, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração: Engenharia de Software.

Orientador (a): Prof. MSc. Fabrícia Pires Souza.

CARATINGA

2017

rede de ensino DOCTUM

FACULDADES INTEGRADAS DE CARATINGA

FOLHA DE APROVAÇÃO

O trabalho de Conclusão de Curso intitulado: MODULARIZAÇÃO DE PROJETOS COMO SOLUÇÃO À COMPLEXIDADE E MUTABILIDADE NA CONSTRUÇÃO DE UM SOFTWARE EM ARQUITETURA MVC, elaborado pelo aluno TALES SATHLER NOVAIS foi aprovado por todos os membros da Banca Examinadora e aceita pelo curso de Ciência da Computação das Faculdades Integradas de Caratinga, como requisito parcial da obtenção do título de

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Caratinga, 11 de Dezembro 2017

Profª. Fabrícia Pires Souza

Prof. Glauber Luiz Da Silva Costa

Prof. Maicon Vinicius Ribeiro

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me permitido alcançar essa conquista, e sempre ter me dado forças para enfrentar todos os desafios, dificuldades e seguir sempre em frente.

Aos meus amados pais Tadeu e Adriana, por sempre me incentivarem, me ensinando desde cedo o valor dos estudos, sempre estando ao meu lado me motivando a buscar sempre mais. Muito obrigado, por sempre acreditarem em mim, por fazerem dos meus sonos os seus sonhos! À minha namorada Glória Rayane e minha irmã Talita, por sempre me ajudarem e estarem comigo nos momentos bons e ruins, me dando todo o apoio que precisava.

A todos os professores do curso de Ciência da Computação, em especial a minha professora orientadora e coordenadora do curso Fabrícia Pires, por todo o apoio, por ter acreditado em mim e no meu trabalho.

A todos os meus amigos por terem contribuído para minha formação pessoal e acadêmica, em especial aos meus amigos Matheus Ferreira, Lucas Rafael e Edvaldo Rodrigues, por todos os feedbacks e materiais que foram de grande auxílio na elaboração deste estudo.

A todos os meus familiares, por todo o apoio de sempre. Em especial minha avó Joselita, por todos os conselhos, ensinamentos e incentivo que sempre me proporcionou.

ABREVIATURAS E SIGLAS

HTML - Linguagem de Marcação de Hipertexto

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

MVC – *Model-View-Controller*

PHP – *Hypertext Preprocessor*

POO – Programação Orientada a Objetos

TI – Tecnologia da Informação

UML – Linguagem de Modelagem Unificada

WWW – *World Wide Web*

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

RESUMO

A TI (Tecnologia da Informação) nos últimos anos evoluiu e se consolidou em todas as áreas da sociedade, sendo atualmente indispensável na resolução de problemas em construções de soluções de *software* ou análise de dados. Com o passar dos anos, a necessidade de estudos à procura de métodos mais eficientes e mais produtivos no desenvolvimento de soluções de *software* aumentou consideravelmente. A Área de Engenharia de *Software* surgiu propondo metodologias, *frameworks*, padrões de projetos e ferramentas, como soluções às dificuldades no projeto, desenvolvimento e implantação de um sistema. O padrão arquitetural MVC, sendo atualmente um dos mais utilizados no desenvolvimento de aplicações WEB, surgiu para promover a reutilização de código, redução das dependentes dos componentes e além de facilitar a modularidade. A modularidade dos componentes é uma das características arquiteturais no desenvolvimento de um *software* sendo importante para promover a abstração e fraco acoplamento no sistema. Neste trabalho foram apresentados estudos acerca do MVC e da modularidade no desenvolvimento de *software*, bem como suas características. Foi elaborado um questionário qualitativo e distribuído com os profissionais de TI que possuem experiência no desenvolvimento de sistemas, com o intuído de obter resultados acerca de suas experiências ao utilizar o MVC e a modularidade em seus projetos de *software*. Sendo possível confrontar os resultados obtidos através do questionário aplicado, com a parte teórica do estudo. Logo foi possível observar que a utilização do padrão MVC em uma aplicação modular, promove uma melhor utilização de seus recursos principalmente quando a aplicação possui módulos bem construídos e componentes independentes.

Palavras-chave: *Software,* Engenharia de *Software*, modularidade, MVC, arquitetura, padrão de projetos.

ABSTRACT

IT (Information Technology) in recent years has evolved and consolidated in all areas of society, and is currently indispensable in solving problems in software solution solutions or data analysis. Over the years, the need for studies to look for more efficient and more productive methods without the development of software solutions has increased considerably. The Software Engineering Area, methodologies, frameworks, project and tool standards, solutions to difficulties without projects, development and implementation of a system. The MVC architecture standard, which is currently one of the most used in the development of WEB applications, has emerged to promote the reuse of code, reduction of dependents of the components and besides facilitating modularity. The modularity of components is one of the architectural features in the development of software being important to promote abstraction and poor coupling in the system. In this work we have presented studies about MVC and modularity in software development, as well as its characteristics. A qualitative and distributed questionnaire was developed with IT professionals who have experience in systems development, with the aim of obtaining results about their experiences using MVC and modularity in their software projects. It is possible to compare the results obtained through the questionnaire applied, with the theoretical part of the study. It was soon possible to observe that the use of the MVC standard in a modular application promotes a better use of its resources mainly when the application has well-constructed modules and independent components.

Keywords: Software, Software Engineering, modularity, MVC, architecture, project standard.

SUMÁRIO

INTRODUÇÃO

A crise do *software* em 1970, foi um período no qual houve uma alta demanda por soluções de *software* e as empresas de tecnologia não estavam preparadas o suficiente para os desafios de prazos, custos e requisitos de um projeto. Não haviam métodos que possibilitassem um rápido desenvolvimento e ao mesmo tempo cumprindo os prazos e orçamentos estipulados, por consequência disso, os sistemas desenvolvidos eram de baixa qualidade e de alta complexidade de manutenção.

O conceito de *software* está além de simplesmente programas de computadores, cada vez mais estando em todas as áreas da sociedade, existe a necessidade do rápido desenvolvimento de sistemas como solução de rapidez e eficácia em processos complexos, seja no dia a dia ou em grandes produções nas indústrias. Com o intuito de garantir sua qualidade e eficiência, campos de estudo vêm se desenvolvendo para atender uma necessidade maior de melhor planejamento e utilização de métodos capazes de aprimorar o desenvolvimento de um *software*.

> Muitas pessoas pensam que software é simplesmente outra palavra para programas de computador. No entanto, quando falamos de engenharia de software, não se trata apenas do programa em si, mas de toda a documentação associada e dados de configurações necessários para fazer esse programa operar corretamente. Um sistema de software desenvolvido profissionalmente é, com frequência, mais do que apenas um programa; ele normalmente consiste em uma série de programas separados e arquivos de configuração que são usados para configurar esses programas. Isso pode incluir documentação do sistema, que descreve a sua estrutura; documentação do usuário, que explica como usar o sistema; e sites, para usuários baixarem a informação recente do produto. (SOMERVILLE, 2012, p.3)

Quando se fala de um sistema de grande escala, este certamente realizará tarefas com grandes quantidades de dados e consequentemente ocasionará grandes quantidades de funcionalidades. Uma das principais dificuldades durante o desenvolvimento de um software é o quão complexo este poderá chegar e ainda ser capaz de ser flexível o suficiente para ser alterado com facilidade por diferentes profissionais.

> Software, de modo genérico, é uma entidade que se encontra em quase constante estado de mudança. Essas mudanças ocorrem por necessidade de corrigir erros existentes no software e/ou de adicionar novos recursos e funcionalidades. Igualmente, os sistemas computacionais (isto é, aqueles que têm software como um de seus principais elementos) também sofrem mudanças freqüentemente. (FILHO, 2011, p.46)

Áreas como engenharia e arquitetura de *software* definem estratégias necessárias para lidar com a complexidade (dificuldade de compressão sobre as regras estabelecidas), manutenibilidade (característica de um *software* ao que se refere à facilidade e economia de gastos em sua manutenção, sendo ao incluir novas funcionalidades ou realizar correções necessárias devido às mudanças estratégicas) e as frequentes mudanças enfrentadas no desenvolvimento de um *software*.

> O projeto de arquitetura é um processo criativo no qual você projeta uma organização de sistema para satisfazer aos requisitos funcionais e não funcionais de um sistema. Por ser um processo criativo, as atividades no âmbito do processo dependem do tipo de sistema a ser desenvolvido, a formação e experiência do arquiteto de sistema e os requisitos específicos para o sistema. Por isso, é útil pensar em projeto de arquitetura como uma série de decisões, em vez de uma sequência de atividades. (SOMERVILLE, 2012, p.105)

Segundo Mendes (2002), o frequente crescimento e complexidade durante o desenvolvimento de um *software*, torna-se essencial a utilização de níveis arquiteturais como complementação na gestão de um bom planejamento e desenvolvimento de *software*. O MVC é um padrão arquitetural utilizado na maioria dos *frameworks* de interface com usuário, Gamma (2000) ratifica, que sua abordagem é composta por três tipos de objetos. O modelo é o objeto de aplicação, a visão é a apresentação na tela e o controlador é o que define a maneira como a interface do usuário reage às entradas do mesmo. Mendes (2002) afirma, que um requisito não funcional é aquele que descreve não o que o *software* fará, mas como fará. A modularização conceitua a divisão do sistema em partes, sendo cada uma parte responsável por tomar decisões, realizar ações específicas e mesmo que desacopladas das demais no sistema, possuir relações internas. Promovendo assim melhores decisões acerca de suas regras e desenvolvimento, pois será possível ter uma visão mais abstrata de cada problema.

> Quando projetistas ou engenheiros de software se deparam com sistemas de grande porte e/ou complexos, eles geralmente dividem tais sistemas em partes menores, denominadas de módulos. Um sistema que é composto de um conjunto de módulos é chamado de modular. (MENDES, 2002, p.41)

Este trabalho apresenta de forma clara os conceitos presentes acerca de *software* e suas características, além disso, aprofunda os estudos e avalia a importância acerca da utilização da modularização aliada ao padrão arquitetural MVC como solução aos problemas de complexidade na construção de *software*.

Para tal foi elaborado um questionário qualitativo e compartilhado com os profissionais de TI que possuem experiência no desenvolvimento de sistemas de *software*,

para obter resultados acerca de suas experiências na utilização do MVC e da modularidade, compondo suas características e abordagens, no intuito de identificar os benefícios e pontos de importância de suas utilizações.

Dentre alguns dos resultados encontrados, é possível afirmar que a modularidade promove consideravelmente a redução da complexidade no desenvolvimento de uma aplicação. Além do mais, a padronização da comunicação entre as camadas da aplicação promovida pelo MVC, reduz a complexidade arquitetural e consequentemente promove uma melhor manutenção do *software*.

O presente trabalho foi dividido em 3 capítulos, sendo o primeiro compondo todo o corpo teórico necessário para o entendimento do mesmo, o segundo capítulo aborda como este trabalho foi realizado e o terceiro explora os resultados obtidos com base nos estudos e no questionário aplicado.

1 REFERENCIAL TEÓRICO

Neste capítulo são abordados os conteúdos científicos ao qual o trabalho foi embasado.

1.1 Engenharia de *Software*

Se entende por um *software* todo aquele programa criado por profissionais na área de desenvolvimento, capaz de ser executado, de se comunicar e dar ordens para a realização de tarefas específicas a uma máquina. Esta máquina por sua vez, sendo capaz de ler as informações, armazenar, processar e emitir de alguma forma as informações para ela transferidas.

> Software consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas. (PRESSMAN, 2011, p.32)

Segundo Sommerville (2012), a engenharia de *software* inclui técnicas que auxiliam desde a especificação do projeto, seu desenvolvimento prático e implantação, tendo assim o objetivo de auxiliar o desenvolvimento de uma forma mais profissional. Atualmente o *software* vai além de instruções para o computador interpretar e executar, compõe também, desde seu planejamento até sua execução.

> Essa é uma diferença importante entre desenvolvimento de software profissional e amador. Se você está escrevendo um programa para si mesmo, que ninguém mais usará, você não precisa se preocupar em escrever o manual do programa, documentar sua arquitetura etc. No entanto, se você está escrevendo um software que outras pessoas usarão e no qual outros engenheiros farão alterações, então você provavelmente deve fornecer informação adicional, assim como o código do programa. (SOMMERVILLE, 2012, p.3)

Desenvolver um *software* de qualidade é essencial para que o mesmo possa apresentar resultados satisfatórios a seção seguinte abordará alguns desafios enfrentados

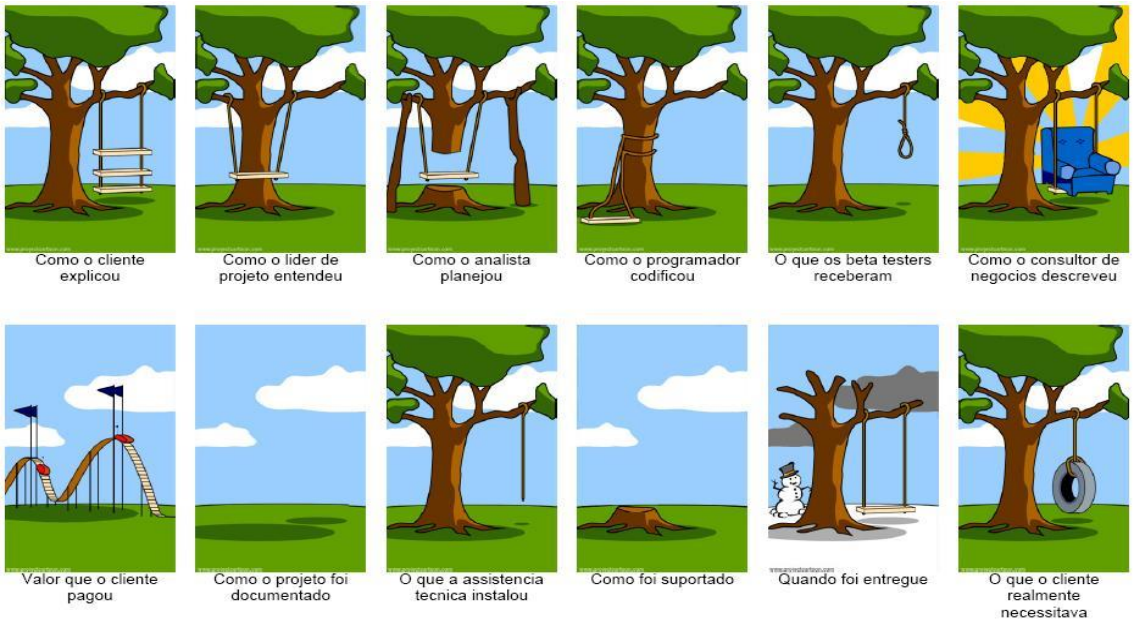
durante o desenvolvimento.

1.1.1 Dificuldades no desenvolvimento de *software*

Nos anos de 1970 quando a houve um rápido crescimento no desenvolvimento de *software*, a engenharia era praticamente inexistente, sendo assim, os recursos e técnicas para se desenvolver aplicações com custo baixo, com ferramentas eficientes e linguagens que atenderiam todas as necessidades também eram. Atualmente o cenário é outro, há diversas metodologias, ferramentas e padrões que facilitam e orientam o desenvolvimento, graças a engenharia de *software*.

A imagem abaixo ilustra as dificuldades atuais.

ILUSTRAÇÃO 1 – Importância da engenharia de *software*



Fonte: Ednaldo (2017)

Observa-se por meio da Ilustração 1, as dificuldades tanto para compreender o que o cliente necessita, quanto para repassar essas informações para os desenvolvedores. A princípio o cliente explica sua necessidade, os analistas projetam e desenvolvem algo totalmente diferente, sendo assim, por culpa da falta de comunicação e entendimento o serviço é cobrado normalmente e entregue apenas o suficiente para o cliente, ocorrendo

falhas desde o seu projeto.

Durante todo o processo de desenvolvimento de um *software*, os *stakeholders*, ou seja, aqueles que de certa forma participam do processo de desenvolvimento, encontram várias dificuldades e empecilhos para que o produto saia tudo como esperado. Segundo Sommerville (2012), os principais desafios na engenharia de um *software* é lidar com o aumento de diversidade, demandas pela diminuição do tempo para entrega e desenvolvimento de software confiável.

> No desenvolvimento de software, a qualidade de um projeto engloba o grau de atendimento às funções e características especificadas no modelo de requisitos. A qualidade de conformidade focaliza o grau em que a implementação segue o projeto e o sistema resultante atende suas necessidades e as metas de desempenho. (PRESSMAN, 2011, p.359)

Sendo assim, para se desenvolver um *software* de qualidade é necessário que este possua algumas características, tais como, facilidade de manutenção, estar em conformidade com o projeto, qualidade de desempenho fornecendo todas as funções e recursos especificados anteriormente e confiabilidade da entrega de recursos sem a ocorrência de falhas.

> A manutenção começa quase imediatamente. O software é liberado para os usuários finais, e em alguns dias, os relatos de bugs começam a chegar à organização de engenharia de software. Em algumas semanas, uma classe de usuários indica que o software deve ser mudado para se adaptar às necessidades especiais de seus ambientes. (PRESSMAN, 2011, p.663)

Para Pressman (2011), outro problema no processo de manutenção de um *software* é a mobilidade dos profissionais, é provável que profissionais ou equipes que anteriormente modificaram o sistema não estejam mais presentes à organização.

A seguinte seção se trata de padrões de projeto e como estes podem ser úteis para o desenvolvimento de um *software* de qualidade.

1.1.2 Padrões de projeto

Segundo Sommerville (2012), os padrões de projeto são formas de compartilhamento e reusabilidade do conhecimento sobre sistemas de *software*. Gamma

(2000) afirma que um padrão de projeto nomeia, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la útil para a criação de um projeto orientado a objetos reutilizável.

> Os padrões de projeto tornam mais fácil reutilizar projetos e arquiteturas bem-sucedidas. Expressar técnicas testadas e aprovadas as torna mais acessíveis para os desenvolvedores de novos sistemas. Os padrões de projeto ajudam a escolher alternativas de projeto que tornam um sistema reutilizável e a evitar alternativas que comprometam a reutilização. Os padrões de projeto podem melhorar a documentação e a manutenção de sistemas ao fornecer uma especificação explícita de interações de classes e objetos e o seu objetivo subjacente. Em suma, ajudam um projetista a obter mais rapidamente um projeto adequado. (GAMMA, 2000, p.18)

Pressman (2011) define que o intuito de cada padrão de projeto é fornecer uma descrição que permite a um projetista determinar se o padrão se aplica ou não ao trabalho em questão e se o padrão pode ou não ser reutilizado. As características específicas do sistema são definidas pelos padrões de arquitetura, por exemplo o padrão de distribuição, que em um ambiente distribuído padroniza a comunicação dos componentes do sistema.

> O problema da distribuição trata a maneira pela qual os sistemas ou componentes nos sistemas se comunicam entre si em um ambiente distribuído. São considerados dois subproblemas: (1) a maneira pela qual as entidades se conectam entre si e (2) a natureza da comunicação que ocorre. (PRESSMAN, 2011, p.327)

Segundo Pressman (2011), dentre alguns exemplos de padrões estruturais, tem-se o padrão *Adapter* que adapta uma interface de uma classe para uma que um cliente esperava. O padrão *Singleton* é um padrão criacionista onde restringe uma classe a apenas uma instância. Pressman (2011) também cita como conceito de projeto a separação por interesses, que sugere que qualquer problema complexo pode ser tratado mais facilmente ser for subdividido em trechos a serem resolvidos ou otimizados independentemente. É mais fácil solucionar dois problemas complexos menores separadamente do que solucionar um problema complexo maior.

