IMPACTO DA ARQUITETURA DIRIGIDA A EVENTOS NA PERFORMANCE: UM ESTUDO EXPLORATÓRIO

Hebert Thiago Cabane[1]
Kleinner Farias[2]

Resumo:

Arquitetura dirigida a eventos tem sido amplamente adotada na indústria nos últimos anos. Esta adoção é motivada pelo aumento da modularidade e principalmente da performance de aplicações monolíticas decomposta. Embora tenha se popularizado, a literatura atual carece de estudos que demonstrem o impacto da arquitetura dirigida a eventos na performance. Sem este conhecimento, desenvolvedores e arquitetos acabam a adotando, considerando a recomendação de especialistas, ao contrário de seguir evidências empíricas. Este estudo, portanto, reporta um estudo empírico sobre o impacto da adoção de arquitetura dirigida a eventos na performance de uma aplicação. Para isso, a performance de uma aplicação implementada com a arquitetura baseada em eventos foi comparada com a performance da mesma aplicação implementada usando uma arquitetura monolítica. A comparação foi feita utilizando métricas, tais como uso de CPU, memória, tempo de resposta, vazão e total de pacotes enviados e recebidos. Os resultados, suportados por testes estatísticos, apontam que a arquitetura monolítica em comparação com a arquitetura dirigida a eventos, consome menos recursos computacionais, além de obter melhores tempos de respostas. Por fim, este estudo traz reflexões sobre a adoção de arquitetura dirigida a eventos, bem como aponta desafios e implicações que precisam ser consideradas pela comunidade científica em pesquisas futuras.

Palavras-chave: Arquitetura dirigida a eventos. Arquitetura monolítica.

1 INTRODUÇÃO

A decomposição vem sendo utilizado na indústria principalmente no contexto de refatorações de aplicações monolíticas para dar suporte à aplicações que sejam menos acopladas e que os comportamentos sejam compartilhados através de eventos. Como pode ser visto no trabalho apresentado por Laigner et al. (2020), que efetuaram a decomposição de uma aplicação monolítica devido à complexidade de manutenção e pelo custo necessário para manter a própria aplicação.

Diante dessa tendência da utilização da arquitetura dirigida a eventos, alguns trabalhos foram realizados ao longo dos últimos anos. Urdangarin, Farias e Barbosa (2021) investigam o uso de técnicas de decomposição baseadas em multi-objetivos e contexto. Laigner et al. (2020) apresentam os resultados obtidos durante a decomposição de uma aplicação big data monolítica em uma aplicação dirigida a eventos. Schipor, Vatavu e Vanderdonckt (2019) realizaram um

[1]Graduando em Sistemas de Informação pela Unisinos. Email: hebert.cabane@gmail.com

[2]Possui doutorado em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2012), mestrado em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (2008), graduação em Ciência da Computação pela Universidade Federal de Alagoas (2006) e em Tecnologia da Informação pelo Instituto Federal de Alagoas. Email: kleinnerfarias@unisinos.br

estudo empírico utilizando a arquitetura dirigida a eventos em aplicações de ambientes inteligentes. Tragatschnig, Stevanetic e Zdun (2018) realizaram um estudo empírico para avaliar o uso do change patterns no desenvolvimento de aplicações dirigida a eventos. Djogic, Ribic e Donko (2018) apresentaram os benefícios da reconstrução de uma