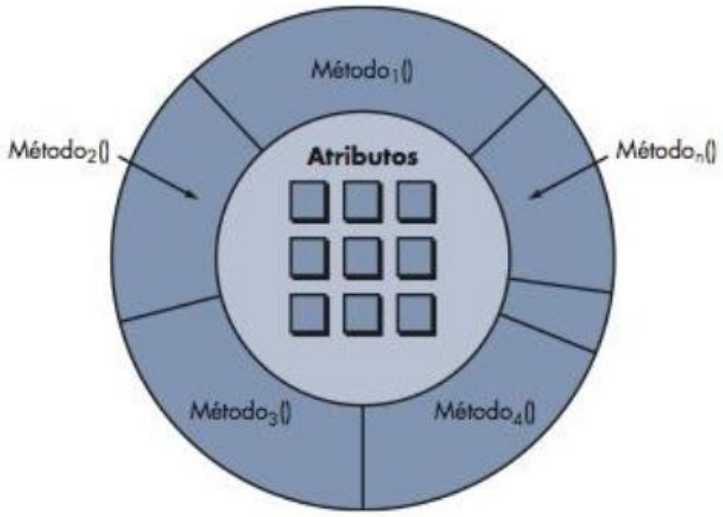
A seção seguinte aborda o paradigma orientação a objetos, compondo suas características e seus benefícios no desenvolvimento.

1.1.3 Orientação a objetos

A orientação a objetos é um paradigma da programação composta por classes, atributos, objetos e o relacionamento entre elas, segundo Loudon (2010), a ideia fundamental por trás da programação orientada a objetos (POO) é a de agruparmos os dados (chamados membros de dados na linguagem típica de POO) com operações que realizam ações com eles (chamadas métodos).

Pressman (2011) afirma que classe é um conceito que encapsula os dados e abstrações procedurais necessárias para descrever o conteúdo e comportamento de alguma entidade do mundo real. A instância desta classe é chamada de objeto.

ILUSTRAÇÃO 2 – Representação esquemática de uma classe



Fonte: Pressman (2011)

É possível observar na Ilustração 2 a representação de uma classe, contendo em seu centro os atributos que representam a mesma e os métodos que são utilizados para manipular estes atributos. Segundo Pressman (2011), o atributo pode assumir um valor definido por um domínio enumerado e cada uma das operações encapsuladas por um objeto proporciona uma representação de um dos comportamentos do objeto.

A seção seguinte tratará sobre os aspectos que tange a arquitetura de *software* e seus aspectos de relevância para a construção de um *software*.

1.1.4 Arquitetura de *software*

Durante todo o processo de planejamento e desenvolvimento de um *software* são definidos elementos de organização e decisão. Estes incidem nas formas de execução, linguagens e ferramentas que serão utilizadas.

Um planejamento arquitetural de um *software*, se inicia tendo como conhecimento todo o conjunto de requisitos funcionais e não funcionais do sistema a ser desenvolvido. Os requisitos funcionais são aqueles cujo o sistema será capaz de realizar, como por exemplo, uma funcionalidade de saque de dinheiro em um caixa eletrônico disponível para o usuário. Os requisitos não funcionais, são critérios de como será realizada determinada tarefa, como por exemplo, modularizar um sistema, permitindo se utilizar de mais camadas de separação de código no desenvolvimento.

> Requisitos funcionais. São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer. Requisitos não funcionais. São restrições aos serviços ou funções oferecidas pelo sistema. Incluem restrições de *timing*, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo. (SOMMERVILLE, 2012, p.59)

Sendo assim, a Arquitetura de *Software* é utilizada como uma forma de planejamento do funcionamento e configuração dos componentes existentes em um *software*.

> A arquitetura descreve quais componentes de alto nível compõem um software, bem como quais responsabilidades esses componentes assumem em relação a outros componentes no sistema. Também descreve a organização dos componentes tanto no nível conceitual quanto em detalhes, visto que cada componente pode ter sua própria estrutura de arquitetura. Por fim, a arquitetura define quais interfaces os componentes expõem aos demais e quais as interfaces e componentes que eles, por sua vez, usam. A arquitetura é criada com base em um conjunto de requisitos obtidos junto aos stakeholders (usuários, desenvolvedores, etc.). (LIMA, ADRIANA, CARNIELLO, 2014, p.1)

A visão arquitetural de um *software* além de apontar os componentes e ferramentas que serão utilizados durante o desenvolvimento de um *software*, possui grande importante também como forma de documentação do *software* a ser desenvolvido. Segundo Germoglio (2010), deixar com que a arquitetura de um *software* seja construída

durante o seu desenvolvimento, não é aconselhável ao que tange ao não aproveitamento total de seus benefícios. Porém com a documentação arquitetural bem construída, esta proporcionará uma forma de análise do projeto ao que tange a análise de seus requisitos, de seus componentes e uma integridade maior do caminho a ser seguido durante o processo de desenvolvimento.

A seção seguinte tratará sobre os aspectos acerca a modularização de *software* seus aspectos de relevância para a construção de um *software* utilizando o padrão arquitetural MVC.

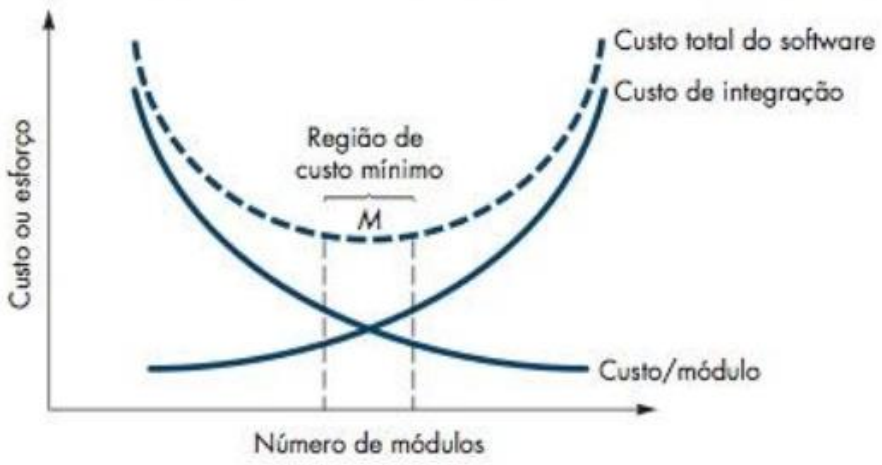
1.1.4.1 Modularização de componentes de *software*

Uma das características arquiteturais de um projeto de *software* é a modularização esta consiste em subdividir um sistema em subsistemas, ou seja, dividir em módulos, em que cada parte será responsável por suas determinadas funcionalidades e ações. Permitindo assim, abstrair as funcionalidades que serão compostas o sistema como um todo, além de propiciar com que os desenvolvedores e projetistas consigam focar parte por parte, para que no final as partes separadas se tornem um sistema completo.

> Na arquitetura, o pensamento é sempre componentizar ou modularizar o sistema, de tal forma que um sistema grande e complexo se transforme em um pequeno e coeso, preservando a interação entre seus elementos ou componentes. (CASSIMIRO, 2010, p.28)

Segundo Pressman (2011), um software monolítico é um grande programa composto por um único módulo e que geralmente não é facilmente entendido por um engenheiro de *software*. Para facilitar a compreensão e despesas desnecessárias para a sua construção, deve-se dividir um *software* em vários módulos.

ILUSTRAÇÃO 3 – Modularidade e custo do software



Fonte: Pressman (2011)

A Ilustração 3 apresenta quando deve-se aplicar a modularidade levando em conta o custo total do *software* e o número necessários de módulos presentes no mesmo. É possível observar que o custo de integração aumenta de acordo com a quantidade de módulos. Há também cenários em que o custo é alto com poucos módulos, portanto deve-se atentar para alcançar a região de custo mínimo, atendendo às necessidades do *software* a ser desenvolvido.

Segundo Pressman (2011), para engenharia de *software* componente é um bloco construtivo modular para software de computador, sendo um elemento funcional de um programa que incorpora a lógica do processamento, as estruturas de dados e uma interface que de alguma forma aciona e transfere dados.

Gamma (2000) afirma que quando uma aplicação possui classes ou componentes fortemente acoplados se tornam mais difíceis de se reutilizar isoladamente, sendo assim, em aplicações monolíticas não se pode alterar uma classe sem comprometer o comportamento das demais que se encontram atreladas. Para Loudon (2010), os componentes que não são exageradamente dependentes uns dos outros e quando interagem em modos definidos e com clareza apresentam acoplamento fraco.

Segundo Pressman (2011), para se decompor uma aplicação em módulos deve se levar em conta o princípio de encapsulamento das informações, sendo assim os módulos devem conter componentes e dados em comum, que consequentemente as descaracterizam das demais do sistema. Estas sendo desprezíveis aos demais módulos da aplicação. Promovendo assim a separação por interesses, que segundo Mendes (2002), é sempre quando um sistema é decomposto em vários módulos de modo que a arquitetura do sistema contenha mais de um componente.

Pressman (2011) também afirma que a modularidade é a manifestação mais comum da separação por interesses, de tal forma que seus componentes são separadamente especificados e endereçáveis. Sendo assim, em uma aplicação o uso das *namespaces* permite o agrupamento endereçável das classes e componentes existentes, possibilitando o acesso aos mesmos como se fossem diretórios em toda a aplicação.

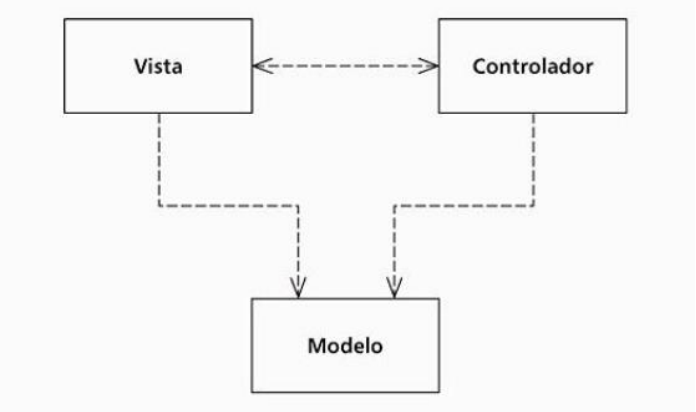
A seção a seguir trata-se acerca das definições do padrão arquitetural MVC, muito utilizado atualmente no desenvolvimento.

1.1.4.2 MVC (Model-View-Controller)

O padrão arquitetural MVC é muito utilizado em aplicações WEB, sendo responsável por dividir cada subsistema ou módulo em camadas de negócio (*Model*), controle do tráfego dos dados (*Controller*) e visualização da informação (*View*). Segundo Cassimiro (2010), cada camada é responsável por produzir independentemente uma lógica do negócio com a interface com o usuário.

> Separa a apresentação e a interação dos dados do sistema. O sistema é estruturado em três componentes lógicos que interagem entre si. O componente Modelo gerencia o sistema de dados e as operações associadas a esses dados. O componente Visão define e gerencia como os dados são apresentados ao usuário. 0O componente Controlador gerencia a interação do usuário (por exemplo, teclas, cliques do mouse etc.) e passa essas interações para a Visão e o Modelo. (SOMMERVILLE, 2012, p.109)

ILUSTRAÇÃO 4 – Arquitetura MVC



Fonte: Fowler (2006)

Observa-se na Ilustração 4 a disposição de cada componente existente no MVC, cada qual responsável pelas suas características. Um de seus benefícios é a possibilidade de substituição da camada de modelo já que esta não depende das demais. Segundo Fowler (2006), vista representa a exibição dos dados ao usuário, modelo é um objeto que representa a área de negócios, sendo parecidos com as tabelas de um banco de dados. Já o controlador envia os dados para a visualização a partir de uma entrada do usuário, como por exemplo, retornando alguma informação consultada pelo modelo no banco de dados.

Segundo Loudon (2010), objetos em sistemas orientados a objetos são naturalmente coesos, pois seus dados e métodos são agrupados em pacotes encapsulados que são definidos dentro de uma classe.

O reuso de *software* é uma das vantagens ao se utilizar o MVC, segundo Sommerville (2012), o reuso de software tornou-se a abordagem dominante para a construção de sistemas WEB. Quando construímos esses sistemas, pensamos em como podemos montá-los a partir de componentes e sistemas de software preexistentes.

A seção seguinte abordará as definições, utilização e o funcionamento de aplicações WEB.

1.3 Aplicações WEB

Desde a invenção da internet é possível a realização de troca de dados entre computadores, estes dados podem ser desde simples textos, até páginas interpretadas através do navegador de internet.

As aplicações WEB são sistemas desenvolvidos e interpretados através do navegador de internet e processados por um servidor WEB a partir do protocolo HTTP (protocolo utilizado na comunicação). Os servidores WEB por sua vez são responsáveis por interpretar os dados recebidos através do protocolo e enviar uma resposta, como por exemplo uma página de internet.

> Com o passar dos anos, as aplicações web evoluíram rapidamente de simples web sites cujo propósito era apenas navegação sobre a informação para verdadeiros sistemas de informação altamente complexos, repletos de dados e transações, voltados para a implementação de processos de negócio intra- e inter-organização. Diante deste quadro, a necessidade de um processo de software

> sistemático que ajude a gerenciar o ciclo de vida de tais aplicações surge naturalmente. (JACYNTHO, 2008, p.1)

Segundo Souza (2005), os principais atributos que distinguem o desenvolvimento WEB dos demais, são a crescente coleta de requisitos e a rápida mudança nas informações apresentadas. Consequentemente estas aplicações devem-se adaptar e atender à evolução a todo instante.

Estas aplicações podem ser constituídas em uma linguagem de programação sendo responsável pela camada lógica do *software*, como por exemplo, a linguagem PHP. Utiliza-se também de uma linguagem interpretada pelo navegador, como por exemplo, a linguagem HTML.

A seção seguinte tratará sobre os aspectos teóricos acerca da linguagem de programação PHP, sendo esta a mais utilizada no desenvolvimento de aplicações WEB.

1.4 Linguagem PHP

PHP (Hypertext Preprocessor) surgiu em 1995 sendo uma linguagem interpretada de código aberto e de uso geral. Entre suas características, está incluída sua disponibilidade de código aberto (open source), este garante que todo seu código está disponível para modificações ou estudos, sendo assim sua utilização é gratuita. Foi criado para se executar códigos de programação no servidor WEB.

> O PHP é uma linguagem de criação de *scripts* do lado do servidor que foi projetada especificamente para *WEB*. Dentro de uma página HTML, você pode embutir código de PHP que será executado toda vez que a página for visitada. O código de PHP é interpretado no servidor *WEB* e gera HTML ou outra saída que o visitante verá. (WHELLING, 2005, p.26).

Segundo Converse (2004), PHP é uma linguagem de script ao lado do servidor, mas que também pode ser incorporada com HTML, foi originalmente chamado de Personal Home Page Tools (Ferramentas de Página Pessoal), porém seu nome atual foi escolhido pelo voto da comunidade.

A seção seguinte tratará sobre os aspectos teóricos acerca da linguagem de modelagem UML necessários para a exposição deste trabalho.

1.4.1 *Frameworks*

Segundo Minetto (2007), *framework* de desenvolvimento pode ser definido como uma “base” que se pode desenvolver algo maior ou mais específico. É uma coleção de códigos fonte, classes, funções, técnicas e metodologias que facilitam a construção de novas aplicações. Uma de suas vantagens de utilização é possuir padrões de projetos já testados e validados por toda a comunidade de desenvolvimento.

Dentre aos mais utilizados para PHP segundo Coders Eye (2017), apresenta-se o Laravel *Framework*. Este atualmente na versão 5.5 utiliza a arquitetura MVC e apresenta características de modularidade em sua utilização, sintaxe simples e coesa.

Os exemplos a seguir representam os componentes e a disposição das camadas do padrão arquitetural MVC no módulo denominado Library. Nesta abordagem é possível ter uma visão clara da estrutura arquitetural dos elementos existentes.

ILUSTRAÇÃO 5 – Author*Controller*

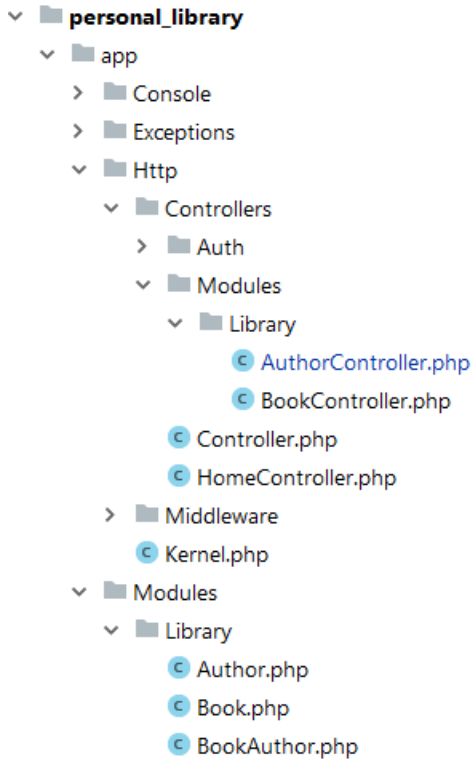
```php
<?php

namespace App\Http\Controllers\Modules\Library;

use ...

class AuthorController extends Controller
{
    public function index(){...}

    public function create(){...}

    public function edit(){...}

    public function delete(){...}
}
```

Fonte: Próprio autor (2017)

Na Ilustração 5 é possível observar a classe AuthorController e quais as possíveis interações com o usuário através das funções declaradas, um exemplo é através da função *create*, que receberá dados enviados pelo usuário através da camada de *view*, estes são

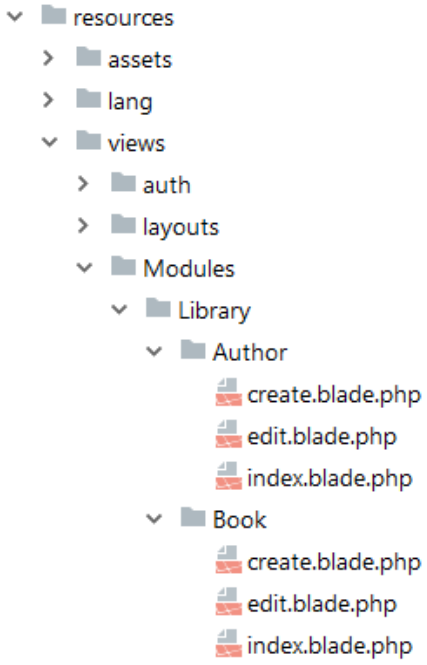
enviados pelo modelo ao banco de dados e retorna ao usuário o resultado se foi possível ou não criar o cadastro de um autor.

ILUSTRAÇÃO 6 – Estrutura dos modelos e controladores



Fonte: Próprio autor (2017)

ILUSTRAÇÃO 7 – Estrutura das visões



Fonte: Próprio autor (2017)

Através das Ilustrações 6 e 7 é possível observar a estrutura da aplicação modular no Laravel *Framework*, embora os arquivos não estejam fisicamente pertos, logicamente eles estão interligados, graças às *namespaces* (gráfico 6). Em toda a aplicação se é possível compreender onde exatamente um determinado arquivo pertence e quais interações o mesmo necessita ou está fazendo.

A seção seguinte tratará sobre os aspectos acerca das aplicações WEB, sendo este um tema muito abordado no desenvolvimento *framework* descrito acima.

1.5 Linguagem UML

UML (Linguagem de Modelagem Unificada) é uma proposta de um padrão de modelagem de *software* que tem por objetivo amplificar a visualização dos aspectos arquiteturais e estruturais de um *software*, especificar requisitos e estratégias utilizadas aos mesmos, exemplificar a construção de um *software* e documentar suas funcionalidades.

> A modelagem de software normalmente implica a construção de modelos gráficos que simbolizam os artefatos dos componentes de software utilizados e os seus inter-relacionamentos. Uma forma comum de modelagem de programas orientados a objeto é através da linguagem unificada UML. (VARGAS, p.1)

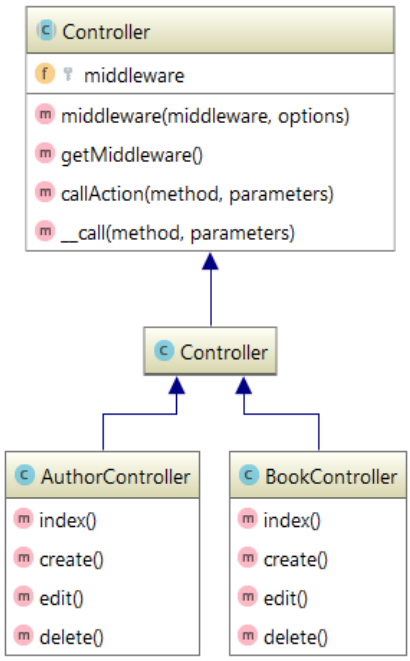
A linguagem UML utiliza-se de diagramas (representação gráfica) dos componentes existentes e suas relações no sistema. São divididos entre estáticos e dinâmicos. Estáticos são aqueles compostos por classes, objetos e seus componentes, já os dinâmicos são casos de uso, transições entre estados, atividade e colaboração.

Segundo Booch (2000), a UML fornece um repertório semântico, contendo todas as partes de todos os modelos de determinado sistema, cada parte relacionada às demais de uma forma consistente. Fornece também visões sobre o comportamento do projeto como resultado final e implementação do mesmo.

Os exemplos de diagramas estáticos a seguir foram gerados utilizando o ambiente de desenvolvimento integrado (IDE) PhpStorm e representam os componentes do módulo denominado Library, além da disposição das camadas do padrão arquitetural MVC. Nesta abordagem é possível ter uma visão clara da disposição dos elementos ao que tange suas

características arquiteturais.

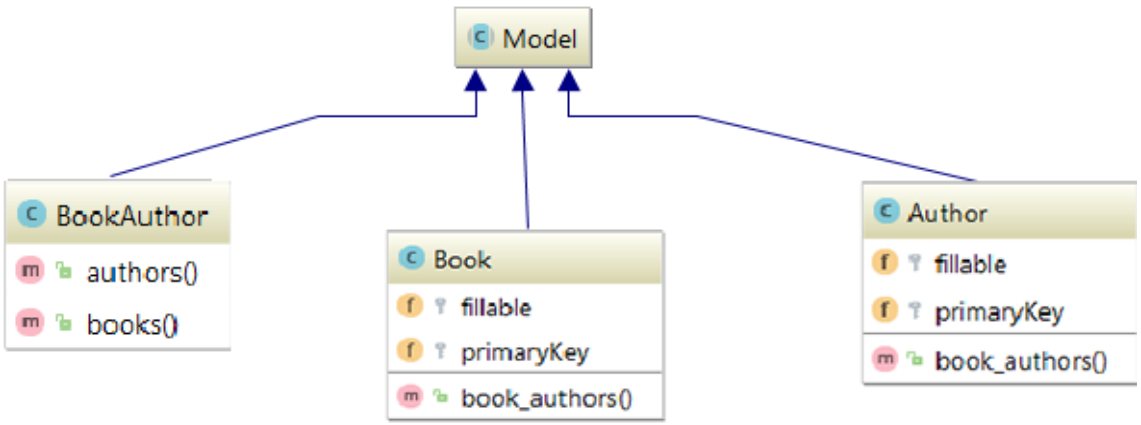
ILUSTRAÇÃO 8 – *Controller*s



Fonte: Próprio autor (2017)

Observa-se na Ilustração 8 a divisão das classes denominadas controladores no módulo de Library, entre Author e Book. Ambos possuem dependências acerca de uma classe abstrata chamada Controller e encapsula seus dados através de seus métodos, sendo assim tratando os dados de suas respectivas responsabilidades.

ILUSTRAÇÃO 9 – *Models*



Fonte: Próprio autor (2017)

A Ilustração 9 demonstra o relacionamento dos *models* da aplicação, cada uma

representando uma tabela no banco de dados, sendo assim, as tabelas de *book* e *author* foram encapsuladas para somente o tratamento dos dados de suas respectivas ações. Porém assim como no banco de dados, há um outro *model* responsável pela interligação entre as tabelas de *book* e *author*.

Segundo Fowler (2006), um modelo de domínio simples é uma camada que modela a área de negócios, e se parece com um projeto de banco de dados, tendo a maior parte das vezes, um objeto de domínio para cada tabela do banco de dados.

2 METODOLOGIA

O presente trabalho tem como objetivo avaliar, validar e identificar os benefícios da utilização de componentes modulares e do padrão arquitetural MVC, como solução à complexidade e mutabilidade no desenvolvimento de um *software*. Dessa forma, foram realizadas pesquisas bibliográficas na área de Engenharia de *Software*, bem como sua subárea de estudo, Arquitetura de *Software*, no intuito de fixar as definições e técnicas que compõem este trabalho científico.

Este trabalho foi dividido em duas etapas, a primeira voltada para a modularização, este que pode ser definido como um requisito não funcional no desenvolvimento de um *software*. Sua característica é permitir que seus componentes sejam subdivididos em pedaços encapsulados e fracamente acoplados. A segunda etapa é voltada para o estudo do padrão de projeto arquitetural MVC, que visa uma melhor eficiência no desenvolvimento e manutenção de um *software*.

Durante o desenvolvimento deste trabalho foi constatada a necessidade da elaboração de um questionário de pesquisa qualitativa, no intuito da obtenção de resultados reais baseados na experiência e realidade dos profissionais de TI que utilizam o padrão e o requisito acima descritos.

Dessa forma o objetivo é confrontar os resultados obtidos através do questionário acima descrito, com a parte teórica por parte das definições e técnicas e os exemplos práticos desenvolvidos neste estudo.

2.1 Público-alvo do estudo

O público-alvo deste trabalho e do questionário que se encontra no APÊNDICE 1, são os profissionais de TI que possuem experiência na área de desenvolvimento de software, e que tenham já desenvolvido *softwares* com componentes modulares e utilizando-se do padrão MVC.

2.2 Elaboração do questionário de pesquisa

O questionário de pesquisa utilizado para a coleta de dados neste trabalho acadêmico, foi elaborado utilizando-se de 38 perguntas, dentre elas possuem 24 perguntas específicas às técnicas utilizadas no padrão MVC e na modularidade dos componentes de *software*. Relacionadas ao perfil dos entrevistados, foram elaboradas 10 perguntas e 4 relacionadas à experiência profissional nas empresas daqueles respondentes que afirmaram nunca ter desenvolvido *softwares* modulares com MVC.

Estas perguntas foram divididas em 4 seções e baseadas nos autores Gamma (2000), Mendes (2002), Fowler (2006), Loudon (2010) e Sommerville (2012):

- A primeira seção trata-se do perfil do participante, contendo perguntas de informações pessoais profissionais e acadêmicas. Possibilitando assim identificar e melhor classificar o participante conforme suas respostas;
- Na segunda seção, as perguntas têm por objetivo obter o conhecimento do participante no desenvolvimento de *software* modular e identificar o quanto a adoção das técnicas de modularidade em um software promovem a redução da complexidade no desenvolvimento;
- A terceira seção tem por objetivo identificar o conhecimento do participante das técnicas e características do padrão MVC e obter respostas sobre o quanto a aplicação destas técnicas promovem um desenvolvimento de software mais eficiente;
- A quarta seção foi destinada aos participantes que nunca desenvolveram *softwares* modulares utilizando o MVC. Nesta é abordado de acordo com a experiência profissional se nas empresas na qual trabalhou utiliza as técnicas descritas.

Além disso, foi abordado o nível de conhecimento e experiência do participante do questionário acerca das técnicas descritas, no intuito de se obter com mais veracidade os resultados. Foi abordado também o quanto a utilização dessas técnicas promove um desenvolvimento de *software* e sua manutenção mais eficientes, reduzindo assim, sua complexidade.

2.4 Coleta dos dados

O questionário qualitativo foi desenvolvido utilizando a plataforma online Google Forms (Formulário Google), esta foi escolhida pois proporciona a fácil criação e compartilhamento, além do gerenciamento das respostas obtidas.

O questionário elaborado foi compartilhado em grupos específicos de desenvolvimento de *softwares* nas redes sociais, tais como, Facebook e Google Plus, de diferentes linguagens que podem se utilizar do padrão arquitetural e requisito não funcional descritos, cujos membros são desenvolvedores de *softwares*, no intuito de coletar o máximo de informações possíveis atingindo o público específico. Foram coletadas no total de 71 respostas, e ficou disponível entre os dias 19/09/2017 a 01/11/2017.

2.5 Tratamento dos dados

As respostas obtidas pelo questionário foram armazenadas automaticamente pela ferramenta do Google e disponibilizadas tanto para análise, quanto para exportar no computador em formato “xlsx” aceito pela ferramenta Excel da Microsoft. Estas foram extraídas e tratadas de acordo com o perfil e a experiência de desenvolvimento do participante, através da ferramenta Microsoft Excel no intuito de promover uma melhor apresentação através da criação de gráficos e tabelas.

Os resultados das análises e dos tratamentos dos dados obtidos através do questionário são apresentados na próxima seção.

3 ANÁLISE DOS RESULTADOS

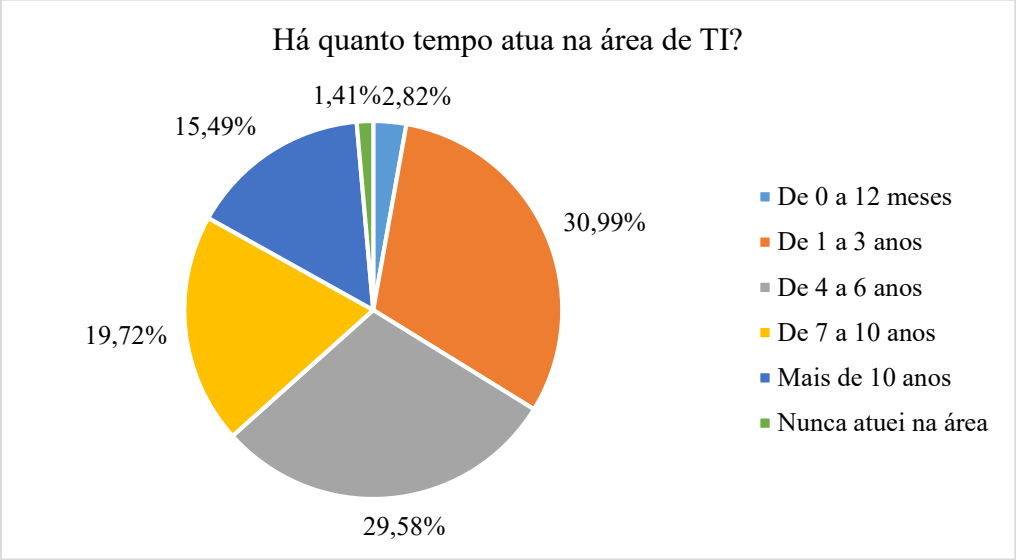
Nesta seção serão analisados e apresentados os resultados referentes as respostas dos 71 participantes do questionário aplicado junto deste trabalho. Nos quais 54 dos respondentes afirmaram ter trabalhado no desenvolvimento de *softwares* modulares, já 55 dos respondentes afirmaram ter trabalhado no desenvolvimento de *softwares* utilizando a arquitetura MVC. Sendo assim, é possível observar que a maioria dos respondentes possuem experiência em ambas as abordagens. Os demais 16 respondentes foram os que afirmaram não possuir experiência e conhecimento necessário para responder as questões características de modularidade e do MVC, sendo questionados somente acerca da empresa na qual trabalham.

3.1 Perfil dos entrevistados

Dentre o perfil dos participantes desta pesquisa, constatou-se que 50,70% dos respondentes possuem entre 16 a 25 anos, 40,85% possuem entre 26 a 35 anos, 5,63% possuem entre 36 a 45 anos e 2,82% possuem entre 46 a 55 anos.

A formação acadêmica dos participantes deste questionário em sua maioria foi composta por 36,62% de Ensino Superior Completo, 32,39% Ensino Superior Incompleto, 21,13% Pós-Graduação, 4,23% composto por Mestrado e Técnico e 1,41% Doutorado.

GRÁFICO 1 - Há quanto tempo atua na área de TI?



Fonte: Próprio autor (2017)

De acordo com o Gráfico 1, referente ao tempo de atuação na área de TI dos participantes, foi constatado um balanço satisfatório de respostas possuindo a maioria delas de profissionais experientes atuantes a mais de 4 anos e outras não muito experientes. Sendo 30,99% de 1 a 3 anos, 29,58% compostas por profissionais atuantes de 4 a 6 anos, 19,72% de 7 a 10 anos, 15,49% mais de 10 anos de atuação na área de TI, 2,82% até 1 ano e apenas 1,41% nunca atuou na área de TI.

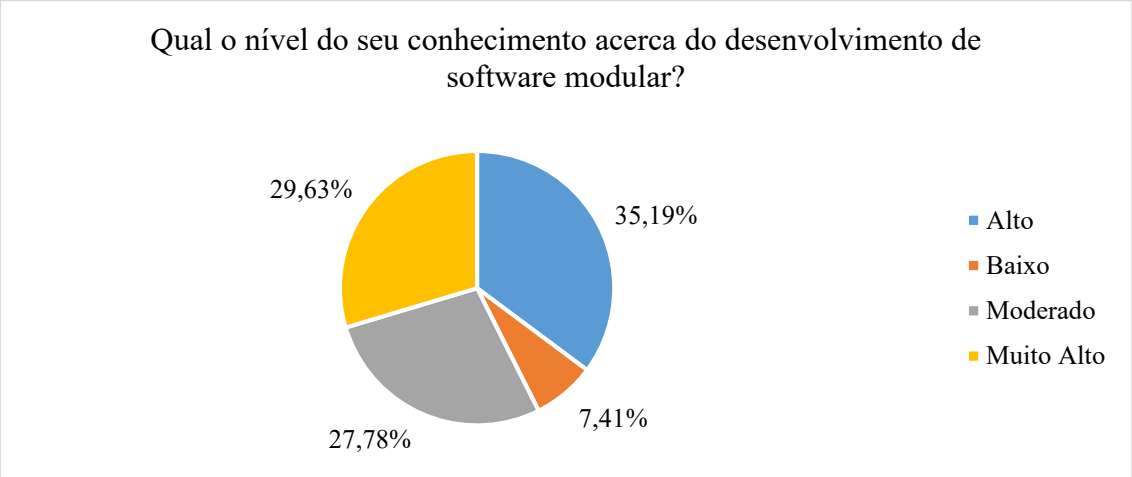
Dentre os cargos listados no questionário, os mais atuantes são 30,99% Desenvolvedores Web, 26,76% Analistas de Sistemas, 7,04% Arquitetos de *Softwares* e Gerentes de Projetos e 5,63% Desenvolvedores Mobile. Os demais respondentes estão voltados para outras atuações como, Desenvolvedor Desktop e Engenheiro Backend.

3.2 Resultados da utilização de *softwares* modularizados

Nesta seção serão apresentados os resultados acerca das 54 respostas, estas são referentes aos participantes que afirmaram possuir experiência no desenvolvimento de softwares modulares. Porém foi desconsiderada uma resposta nas quais o participante afirma possuir nenhum conhecimento, sendo assim, serão consideradas apenas as respostas nas quais os mesmos afirmaram possuir conhecimento necessário para

responder as demais perguntas.

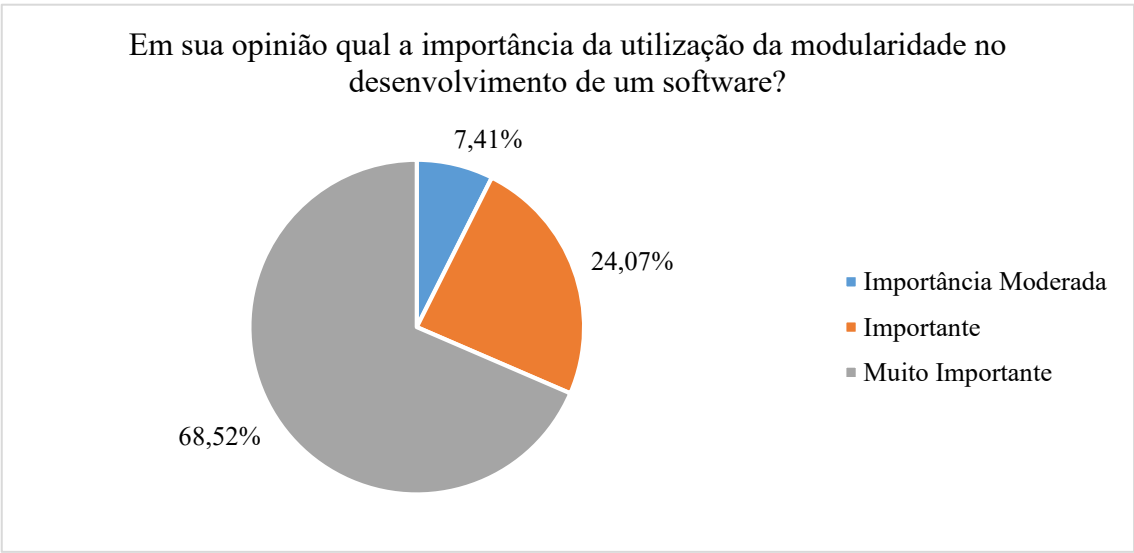
GRÁFICO 2 - Qual o nível do seu conhecimento acerca do desenvolvimento de *software* modular?



Fonte: Próprio autor (2017)

De acordo com o Gráfico 2, é possível notar que 35,19% dos respondentes afirmaram possuir um conhecimento alto, 29,63% afirmaram possuir um conhecimento muito alto, 27,78% possuem conhecimento moderado e 7,41% possuem conhecimento baixo. Sendo assim, é possível notar que a maioria dos respondentes informaram possuir os conhecimentos necessários para serem aptos a responder as questões técnicas acerca de *softwares* modulares e suas características.

GRÁFICO 3 - Em sua opinião qual a importância da utilização da modularidade no desenvolvimento de um *software*?



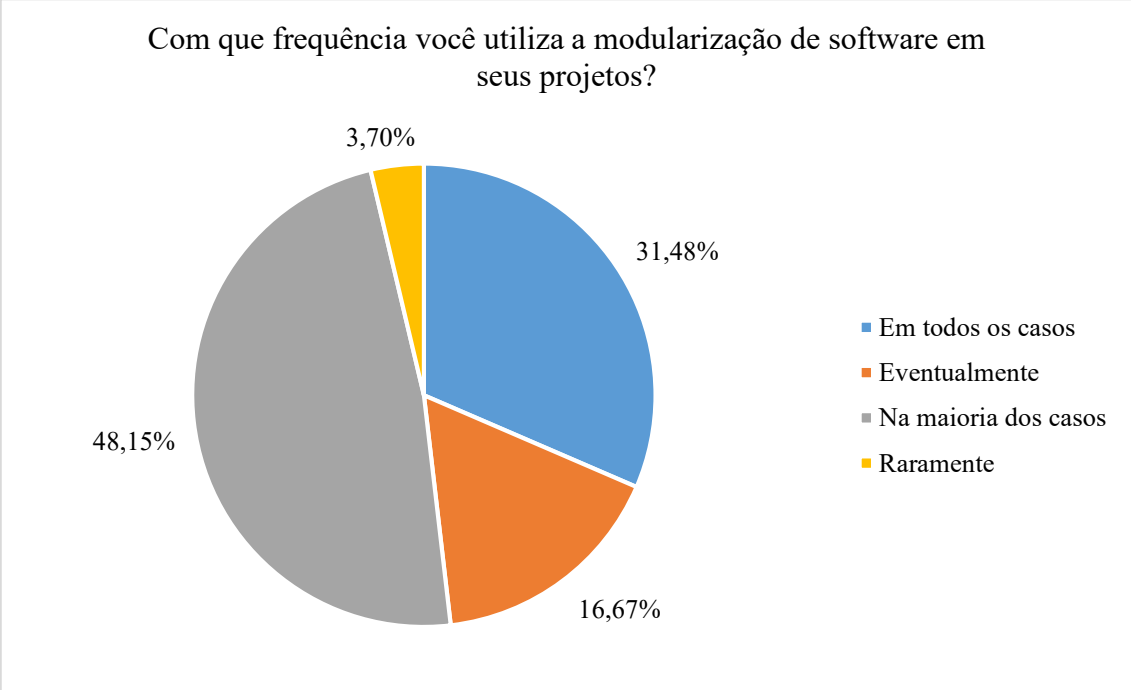
Fonte: Próprio autor (2017)

De acordo com o Gráfico 3, é possível notar que ampla maioria considerou importante a utilização da modularidade no desenvolvimento de um *software*. Dentre os 54 participantes do questionário qualitativo, 68,52% afirmaram ser muito importante, importante 24,07% e apenas 7,41% afirmaram ser de importância moderada.

Para Sommerville (2012), os requisitos de sistema não apenas especificam os serviços ou as características necessárias ao sistema, mas também a funcionalidade necessária para garantir que esses serviços/características sejam entregues corretamente.

Sendo assim, é possível inferir que há uma grande quantidade de desenvolvedores de *software*, nos quais reconhecem a importância da modularidade de tal forma que consideram e utilizariam a modularidade em seus projetos atuais e futuros.

GRÁFICO 4 - Com que frequência você utiliza a modularização de *software* em seus projetos?



Fonte: Próprio autor (2017)

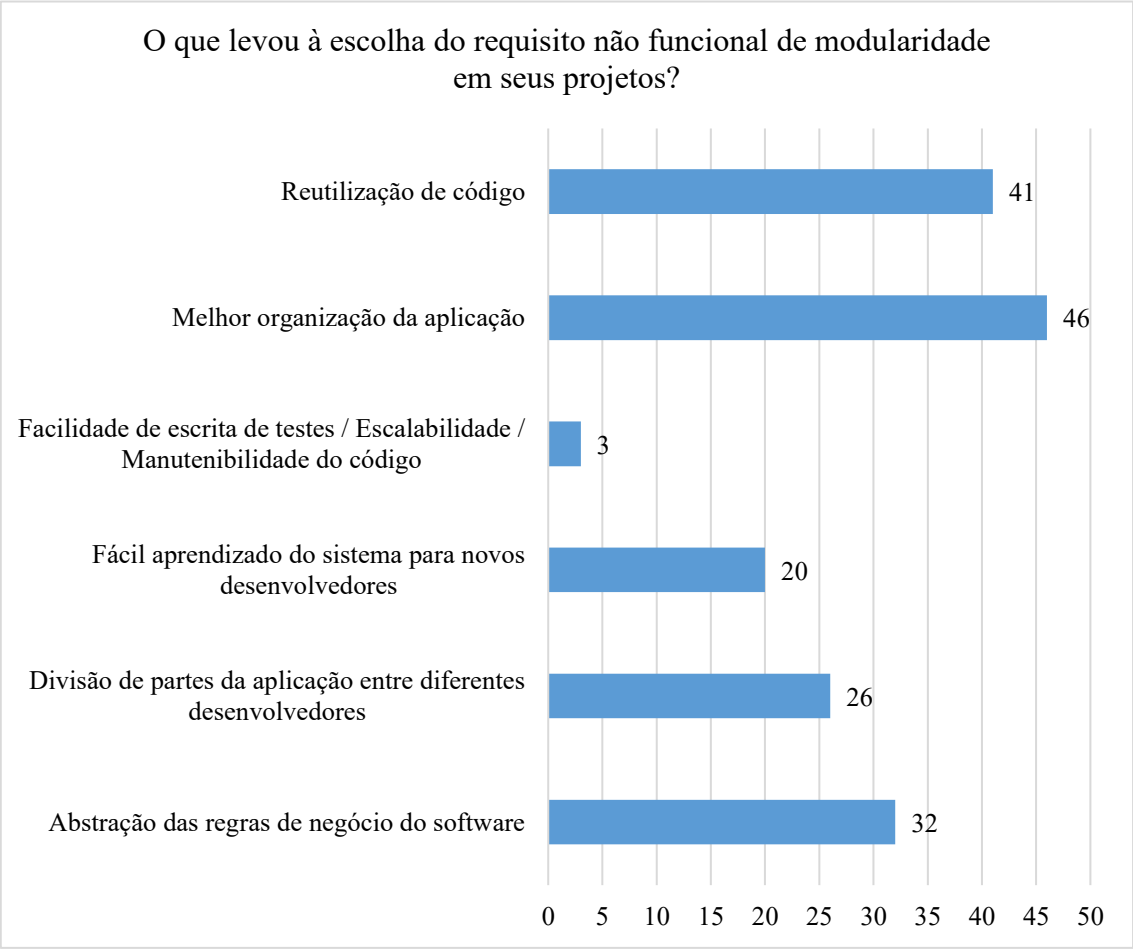
De acordo com o Gráfico 4, é possível observar que dentre os participantes do questionário, 48,15% afirmaram utilizar o requisito não funcional de modularização na maioria dos casos em seus projetos, 31,48% utilizam em todos os casos, 16,67% eventualmente utilizam e apenas 3,70% raramente utilizam apesar de reconhecer sua importância.

Para Sommerville (2012), projetar e construir um *software* elegante que resolva o problema errado não atende às necessidades de ninguém. Para Pressman (2011),

modularizamos um projeto de modo que o desenvolvimento possa ser planejado mais facilmente; incrementos de *software* possam ser definidos e entregues e as mudanças possam ocorrer mais facilmente.

Sendo assim, é possível concluir que atualmente a modularidade continua sendo muito utilizada no desenvolvimento de sistemas, visto que seus benefícios são grande valor quando tratado a um grande projeto de *software*, seja nas empresas ou em projetos pessoais.

GRÁFICO 5 - O que levou à escolha do requisito não funcional de modularidade em seus projetos?



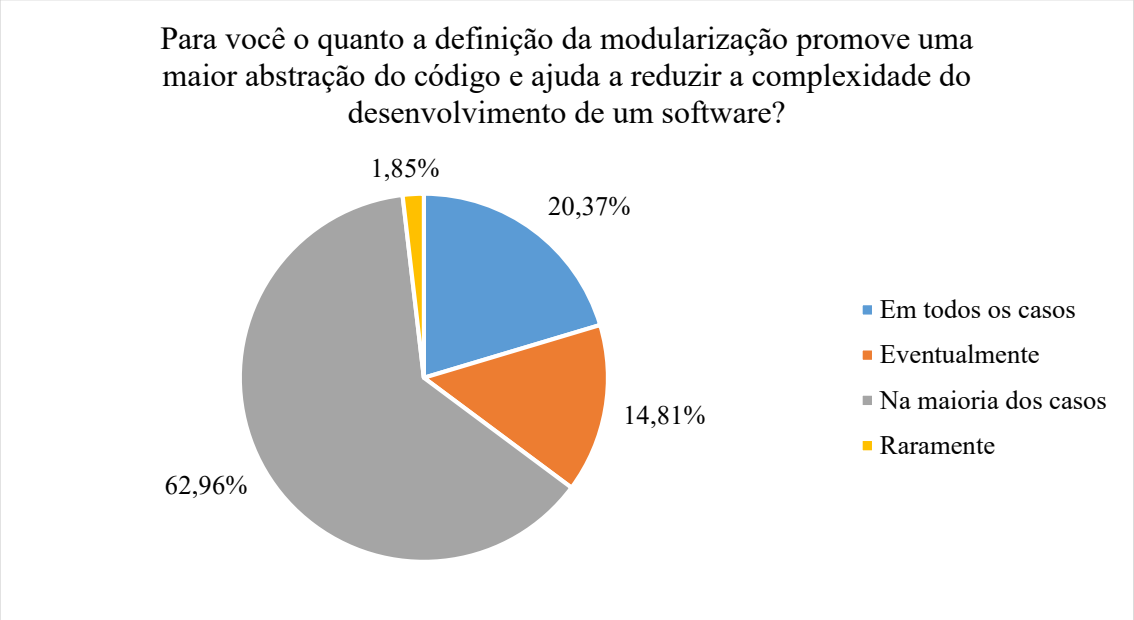
Fonte: Próprio autor (2017)

De acordo com o Gráfico 5, há vários motivos que levaram dentre os 54 participantes a modularizar seus *softwares* ou modularizar projetos que já participaram. O principal foi que este requisito promove uma melhor organização da aplicação, tendo sido escolhido 46 vezes. Outro motivo escolhido 41 vezes foi que este promove uma melhor reutilização de código, 32 vezes disseram que escolheram modular o *software*, porque permite abstrair as regras de negócio. 26 vezes escolheram, pois este permite a

divisão de partes da aplicação entre diferentes desenvolvedores. O fácil aprendizado do sistema para novos desenvolvedores também foi um fator relevante para 20 respondentes.

Os demais respondentes optaram por escolher modular a aplicação de *software*, pois buscavam a manutenibilidade de código, escalabilidade e até mesmo uma maior facilidade na escrita de testes.

GRÁFICO 6 - Para você o quanto a definição da modularização promove uma maior abstração do código e ajuda a reduzir a complexidade do desenvolvimento de um *software*?



Fonte: Próprio autor (2017)

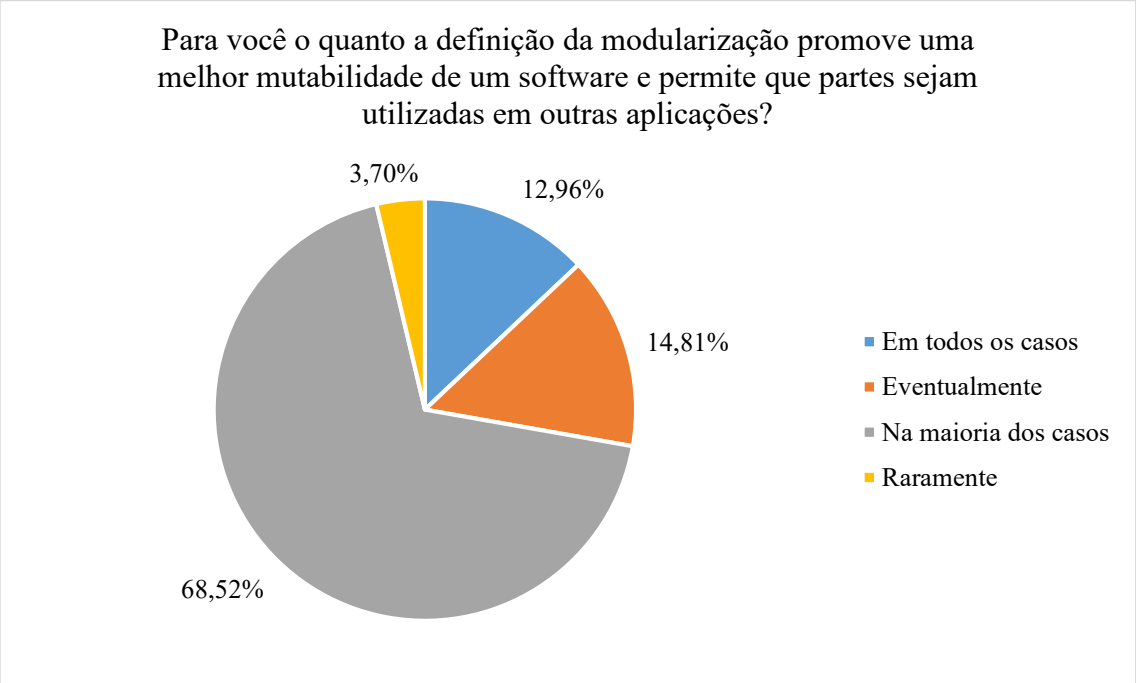
Como observado no Gráfico 6, 62,96% dos respondentes afirmaram que na maioria dos casos a modularização de um *software* promove uma maior abstração, e consequentemente a redução na complexidade do desenvolvimento da aplicação. De acordo com 20,37% em todos os casos a modularização foi positiva para a redução da complexidade no desenvolvimento. 14,81% afirmaram que eventualmente este fato se ocorre e 1,85% afirmaram raramente promover a redução.

> Ao considerar uma solução modular para qualquer problema, muitos níveis de abstração podem se apresentar. No nível de abstração muito alto, uma solução é expressa em termos abrangentes usando a linguagem do domínio do problema. Em níveis de abstração mais baixos, uma descrição mais detalhada da solução é fornecida. A terminologia do domínio do problema é associada à terminologia de implementação para definir uma solução. Por fim, no nível de abstração mais baixo, a solução técnica do software é expressa de maneira que pode ser diretamente implementada. (PRESSMAN, 2011 p.212)

De acordo com Mendes (2002), a abstração é o processo de ocultação de detalhes

que não devem ser observados quando se trabalha com um módulo de um ponto externo à sua implementação. Sendo assim, o desenvolvedor apenas precisará se concentrar acerca do entendimento nas regras do sistema em apenas uma parte da aplicação, podendo abstrair os demais módulos e suas regras.

GRÁFICO 7 - Para você o quanto a definição da modularização promove uma melhor mutabilidade de um *software* e permite que partes sejam utilizadas em outras aplicações?



Fonte: Próprio autor (2017)

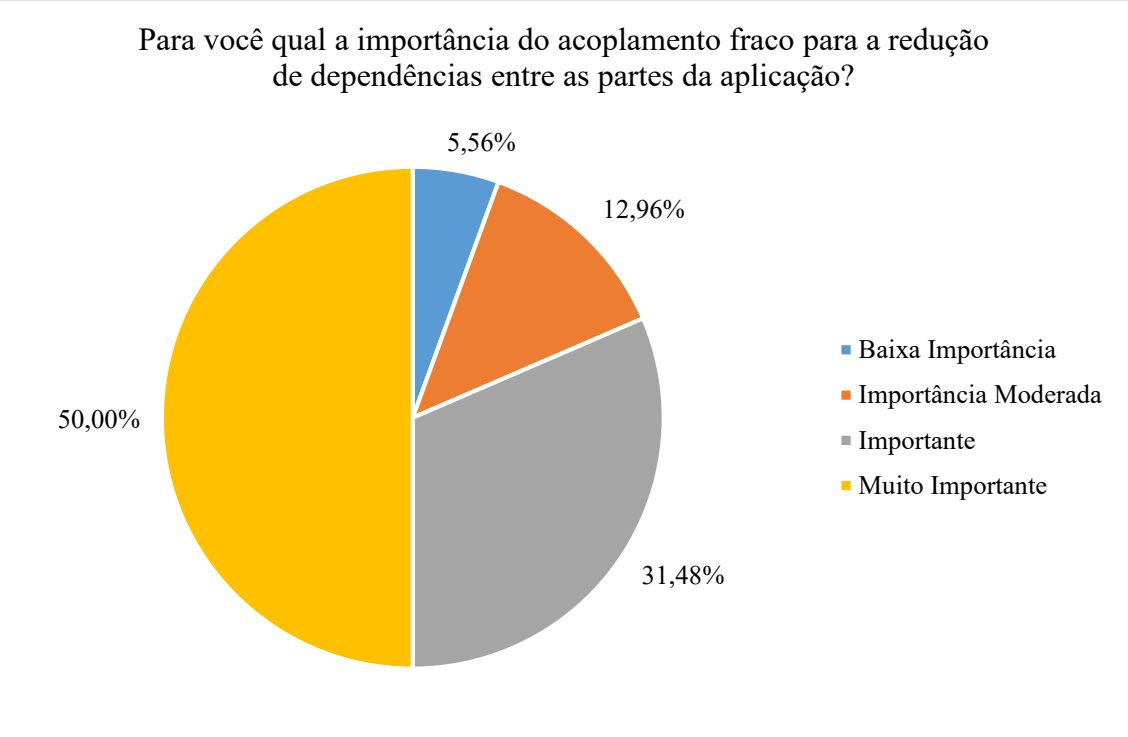
Segundo o Gráfico 7, pode-se notar que 68,52% dos respondentes afirmaram que na maioria dos casos a modularidade promove a evolução contínua do *software* e sua escalabilidade. Já para 14,81% eventualmente a modularidade promove uma mudança no *software* de forma mais flexível e abstrata, em todos os casos compõe um total de 12,96% de participantes. Porém somente 3,70% afirmaram, de acordo com suas experiências, que raramente a modularidade foi positiva para proporcionar um *software* mais preparado para aceitar mudanças.

> [...] a natureza evolutiva do software é tal que requer que o engenheiro de software muitas vezes reveja projetos anteriores a fim de modifica-los, corrigindo algum defeito ou acrescentando novas funcionalidades. Dessa forma, quando temos necessidade de manutenção, a modularidade ajuda a confinar a fonte de problema a um único componente. (MENDES, p.41)

Portanto Mendes (2002) afirma que a modularidade de um sistema, isto é,

entender cada parte de um sistema, separadamente, facilita modificações que se façam necessárias ao sistema.

GRÁFICO 8 - Para você qual a importância do acoplamento fraco para a redução de dependências entre as partes da aplicação?

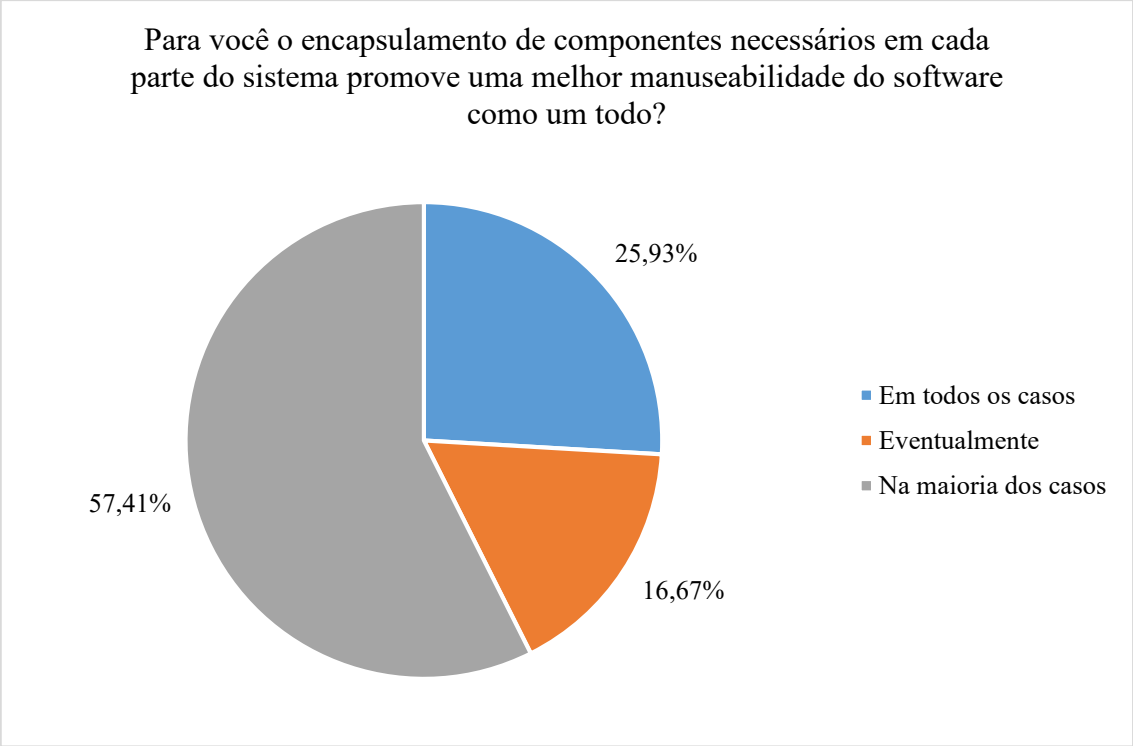


Fonte: Próprio autor (2017)

Pode-se observar no Gráfico 8 que 50% dos respondentes afirmaram que o fraco acoplamento da aplicação, ou seja, a fraca dependência dos componentes pertencentes à aplicação desenvolvida, é muito importante para a redução da dependência entre as partes, ou seja, os módulos da mesma. Já para 31,48% afirmaram ser apenas importante, para 12,96% possui importância moderada e para 5,56% baixa importância.

As respostas obtidas condizem com a afirmação de Mendes (2002), ele afirma que a separação por interesses presente na modularidade, é uma prática utilizada para sobrepujar dificuldades encontradas e lidar com dificuldades inerentes no desenvolvimento de sistemas de *software*, onde um sistema é decomposto por vários módulos e componentes fracamente acoplados.

GRÁFICO 9 - Para você o encapsulamento de componentes necessários em cada parte do sistema promove uma melhor manuseabilidade do *software* como um todo?



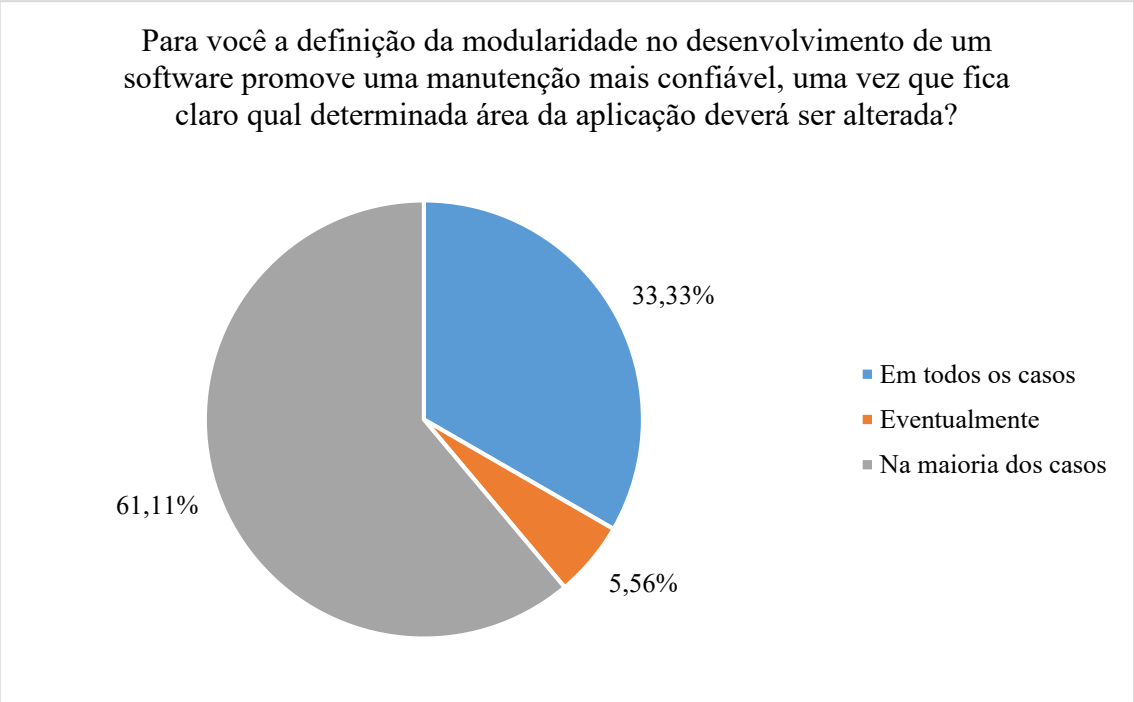
Fonte: Próprio autor (2017)

O Gráfico 9 aborda se o encapsulamento dos componentes presentes em cada módulo da aplicação, promove um *software* de melhor qualidade e de fácil modificação. De acordo com 57,41% dos respondentes na maioria dos casos o encapsulamento promove uma melhor manuseabilidade do *software*, para 25,93% em todos os casos e para 16,67% eventualmente promove.

Para Pressman (2011), como a maioria dos detalhes procedurais são ocultos para outras partes do *software*, o uso do encapsulamento em projetos modulares fornece maiores benefícios durante os testes e durante a manutenção do *software*, pois promove a modificação de um módulo sem que esta implique em outros módulos.

Sendo assim os resultados exibidos no gráfico 9, condizem com os benefícios promovidos pelo encapsulamento dos módulos de uma aplicação.

GRÁFICO 10 - Para você a definição da modularidade no desenvolvimento de um *software* promove uma manutenção mais confiável, uma vez que fica claro qual determinada área da aplicação deverá ser alterada?



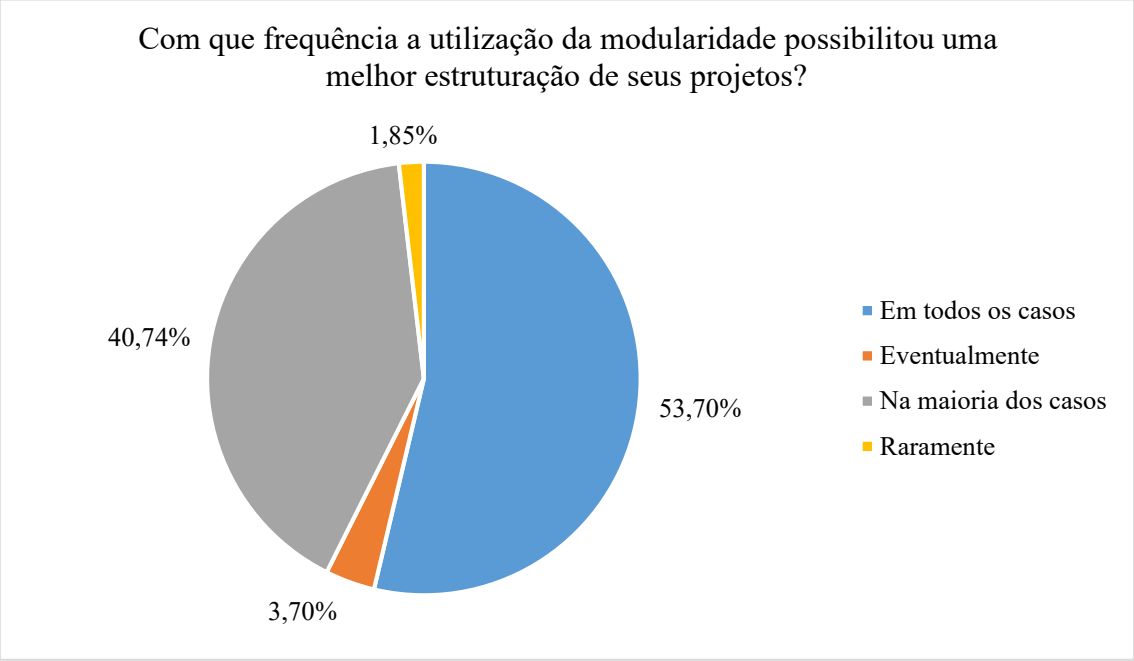
Fonte: Próprio autor (2017)

No Gráfico 10 foi abordado se a adoção da modularidade no desenvolvimento de um software, é importante de tal forma, pois fornece uma melhor organização arquitetural dos componentes da aplicação, possibilitando assim, uma maior facilidade para os desenvolvedores darem manutenção na mesma. De acordo com 61,11% dos respondentes, na maioria dos casos a modularidade promove uma melhor manutenção de software, 33,33% afirmaram em os casos, já para 5,56% eventualmente promove.

Para Loudon (2010), com a modularidade fica clara qual área da aplicação deve ser alterada e como as alterações devem ser feitas. Módulos de fácil manutenção são mais confiáveis por reduzirem o risco de efeitos colaterais quando se realizam as mudanças.

Portanto os resultados obtidos através das respostas dos participantes, onde afirmaram a modularidade sendo importante para a melhor manutenção, condizem com os benefícios e as características da modularidade.

GRÁFICO 11 - Com que frequência a utilização da modularidade possibilitou uma melhor estruturação de seus projetos?

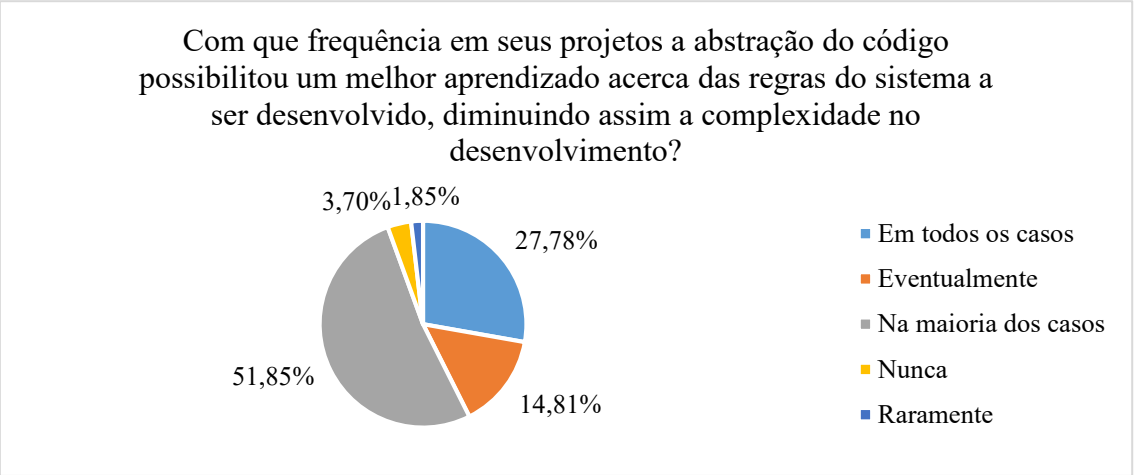


Fonte: Próprio autor (2017)

No Gráfico 11 é exibido o resultado referente às respostas obtidas, se a utilização da modularidade possibilita uma melhor estruturação do *software* como um todo. Para 53,70% dos respondentes a modularidade promove em todos os casos uma melhor estruturação de projetos de *software*, 40,74% afirmaram na maioria dos casos, 3,70% responderam que eventualmente e para 1,85% isso raramente acontece.

Pressman (2011) afirma que utilizando a modularidade, os testes e depuração podem ser conduzidos de forma mais eficaz e a manutenção no longo prazo pode ser realizada sem efeitos colaterais sérios. Ter um sistema melhor estruturado, de certa forma, promove um melhor desenvolvimento de *software*, pois define o padrão de desenvolvimento a ser seguido, padronizando os relacionamentos e dependências em cada parte da aplicação.

GRÁFICO 12 - Com que frequência em seus projetos a abstração do código possibilitou um melhor aprendizado acerca das regras do sistema a ser desenvolvido, diminuindo assim a complexidade no desenvolvimento?

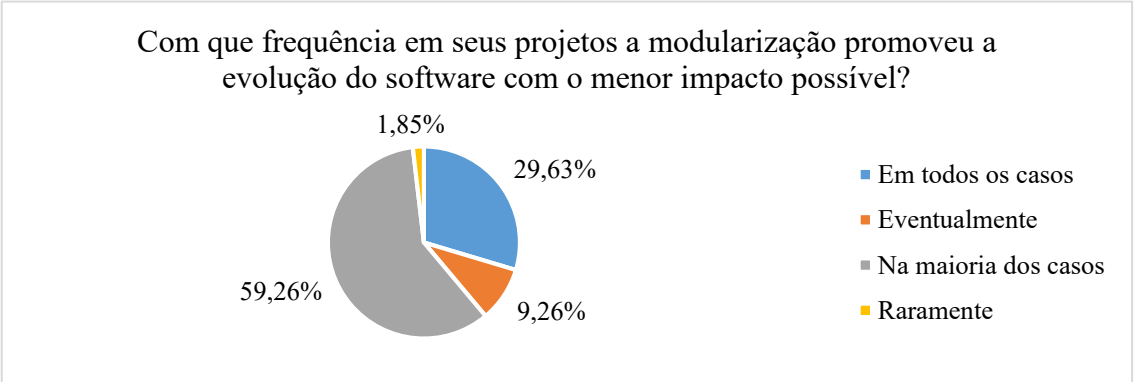


Fonte: Próprio autor (2017)

No Gráfico 12 é abordado o quanto a abstração do código diminui a complexidade no desenvolvimento, possibilitando um melhor entendimento acerca das regras de negócio da aplicação. De acordo com 51,85% dos participantes do questionário, na maioria dos casos a abstração contribui no desenvolvimento do *software*, 27,78% afirmaram em todos os casos, 14,81% eventualmente, 3,70% nunca e 1,85% raramente.

De acordo com Mendes (2002), engenheiros de *software* fazem uso da abstração para poder lidar com a complexidade de sistemas a serem desenvolvidos. Um exemplo é utilizar a palavra *notebook*, para identificar um objeto composto com diversos componentes e placas de circuito. Sendo assim, é possível explicar como utilizar um *notebook* a alguém que não conhece, sem a necessidade de compreender sua arquitetura e funcionamento interno.

GRÁFICO 13 - Com que frequência em seus projetos a modularização promoveu a evolução do *software* com o menor impacto possível?



Fonte: Próprio autor (2017)

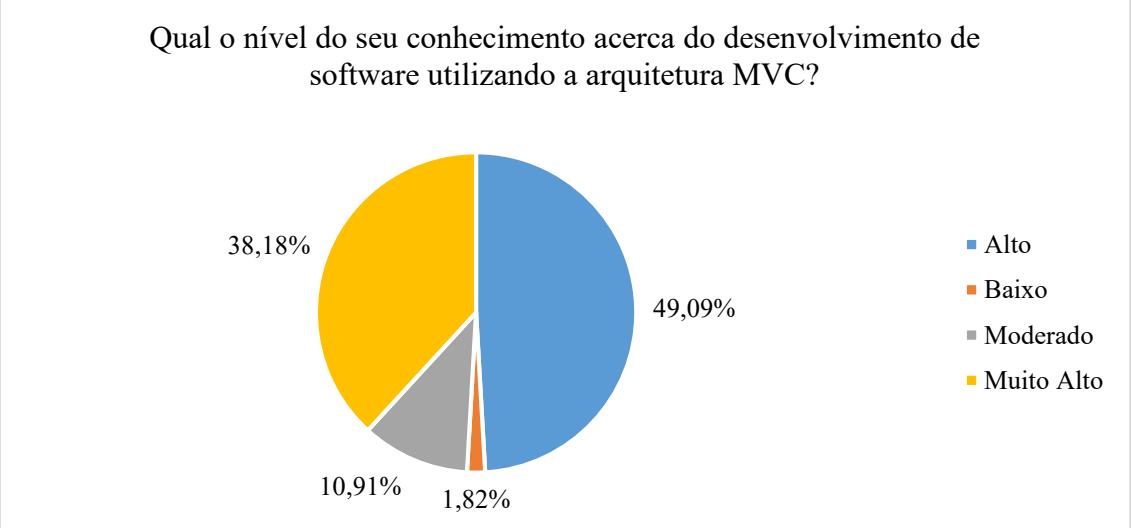
De acordo com o Gráfico 13, no qual é perguntado com que frequência a adoção da modularidade no desenvolvimento de um *software* promove a capacidade de sua evolução, sendo assim, não possuindo grades impactos em sua modificação. Para 59,26% dos respondentes, na maioria dos casos a modularidade foi importante para promover a evolução de seus projetos, 29,63% afirmaram em todos os casos, 9,26% eventualmente e apenas 1,85% raramente.

Mendes (2002) afirma, que a modularidade possibilita a composição de um sistema a partir de componentes que incrementalmente vão sendo agregados até a obtenção de um sistema completo. Sendo assim, as aplicações terão capacidade de suportar o crescimento e modificações futuras.

3.3 Resultados da utilização do MVC

Nesta seção serão apresentados os resultados acerca das 55 respostas consideradas válidas, estas são referentes aos participantes que afirmaram possuir experiência no desenvolvimento de *softwares* utilizando o MVC, sendo capazes de responder as perguntas referentes às técnicas do tema proposto.

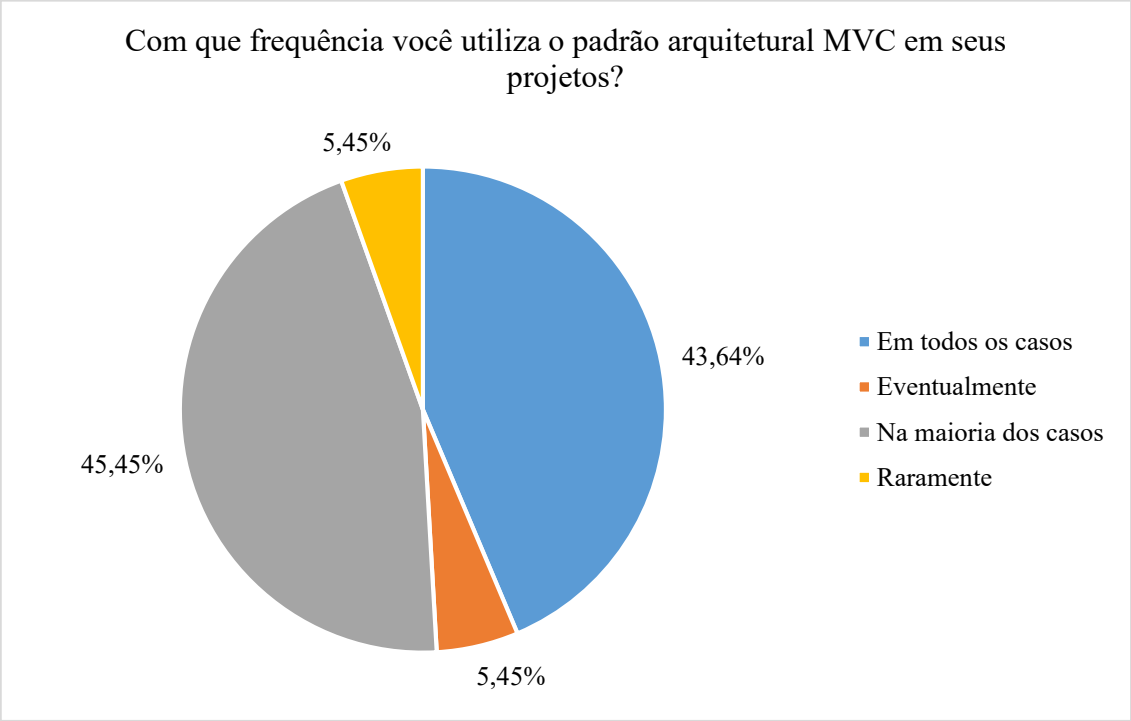
GRÁFICO 14 - Qual o nível do seu conhecimento acerca do desenvolvimento de *software* utilizando a arquitetura MVC?



Fonte: Próprio autor (2017)

O nível de conhecimento no desenvolvimento de *software* utilizando o MVC dos participantes, demonstrado no Gráfico 14, é 49,09% alto, 38,18% afirmaram ser muito alto, 10,91% possuem conhecimento moderado e apenas 1,82% possuem um baixo conhecimento. Sendo assim, é possível notar que a maioria dos respondentes informaram possuir os conhecimentos necessários para serem aptos a responder as questões técnicas acerca de *software* MVC e suas características.

GRÁFICO 15 - Com que frequência você utiliza o padrão arquitetural MVC em seus projetos?

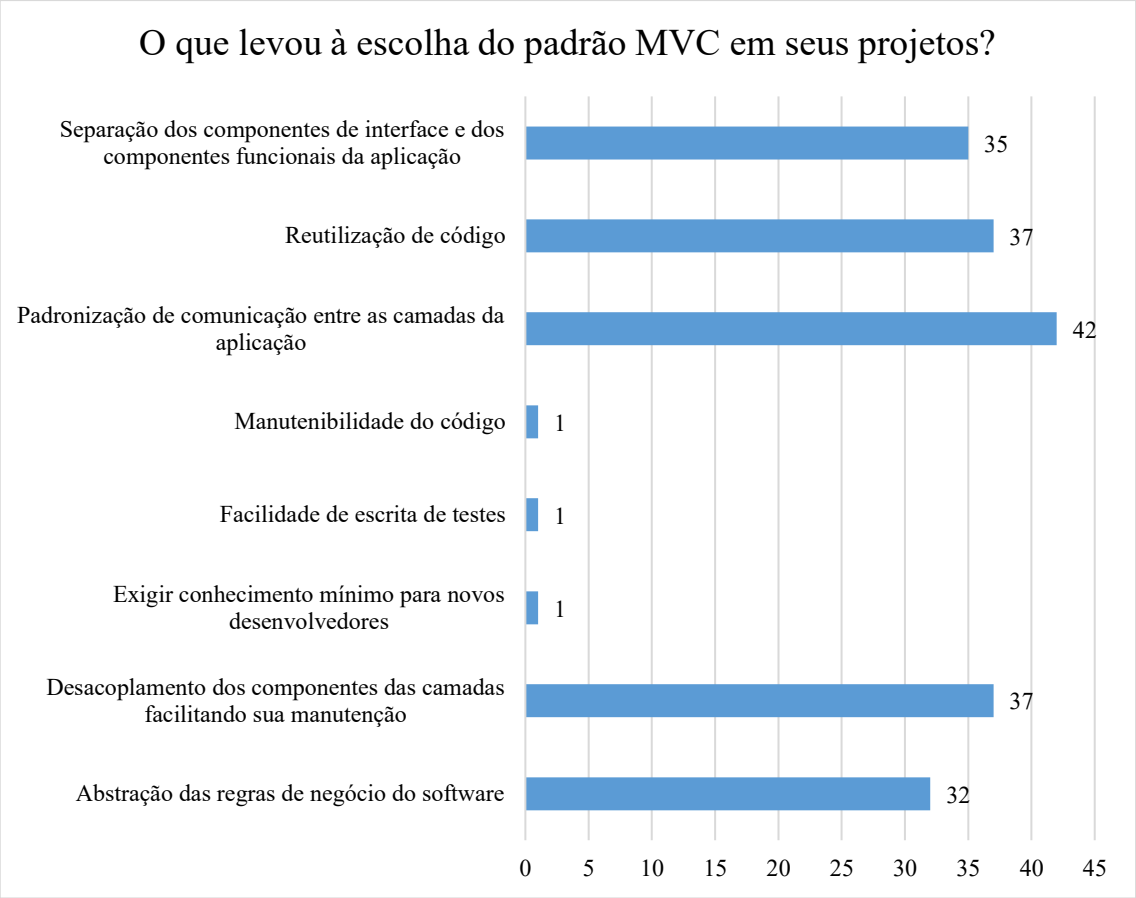


Fonte: Próprio autor (2017)

No Gráfico 15 são exibidos os resultados acerca da frequência de utilização do padrão arquitetural MVC nos projetos em que os entrevistados tiveram contato e experiência profissional. Em todos os casos afirmaram 43,64%, 45,45% utilizaram o MVC na maioria dos casos e 5,45% afirmaram utilizar eventualmente ou raramente.

É possível notar a partir destes resultados que embora o MVC seja um padrão relativamente antigo, com o passar dos anos vêm sendo consolidado e atualmente continua sendo bastante utilizado nas elaborações de aplicações em *software*.

GRÁFICO 16 - O que levou à escolha do padrão MVC em seus projetos?

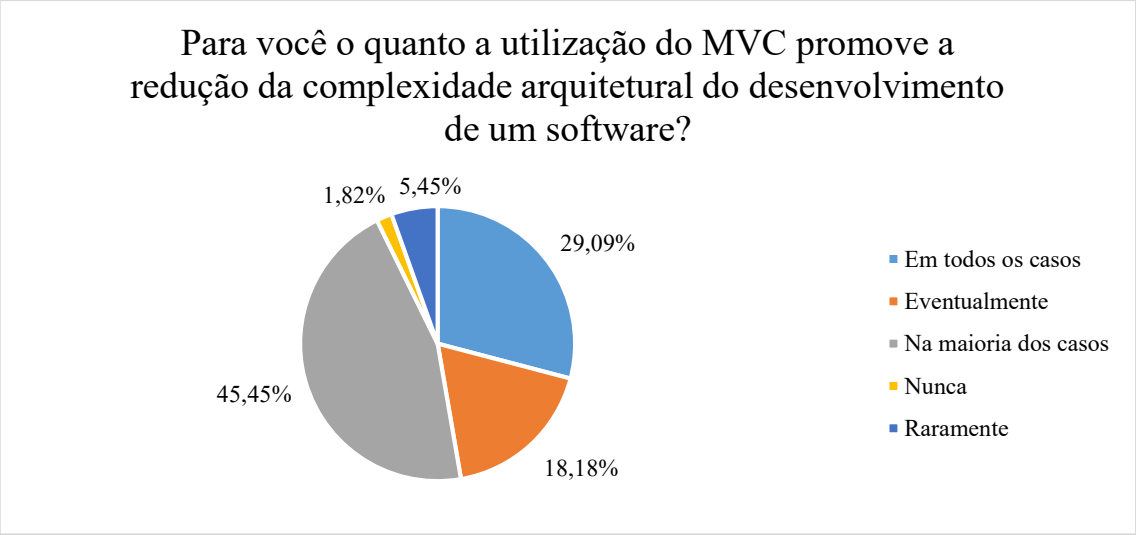


Fonte: Próprio autor (2017)

O Gráfico 16 destaca quais são os motivos que motivaram, por parte dos profissionais participantes deste questionário ou até mesmo de sua equipe de desenvolvimento, a utilizarem o padrão MVC em seus projetos de *software*.

O principal motivo a ser destacado com 42 escolhas dentre os 55 respondentes, é o benefício da padronização de comunicação entre as camadas da aplicação. Já a reutilização de código e o desacoplamento dos componentes das camadas, ambos possuem 37 escolhas. A separação dos componentes de interface e dos componentes funcionais da aplicação foi outro motivo bastante significativo, tendo 35 escolhas. Para 32 respondentes a abstração das regras de negócio do software foi um dos motivos que levou a adoção do MVC. Porém a manutenibilidade do código, exigência de conhecimento mínimo para novos desenvolvedores e facilidade de escrita de testes foram fatores menos importantes tendo apenas 1 escolha cada.

GRÁFICO 17 - Para você o quanto a utilização do MVC promove a redução da complexidade arquitetural do desenvolvimento de um software?

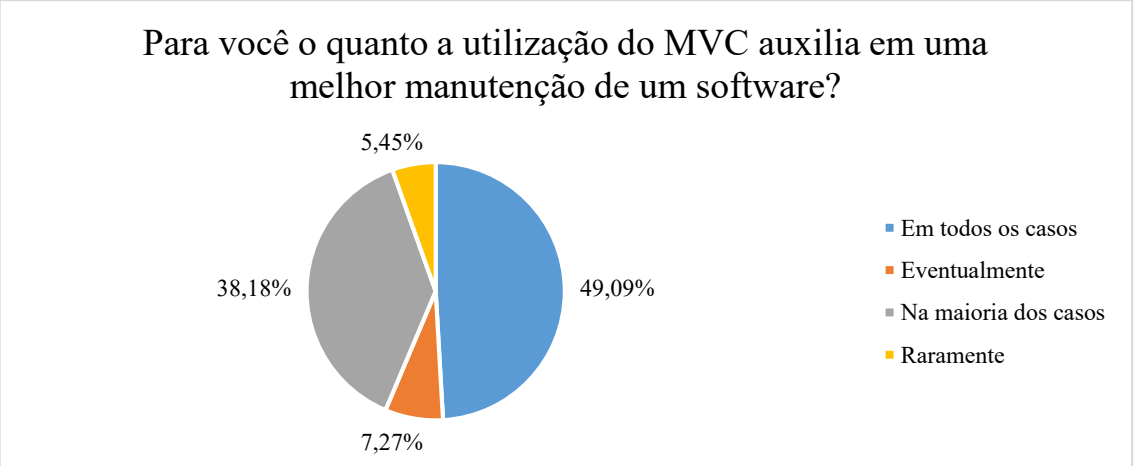


Fonte: Próprio autor (2017)

O Gráfico 17 aborda o quanto a utilização do padrão arquitetural MVC promove e auxilia de alguma forma para a redução da complexidade no desenvolvimento de um software. Dentre os 55 respondentes, 45,45% afirmaram que na maioria dos casos o MVC promove sim a redução da complexidade, 29,09% indicaram em todos os casos, 18,18% eventualmente, 5,45% raramente e apenas 1,82% afirmaram que nunca promove.

Para Fowler (2006), o valor do MVC está na separação de camadas, sendo assim, a separação da *view* e do modelo é um dos mais importantes princípios no projeto de um *software*, deve-se aplicar quando tiver alguma lógica não visual. Promovendo assim, a melhor estruturação das camadas do sistema e padronizando a comunicação da aplicação.

GRÁFICO 18 - Para você o quanto a utilização do MVC auxilia em uma melhor manutenção de um software?

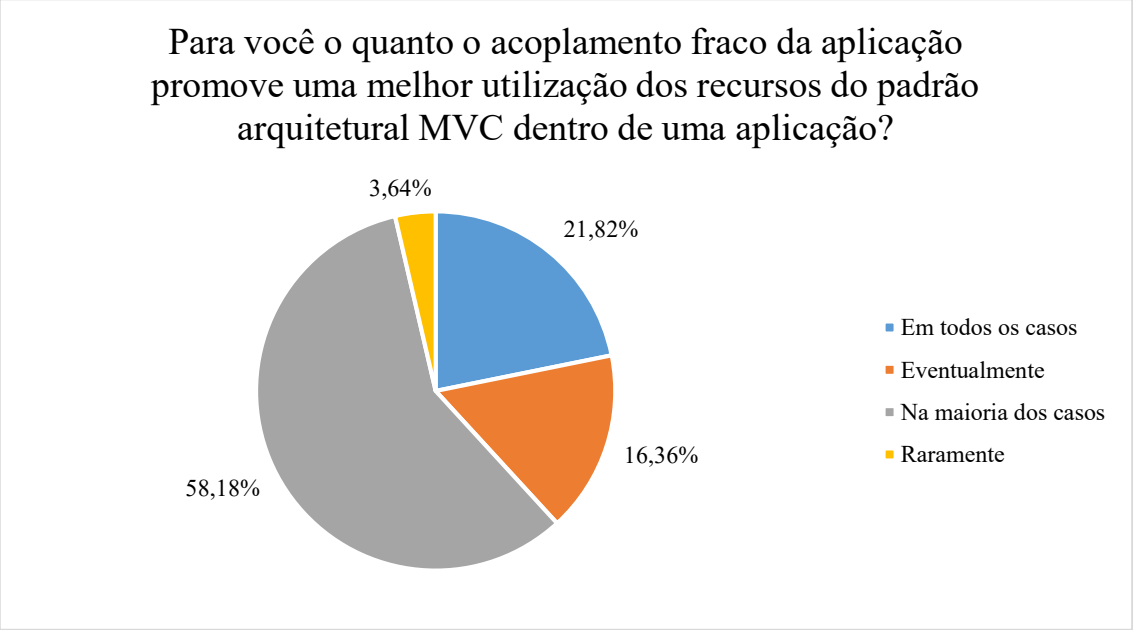


Fonte: Próprio autor (2017)

Observando o Gráfico 18 pode-se analisar o resultado acerca das 55 respostas válidas, referente ao quanto a utilização do MVC auxilia em uma melhor manutenção de um *software*. De acordo com 49,09% dos respondentes, em todos os casos o MVC promove de alguma forma uma melhor manutenção de *software*. Para 38,18% na maioria dos casos isso ocorre, 7,27% afirmaram eventualmente e apenas para 5,45% raramente ocorre.

O padrão MVC promove uma melhor manutenção do *software*, pois segundo Gamma (2000), este separa os objetos componentes pertencentes ao padrão para aumentar a flexibilidade e a reutilização de código, estabelecendo um protocolo de comunicação entre eles.

GRÁFICO 19 - Para você o quanto o acoplamento fraco da aplicação promove uma melhor utilização dos recursos do padrão arquitetural MVC dentro de uma aplicação?



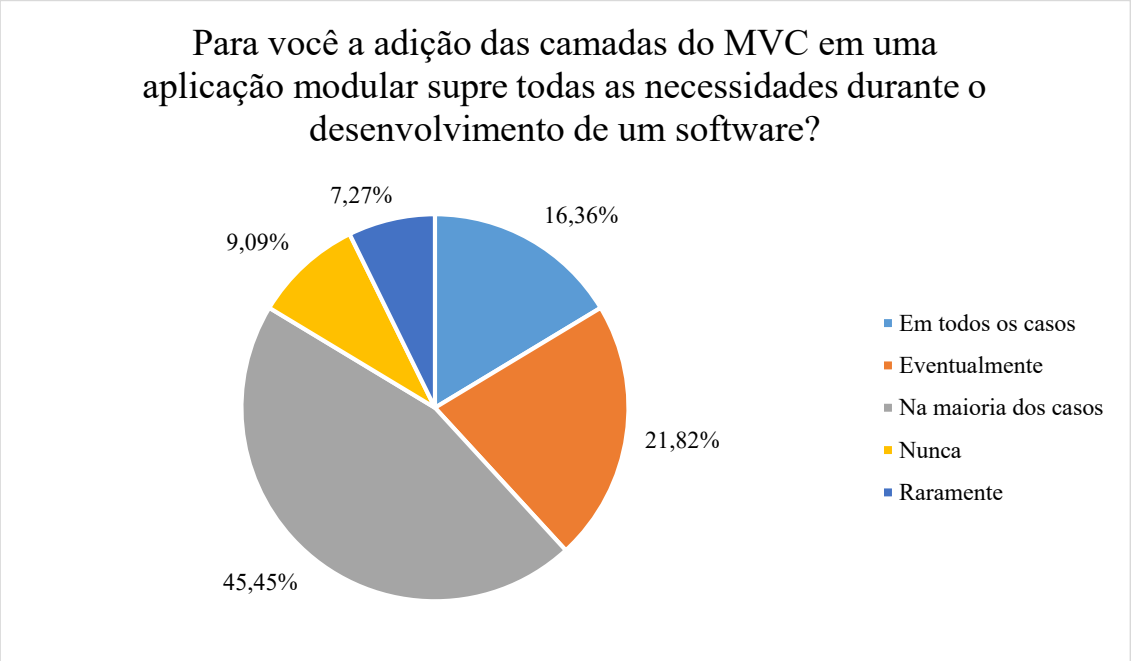
Fonte: Próprio autor (2017)

O Gráfico 19 apresenta os resultados obtidos através do questionamento, se os recursos do padrão arquitetural MVC na aplicação são melhores aproveitados tendo um sistema fracamente acoplado. Para 58,18% dos respondentes na maioria dos casos você ter uma aplicação menos dependente é um cenário ideal para a utilização do MVC, 21,82% afirmaram ser em todos os casos, 16,36% eventualmente e apenas para 3,64% nunca.

É possível aferir que parte dos respondentes que tiveram experiência no desenvolvimento de sistemas modulares com MVC, não conseguiram compreender a

interação entre o requisito não funcional com o padrão de projeto. Segundo Pressman (2011), a coesão é importante no projeto de componentes e classes, encapsulando atributos e operações que estejam relacionadas entre si, sendo fáceis de serem implementadas e mantidas.

GRÁFICO 20 - Para você a adição das camadas do MVC em uma aplicação modular supre todas as necessidades durante o desenvolvimento de um software?

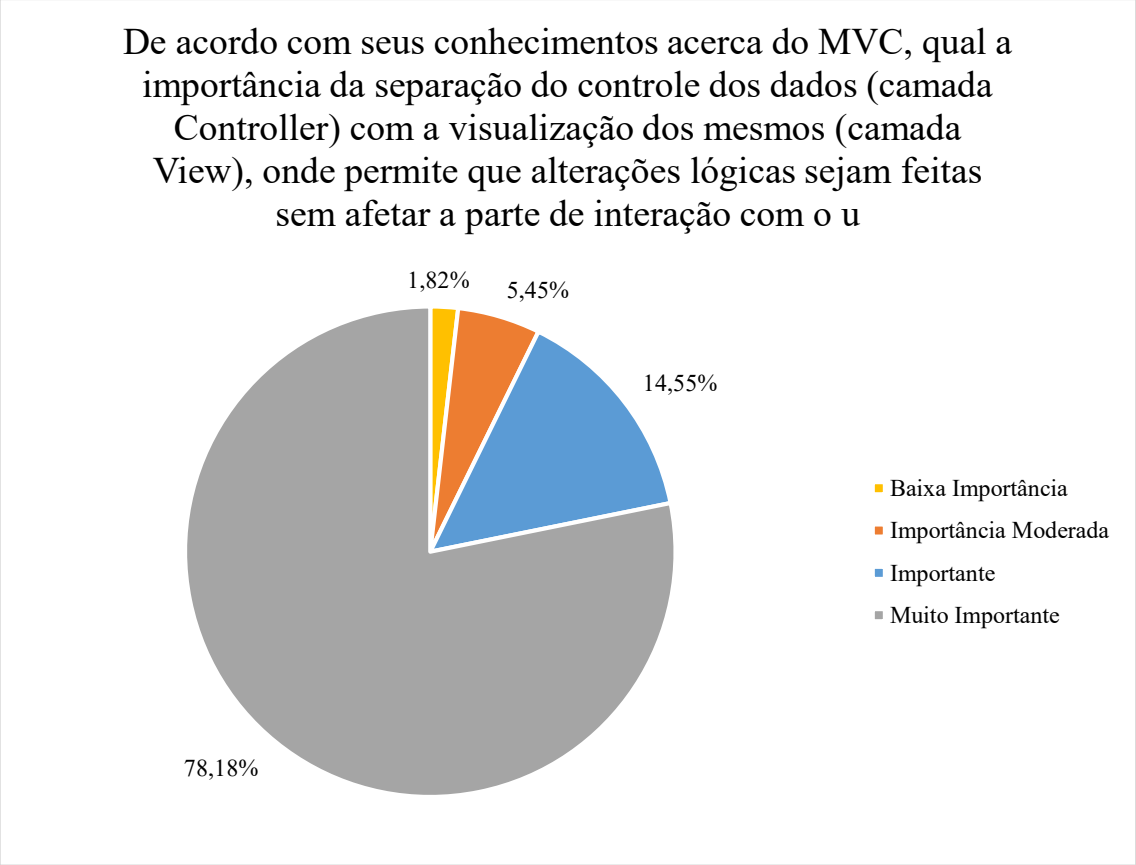


Fonte: Próprio autor (2017)

Observa-se no Gráfico 20, que 45,45% dos participantes afirmaram que na maioria dos casos a combinação do MVC em uma aplicação modular supre todas as necessidades no desenvolvimento de um *software*. 21,82% afirmaram que eventualmente isto ocorre, já para 16,36% em todos os casos a aplicação se beneficia da utilização do MVC e da modularização. Para 9,09% isto nunca ocorre e para apenas 7,27% raramente.

Sendo assim, é possível notar que a aplicação do MVC em uma aplicação modular deve ser planejada levando em conta às necessidades do projeto e quais serão seus benefícios no mesmo.

GRÁFICO 21 - De acordo com seus conhecimentos acerca do MVC, qual a importância da separação do controle dos dados (camada Controller) com a visualização dos mesmos (camada View), onde permite que alterações lógicas sejam feitas sem afetar a parte de interação com o usuário?

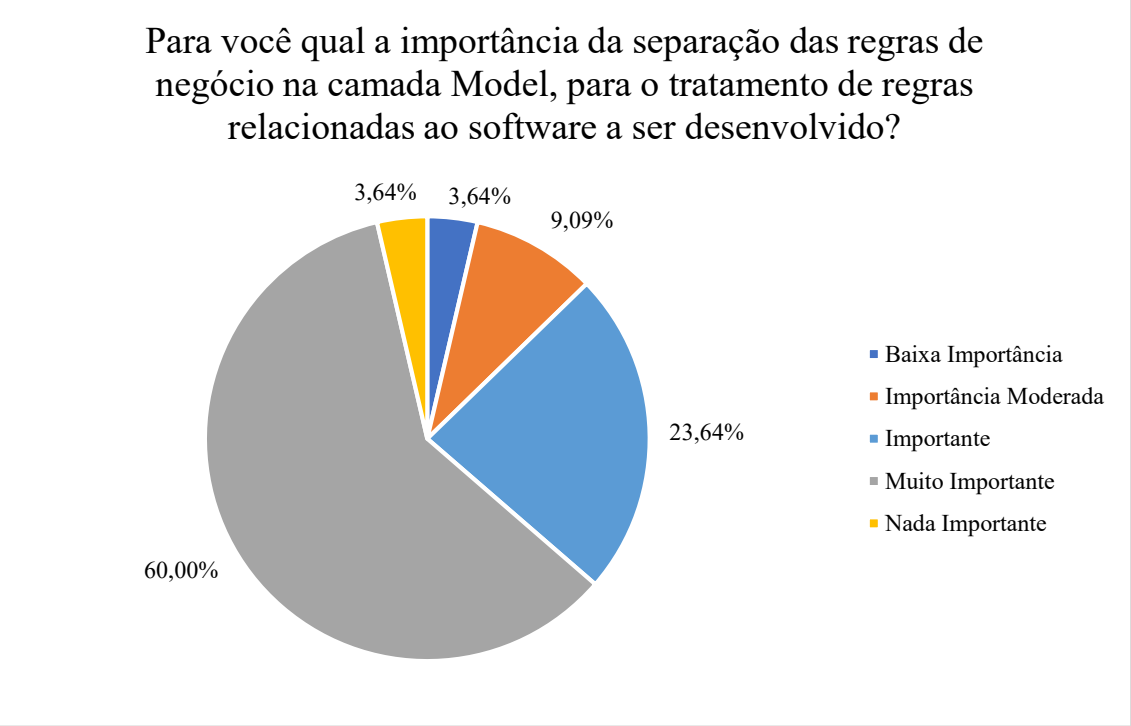


Fonte: Próprio autor (2017)

O Gráfico 21 apresenta os resultados obtidos acerca da importância de se permitir alterações na lógica e fluxo dos dados através da camada de *Controller* na aplicação, sem afetar a interação direta com o usuário na camada de *View*. Observa-se que 78,18% dos respondentes afirmara ser muito importante a separação das camadas, 14,55% afirmaram ser importante, 5,45% importância moderada e apenas 1,82% afirmou de ser baixa importância este tipo de abordagem.

Segundo Fowler (2006), a separação das camadas de *view* com a de *controller* é importante para suportar comportamento editável e não-editável, tendo um controlador para cada página lógica de uma *view* onde é possível alterar sem interferir no tratamento dos dados.

GRÁFICO 22 - Para você qual a importância da separação das regras de negócio na camada Model, para o tratamento de regras relacionadas ao software a ser desenvolvido?

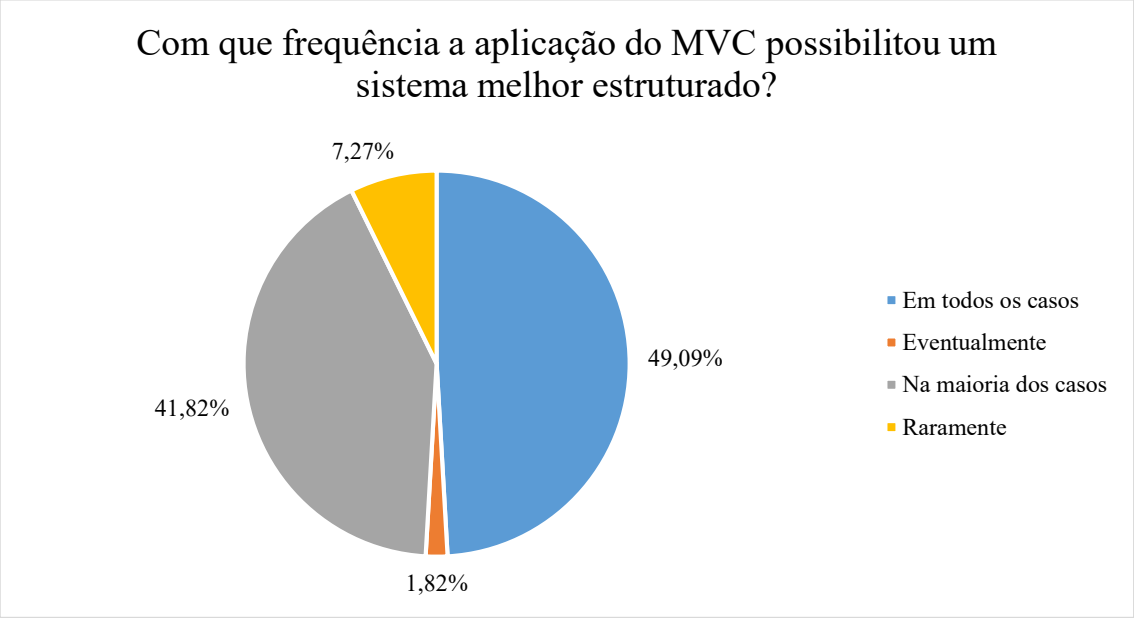


Fonte: Próprio autor (2017)

No Gráfico 22 observa-se os resultados referente à importância de ser ter a camada de *Model* na aplicação, para a separação das regras de negócio e que seja possível o melhor tratamento das regras do mesmo. De acordo com 60% é muito importante que a aplicação tenha a separação de suas regras específicas, 13,64% afirmaram que é importante e 9,09% importância moderada. Já baixa importância e nada importante afirmaram 3,64% para cada.

Os resultados obtidos e expostos no Gráfico 22 vão de encontro ao que Fowler (2006) afirmou, para ele um ponto chave é a direção de dependências, a camada de *view* depende da camada de *model*, mas para cria um *model* não é necessário possuir conhecimento da camada de *view*. Deve-se levar em conta, que é de responsabilidade da camada de *controller* encaminhar os dados entre a *view e o model*. Sendo assim, as regras de negócio não sofrerão interferências das demais camadas.

GRÁFICO 23 - Com que frequência a aplicação do MVC possibilitou um sistema melhor estruturado?

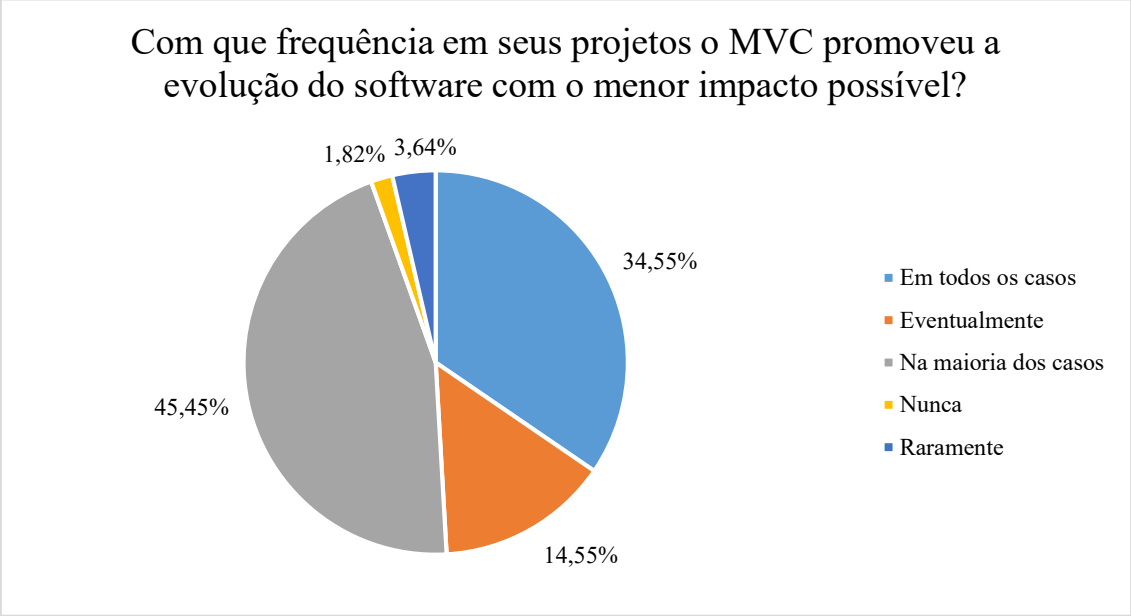


Fonte: Próprio autor (2017)

Observa-se no Gráfico 23 que 49,09% afirmaram que em todos os casos, a aplicação das camadas do MVC possibilitou a construção de um sistema de *software* melhor estruturado. Para 41,82% apenas na maioria dos casos, já 7,27% afirmaram que raramente o MVC promove uma melhor estruturação da aplicação e apenas 1,82% afirmaram que eventualmente isto ocorre.

É possível aferir com base nestes resultados que o padrão arquitetural MVC é muito utilizado em projetos de sistemas de *software* visando alcançar uma melhor organização arquitetural da aplicação, seja na melhor estruturação das camadas ou até mesmo em sua comunicação.

GRÁFICO 24 - Com que frequência em seus projetos o MVC promoveu a evolução do software com o menor impacto possível?



Fonte: Próprio autor (2017)

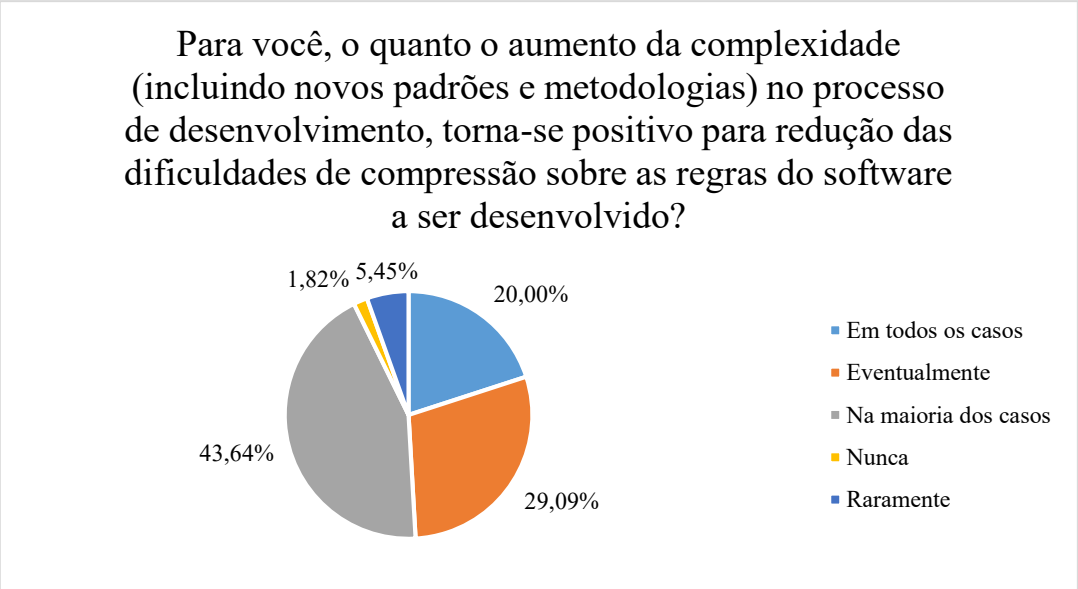
O Gráfico 24 apresenta os resultados que abordam, em qual frequência a utilização do MVC promoveu a evolução do *software* em um menor impacto possível dentro da aplicação. Como resultado, 45,45% afirmaram que na maioria dos casos isto ocorre, 34,55% em todos os casos e eventualmente 14,55%. Já para 3,64% raramente o MVC promove a redução do impacto de evolução de um *software* e para 1,82% nunca isto ocorre.

É possível observar através dos resultados apresentados, que conforme a aplicação de *software* cresce, a complexidade no desenvolvimento também cresce. Mesmo utilizando-se de padrões que auxiliam e estruturam o projeto, estes não impedem 100% os impactos causados pela alteração da aplicação.

3.4 Resultados da utilização de padrões arquiteturais

Nesta seção serão apresentados os resultados obtidos acerca da utilização de padrões arquiteturais no desenvolvimento de um *software*.

GRÁFICO 25 - Para você, o quanto o aumento da complexidade (incluindo novos padrões e metodologias) no processo de desenvolvimento, torna-se positivo para redução das dificuldades de compressão sobre as regras do software a ser desenvolvido?



Fonte: Próprio autor (2017)

O Gráfico 25, que representa os resultados obtidos acerca do quanto é positivo a inclusão de novos padrões e metodologias no desenvolvimento de um *software*, como por exemplo o MVC, promove de certa forma a redução das dificuldades de compreensão das regras de negócio da aplicação.

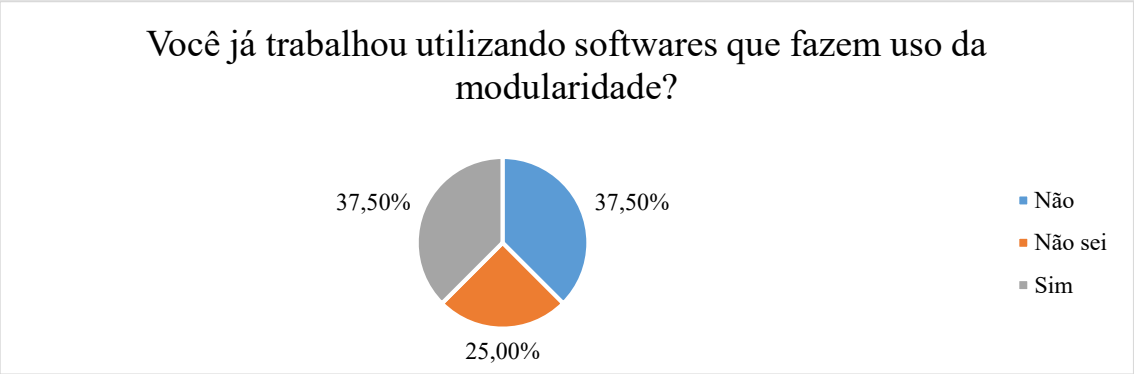
De acordo com 43,64% dos participantes, na maioria dos casos é positivo aumentar a complexidade do desenvolvimento, para facilitar uma melhor compressão do sistema a ser desenvolvido. Já para 29,09% afirmaram ser eventualmente positivo, enquanto 20% disseram ser positivo em todos os casos. Apenas 5,45% afirmaram ser raramente e 1,82% nunca positivo.

Conclui-se que nem sempre a utilização de padrões e metodologias no desenvolvimento, garante a melhora na compreensão das regras do *software* em desenvolvimento. Isto pode ocorrer, pelo fato que conforme a aplicação cresce a complexidade de seu desenvolvimento e manutenção também crescem, não sendo soluções completas, apenas paliativos.

3.5 Resultados dos participantes inexperientes

Nesta seção serão apresentados os resultados das respostas dos 16 participantes, estas são referentes aos que afirmaram não possuir experiência no desenvolvimento de softwares modulares utilizando a arquitetura MVC, apresentando sua experiência profissional e de sua empresa.

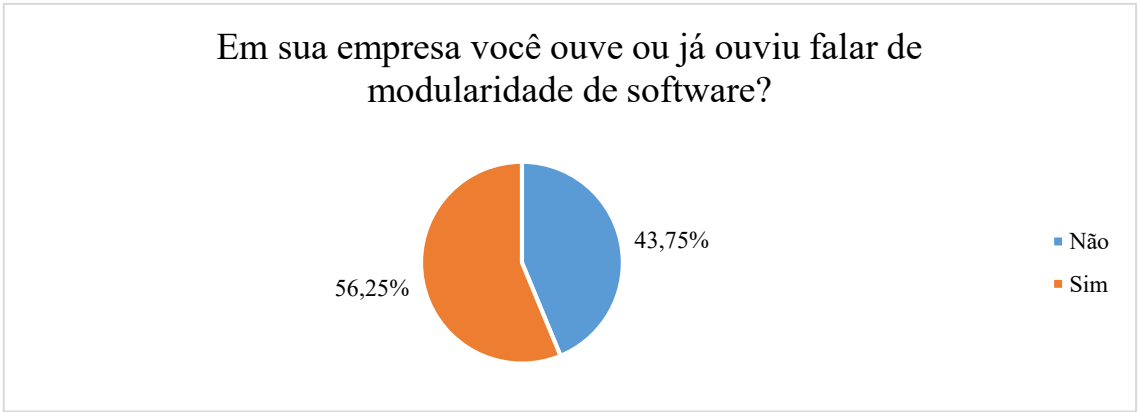
GRÁFICO 26 - Você já trabalhou utilizando softwares que fazem uso da modularidade?



Fonte: Próprio autor (2017)

Observa-se no Gráfico 26, que aproximadamente 37,50% dos respondentes afirmaram que já trabalharam utilizando *softwares* que foram desenvolvidos com a utilização da modularidade. Outros 37,5% afirmaram também que nunca trabalharam, já 25% afirmaram que não sabem se já utilizaram sistemas desenvolvidos com esta abordagem. Sendo assim é possível notar um equilíbrio entre a utilização e não utilização de *softwares* modulares.

GRÁFICO 27 - Em sua empresa você ouve ou já ouviu falar de modularidade de software?

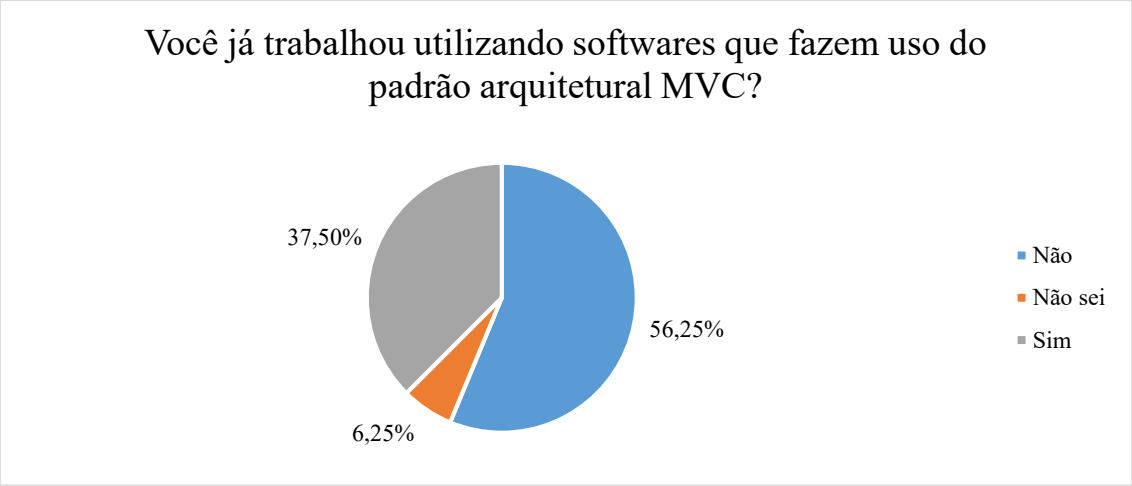


Fonte: Próprio autor (2017)

Através do Gráfico 27 é possível observar que 56,25% dos respondentes já ouviram falar da modularidade de *software* na empresa na qual trabalha, já 43,75% afirmaram que nunca ouviram falar deste tipo de abordagem no desenvolvimento.

Observa-se através destes resultados e relacionando-os ao gráfico 26, nota-se a falta de conhecimento por parte dos respondentes acerca do sistema que utiliza em sua empresa, embora a maioria tenha dito que já ouviu falar desta abordagem durante sua experiência na empresa na qual trabalha.

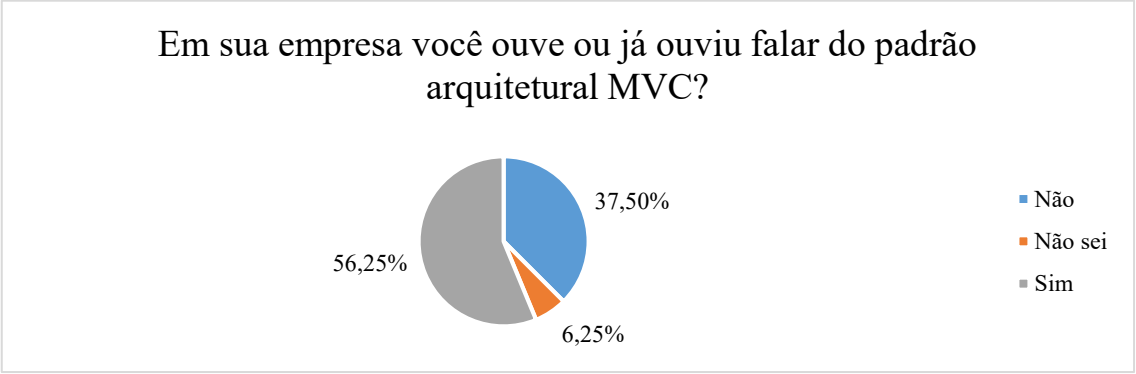
GRÁFICO 28 - Você já trabalhou utilizando softwares que fazem uso do padrão arquitetural MVC?



Fonte: Próprio autor (2017)

De acordo com o Gráfico 28, 56,25% dos participantes afirmaram não terem trabalhado utilizando *softwares* desenvolvidos com o MVC, 37,50% afirmaram ter trabalhado utilizando *softwares* desenvolvidos utilizando MVC, já 6,25% afirmaram não saber.

GRÁFICO 29 - Em sua empresa você ouve ou já ouviu falar do padrão arquitetural MVC?



Fonte: Próprio autor (2017)

Observa-se no Gráfico 29 que 56,25% dos respondentes afirmaram já terem ouvido falar do padrão arquitetural MVC na empresa ao qual trabalha, 37,50% não ouviram falara e apenas 6,25% afirmaram não saber.

Relacionando os resultados obtidos no gráfico 28, é possível notar que a maioria dos respondentes embora terem afirmado não trabalhar utilizando *softwares* desenvolvidos com o MVC, já ouviram falar sobre suas abordagens em seu ambiente de trabalho. Porém é notório o grande número de pessoas que nunca ouviram falar desta abordagem, sendo muito importante dentro de uma organização o compartilhamento de conhecimento com os profissionais que são recém-chegados e até mesmo com aqueles que não desenvolvem *softwares*, mas trabalham diretamente com eles. Para Sommerville (2012), os padrões de *software* são importantes pois são baseados em conhecimentos adquiridos sobre a prática do que é melhor ou mais adequado para uma empresa.

3.6 Discussão dos resultados

Nesta seção serão discutidos os resultados obtidos através do questionário qualitativo e das pesquisas realizadas ao longo deste trabalho, com o intuito de avaliar a importância e os benefícios de se desenvolver *softwares* modulares em arquitetura MVC.

Com base no perfil dos participantes entrevistados, foi possível constatar que os profissionais em seus primeiros 6 anos de experiência na área, tiveram mais contato com o padrão MVC do que com a modularidade, possuindo assim mais conhecimento no padrão arquitetural. Observou-se também, que a maioria dos profissionais considerou muito importante a utilização da modularidade no desenvolvimento de um *software*.

Foi apontada pela maioria dos respondentes que a utilização da modularidade promove a redução da complexidade no desenvolvimento, pois possibilita a abstração de código. Além disso, promove uma melhor organização da aplicação e a reutilização de código, pois permite a divisão de partes da aplicação entre diferentes desenvolvedores.

Observa-se que a adoção da modularidade promove um *software* mutável, ou seja, promove uma melhor aceitação tanto na realização de mudanças evolutivas, quanto na necessidade de realizar manutenções rotineiras, causando menor impacto à aplicação.

Além disso a abstração de código promove a redução da complexidade no desenvolvimento, pois permite uma melhor análise das regras de negócio da aplicação.

De acordo com os respondentes, o acoplamento fraco dos módulos e o encapsulamento dos componentes, sendo características indispensáveis para atingir a modularidade no desenvolvimento de uma aplicação, são muito importantes. Pois promove a redução das dependências entre os componentes presentes em cada módulo e promove uma melhor manuseabilidade da aplicação.

> Uma vez que você tenha criado módulos que aproveitam do encapsulamento, da abstração e dos acoplamentos fracos, terá como resultado os benefícios da capacidade de reutilização, da sustentabilidade e da confiabilidade em toda a sua aplicação web. Quando um módulo é reutilizável, fica claro como devemos proceder para fazê-lo operar e realizar novas tarefas. Você pode movê-lo, entre vários pontos da aplicação, certo de que tudo de que ele necessita para funcionar corretamente o acompanhará e de que você não encontrará consequências inesperadas quando reutilizá-lo. (LOUDON, 2010, p.21).

Foi apontada pela maioria dos participantes que a utilização do MVC promove a redução da complexidade arquitetural de um *software*, pois permite a padronização da comunicação entre as camadas da aplicação. Além de facilitar a sua manutenção, promovendo o desacoplamento dos componentes de suas camadas e permitir a separação dos componentes de interface com o usuário dos componentes funcionais da aplicação. Observa-se também que a aplicação do MVC possibilita um sistema melhor estruturado, sendo assim, é importante a separação das regras de negócio na camada de *model*, utilizada no tratamento de regras relacionadas ao software a ser desenvolvido.

> [...] os componentes controller e model comunicam-se uns com os outros através de evento. Os eventos fornecem um mecanismo de comunicação com menor grau de acoplamento, o que permite que a comunicação tenha dependências mínimas. (LOUDON, 2010, p.133).

Foi apontado também, que há uma melhor utilização dos recursos do padrão MVC em uma aplicação, quando aplicado a um sistema modularizado possuindo módulos fracamente acoplados.

CONCLUSÃO

De fato, a necessidade do rápido desenvolvimento de sistemas complexos garantindo assim a melhor qualidade possível, é um grande desafio em qualquer projeto de *software* de grande escala. Para se obter um sistema de melhor qualidade é necessário promover uma maior facilidade de manutenção, reutilização de código, além de atender aos requisitos e desempenho iniciais definidos em seu projeto. A utilização de padrões arquiteturais e paradigmas, como por exemplo, o padrão arquitetural MVC e o requisito não funcional modularidade, são importantes para promover estes objetivos.

Ao decorrer deste trabalho, foi possível aprofundar os estudos acerca da importância e benefícios na utilização do padrão arquitetural MVC no desenvolvimento de aplicações com componentes modulares, avaliando também sua importância para a redução da complexidade, capacidade de mudança e manutenção dos códigos de uma aplicação.

Foi realizado a aplicação de um questionário qualitativo para a obtenção dos resultados, acerca das técnicas e características utilizadas por estas abordagens, tendo como público-alvo profissionais de TI e que já tenham desenvolvido *softwares* com componentes modulares e utilizando-se do padrão MVC. Dentre os resultados encontrados, nota-se um alto índice de utilização das abordagens descritas, sendo a padronização de comunicação entre as camadas o principal motivo para a utilização do MVC e uma melhor organização da aplicação o principal motivo para a utilização da modularidade.

Portanto com base nos resultados obtidos foi possível constatar que o MVC promove a redução da complexidade encapsulando e separando em camadas o tratamento dos dados e a exibição dos mesmos, fazendo com que qualquer alteração e crescimento necessário não impacte nos demais componentes existentes. Já a modularidade promove a redução da complexidade de compreensão do *software*, pois ao decompor o sistema em pequenas partes, faz com que o desenvolvedor abstraia as demais regras existentes, já que um módulo não impactará nos demais, possibilitando o desenvolvimento do mesmo em partes. Sendo assim, a inclusão do MVC e da modularidade é de fato positivo para promover a redução das dificuldades de compreensão acerca das regras da aplicação. Porém é de extrema importância analisar a real necessidade da utilização destes

paradigmas, levando em conta desde os requisitos do sistema até sua estrutura, quais benefícios os mesmos trarão para o projeto e qual o nível de percepção de crescimento o mesmo terá.

TRABALHOS FUTUROS

- Comparativo entre um *software* sendo desenvolvido utilizando somente a modularidade e um *software* desenvolvido utilizando somente o MVC.

- Realização de um comparativo utilizando-se de diferentes *frameworks* acerca da abordagem MVC aliado a modularidade.

- Realizar um estudo de caso com uma equipe de desenvolvimento para a avaliação no desenvolvimento de uma aplicação modular com MVC.

- Comparativo entre a modularidade aliado ao MVC e a modularidade aliada a outro padrão de projeto arquitetural.

REFERÊNCIAS

BOOCH, G; RUMBAUGH, J; JACOBSON, I**. UML: Guia do Usuário.** 9ªed. Editora Campus, 2000.

CASSIMIRO, M. H. de O. **Padrões arquiteturais e seus benefícios no processo de manutenção de *softwar*e**. Belo Horizonte. 2010.

CODERS EYE. **11 Best PHP Frameworks for Modern Web Developers in 2017.** Disponível em: < https://coderseye.com/best-php-frameworks-for-web-developers/>. Acesso em: 16 de nov. 2017.

CONVERSE, T; PARK, J; MORGAN, C. **PHP5 and MySQL Bible.** Editora Wiley Publishing, 2004. Disponível em: <https://www.passeidireto.com/arquivo/1215557/biblia---php-5-and-mysql-mcgraw-hill>. Acesso em: 18 de jun. 2017.

EDNALDO, Erick de Lima. **A crise do software.** Disponível em: < https://pt.linkedin.com/pulse/crise-do-software-erick-ednaldo-de-lima>. Acesso em: 15 de nov. 2017.

FILHO, A. M. da S. **Desenvolvimento de *Software* requer Processo e Gestão.** 2011. Disponível em: <http://periodicos.uem.br/ojs/index.php/EspacoAcademico/article/download/14312/7593>. Acesso em: 18 de jun. 2017.

FOWLER, M; et al. **Padrões de Arquitetura de Aplicações Corporativas.** 1ªed. Porto Alegre: Editora Bookman, 2006.

GAMMA, E; et al. **Padrões de projeto: soluções reutilizáveis de *software* orientado a objetos.** 1ªed. Porto Alegre: Editora Bookman, 2000.

GERMOGLIO, G. **Arquitetura de *software*.** 2010. Disponível em: <http://cnx.org/contents/jh8AyIna@9.1:4fyAw9kj@7/Mensagens-do-Livro>.

JACYNTHO, M. D de A. **Processos para Desenvolvimento de Aplicações Web.** 2008. Disponível em: < ftp://ftp.inf.puc-rio.br/pub/docs/techreports/09_23_jacyntho.pdf>.

LIMA, P; ADRIANA. C; CARNIELLO, A. **Aplicando atam na arquitetura do sistema de informações de prefeituras.** Bragança Paulista, SP. 2014.

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

MENDES, A. **Arquitetura de *Software*: desenvolvimento orientado para arquitetura.** Editora Campus, 1ªed. Editora Campus, 2002.

MINETTO, Elton L. ***Frameworks* para desenvolvimento em PHP.** Editora Novatec Ltda, 2007.

PRESSMAN, R S. **Engenharia de *Software*: Uma Abordagem Profissional.** 7ªed. Porto Alegre: AMGH Editora Ltda. 2011.

PRIKLADNICKI, R. **Problemas, Desafios e Abordagens do Processo de Desenvolvimento de *Software*.** Disponível em: <http://www.inf.pucrs.br/munddos/docs/TI1.pdf>. Acesso em: 29 de out. 2017.

SOMMERVILLE, I. **Engenharia de *Software*.** 9ªed. São Paulo: Editora Pearson Brasil, 2012.

SOUZA, O. R de. **Processos de apoio ao desenvolvimento de aplicações Web.** USP – São Carlos, 2005.

VARGAS, T. C de S. **A história de UML e seus diagramas**. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_721/artigo.tcc.pdf>. Acesso em: 23 de abr. 2017.

WELLING, L; THOMSON, L. **PHP e MySQL: Desenvolvimento Web.** 3ªed. Editora Campus. Disponível em: <https://www.passeidireto.com/arquivo/11002408/livro-php-e-mysql-desenvolvimento-web>. Acesso em: 23 de abr. 2017.

APÊNDICE 1 - QUESTIONÁRIO

Software Modular em arquitetura MVC

Este questionário é destinado ao auxílio da elaboração de trabalho acadêmico do curso de Ciência da Computação.

Tem como objetivo avaliar pontos acerca da utilização da modularidade aliado ao padrão arquitetural Model-View-Controller (MVC) como solução à problemas de complexidade no desenvolvimento de software.

Todos os dados coletados são confidenciais e serão utilizados para fins acadêmicos. Apenas os resultados serão divulgados para a conclusão do trabalho e para o crescimento da comunidade.

Desde já agradeço a contribuição e o incentivo!

*Obrigatório

1. Perfil do Participante

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

1. Informe seu e-mail caso queira receber os resultados deste questionário

2. Qual a sua idade? * Marcar apenas uma oval.

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

3. Em qual estado vive atualmente?

_____

4. Qual a sua formação acadêmica? *
Marcar apenas uma oval.

- ◯ Ensino Médio Completo
- ◯ Técnico
- ◯ Ensino Superior Incompleto
- ◯ Ensino Superior Completo
- ◯ Pós-Graduação
- ◯ Mestrado
- ◯ Doutorado
- ◯ Outro:

5 Atualmente trabalha em qual empresa?

_____

6. Há 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
- ◯ Nunca atuei na área

7. Aproximadamente quantos funcionários em sua empresa atuam na área de TI?
Marcar apenas uma oval.

- ◯ Até 10 funcionários
- ◯ De 11 a 30 funcionários
- ◯ De 31 a 50 funcionários
- ◯ De 51 a 70 funcionários
- ◯ De 71 a 90 funcionários
- ◯ Mais de 100 funcionários
- ◯

_____

8. Aproximadamente quantos funcionários em sua empresa atuam no desenvolvimento de software?
Marcar apenas uma oval.

- Até 10 funcionários
- De 11 a 30 funcionários
- De 31 a 50 funcionários
- De 51 a 70 funcionários
- De 71 a 90 funcionários
- Mais de 100 funcionários
-

9. Qual é o seu cargo atual? *
Marcar apenas uma oval.

- Desenvolvedor Web
- Desenvolvedor Desktop
- Desenvolvedor Mobile
- Analista de Sistemas
- Arquiteto de Software
- Gerente de
- Projetos Outro:
-

10 _____ Você trabalha ou já trabalhou no desenvolvimento de softwares modulares utilizando a arquitetura MVC? * Marcar apenas uma oval.

- Sim
- Não Ir para a pergunta 35.

2. Desenvolvimento de software modular

As questões seguintes têm como objetivo conhecer um pouco sobre seu conhecimento acerca da utilização da modularização de software. Esta é um requisito não funcional que consiste em subdividir um sistema em subsistemas, ou seja, dividir em módulos, onde cada parte será responsável por suas determinadas funcionalidades e ações.

11. Qual o nível do seu conhecimento acerca do desenvolvimento de software modular? * Marcar apenas uma oval.

	0	1	2	3	4	5	
Nenhum							Muito Alto

12. Em sua opinião qual a importância da utilização da modularidade no desenvolvimento de um software? * Marcar apenas uma oval.

	0	1	2	3	4	5	
Nenhuma	◯	◯	◯	◯	◯	◯	Muito importante

13. Com que frequência você utiliza a modularização de software em seus projetos? *
Marcar apenas uma oval.

- ◯ Em todos os casos
- ◯ Na maioria dos casos
- ◯ Eventualmente
- ◯ Raramente
- ◯ Nunca
- ◯

14. O que levou à escolha do requisito não funcional de modularidade em seus projetos? * Marque todas que se aplicam.

- ☐ Reutilização de código
- ☐ Abstração das regras de negócio do software
- ☐ Melhor organização da aplicação
- ☐ Fácil aprendizado do sistema para novos desenvolvedores
- ☐ Divisão de partes da aplicação entre diferentes desenvolvedores
- ☐ Outro: _____

15 Para você o quanto a definição da modularização promove uma maior abstração do código e ajuda a reduzir a complexidade do desenvolvimento de um software? * Marcar apenas uma oval.

- ◯ Em todos os casos
- ◯ Na maioria dos casos
- ◯ Eventualmente
- ◯ Raramente
- ◯ Nunca
- ◯

16. Para você o quanto a definição da modularização promove uma melhor mutabilidade de um software e permite que partes sejam utilizadas em outras aplicações? * Marcar apenas uma oval.

- ◯ Em todos os casos
- ◯ Na maioria dos casos
- ◯ Eventualmente
- ◯ Raramente
- ◯ Nunca
- ◯

17. Para você qual a importância do acoplamento fraco para a redução de dependências entre as partes da aplicação? * Marcar apenas uma oval.

0 1 2 3 4 5

Nenhuma Importância ◯ ◯ ◯ ◯ ◯ ◯ Muito Importante

18. Para você o encapsulamento de componentes necessários em cada parte do sistema promove uma melhor manuseabilidade do software como um todo? * Marcar apenas uma oval.

- ◯ Em todos os casos
- ◯ Na maioria dos casos
- ◯ Eventualmente
- ◯ Raramente
- ◯ Nunca
- ◯

19. Para você a definição da modularidade no desenvolvimento de um software promove uma manutenção mais confiável, uma vez que fica claro qual determinada área da aplicação deverá ser alterada? * Marcar apenas uma oval.

- ◯ Em todos os casos
- ◯ Na maioria dos casos
- ◯ Eventualmente
- ◯ Raramente
- ◯ Nunca
- ◯

20 Com que frequência a utilização da modularidade possibilitou uma melhor estruturação de seus projetos? * Marcar apenas uma oval.

- ◯ Em todos os casos
- ◯ Na maioria dos casos
- ◯ Eventualmente
- ◯ Raramente
- ◯ Nunca
- ◯

21. Com que frequência em seus projetos a abstração do código possibilitou um melhor aprendizado acerca das regras do sistema a ser desenvolvido, diminuindo assim a complexidade no desenvolvimento? * Marcar apenas uma oval.

- ◯ Em todos os casos
- ◯ Na maioria dos casos
- ◯ Eventualmente
- ◯ Raramente
- ◯ Nunca
- ◯

22. Com que frequência em seus projetos a modularização promoveu a evolução do software com o menor impacto possível? * Marcar apenas uma oval.

- Em todos os casos
- Na maioria dos casos
- Eventualmente
- Raramente
- Nunca

3. Desenvolvimento de software em arquitetura MVC

As questões seguintes têm como objetivo conhecer um pouco sobre seu conhecimento acerca da utilização do padrão arquitetural MVC (Model-View-Controller). O padrão arquitetural MVC é muito utilizado em aplicações WEB, sendo responsável por dividir cada subsistema ou módulo em camadas de negócio (Model), controle do tráfego dos dados (Controller) e visualização da informação (View).

23. Qual o nível do seu conhecimento acerca do desenvolvimento de software utilizando a arquitetura MVC? *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Nenhum							Muito Alto

24 Com que frequência você utiliza o padrão arquitetural MVC em seus projetos? * Marcar apenas uma oval.

- Em todos os casos
- Na maioria dos casos
- Eventualmente
- Raramente
- Nunca

25. O que levou à escolha do padrão MVC em seus projetos? * Marque todas que se aplicam.

- Reutilização de código
- Abstração das regras de negócio do software
- Padronização de comunicação entre as camadas da aplicação
- Desacoplamento dos componentes das camadas facilitando sua manutenção
- Separação dos componentes de interface e dos componentes funcionais da aplicação
- Outro: _____

Insignificante	◯	◯	◯	◯	◯	◯	Muito Significante
Insignificante	◯	◯	◯	◯	◯	◯	Muito Significante

0 1 2 3 4 5

26. Para você o quanto a utilização do MVC promove a redução da complexidade arquitetural do desenvolvimento de um software? * Marcar apenas uma oval.

0 1 2 3 4 5

27. Para você o quanto a utilização do MVC auxilia em uma melhor manutenção de um software? *
Marcar apenas uma oval.

0 1 2 3 4 5

28. Para você, o quanto o aumento da complexidade (incluindo novos padrões e

metodologias) no processo de desenvolvimento, torna-se positivo para redução das dificuldades de compressão sobre as regras do software a ser desenvolvido? *
Marcar apenas uma oval.

Insignificante	◯	◯	◯	◯	◯	◯	Muito Significante

29 Para você o quanto o acoplamento fraco da aplicação promove uma melhor utilização dos recursos do padrão arquitetural MVC dentro de uma aplicação? *
Marcar apenas uma oval.

- Em todos os casos
- Na maioria dos casos
- Eventualmente
- Raramente
- Nunca

30. Para você a adição das camadas do MVC em uma aplicação modular supre todas as necessidades durante o desenvolvimento de um software? * Marcar apenas uma oval.

- Em todos os casos
- Na maioria dos casos
- Eventualmente
- Raramente
- Nunca

31. De acordo com seus conhecimentos acerca do MVC, qual a importância da separação do controle dos dados (camada Controller) com a visualização dos mesmos (camada View), onde permite que alterações lógicas sejam feitas sem afetar a parte de interação com o usuário? *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Nenhuma Importância							Muito Importante

32. Para você qual a importância da separação das regras de negócio na camada Model, para o tratamento de regras relacionadas ao software a ser desenvolvido? * Marcar apenas uma oval.

	0	1	2	3	4	5	
Nenhuma Importância							Muito Importante

33. Com que frequência a aplicação do MVC possibilitou um sistema melhor estruturado? * Marcar apenas uma oval.

- Em todos os casos
- Na maioria dos casos
- Eventualmente
- Raramente
- Nunca

34 Com que frequência em seus projetos o MVC promoveu a evolução do software com o menor impacto possível? * Marcar apenas uma oval.

- Em todos os casos
- Na maioria dos casos
- Eventualmente
- Raramente
- Nunca

Pare de preencher este formulário.

4. Experiência Profissional

As questões seguintes têm como objetivo conhecer um pouco sobre seu conhecimento profissional.

35. Você já trabalhou utilizando softwares que fazem uso da modularidade? *
Marcar apenas uma oval.

- Sim
- Não
- Não sei

36. Em sua empresa você ouve ou já ouviu falar de modularidade de software?
* Marcar apenas uma oval.

- Sim
- Não
- Não sei

37. Você já trabalhou utilizando softwares que fazem uso do padrão arquitetural MVC? * Marcar apenas uma oval.

- Sim
- Não
- Não sei

38. Em sua empresa você ouve ou já ouviu falar do padrão arquitetural MVC?
* Marcar apenas uma oval.

- Sim
- Não
- Não sei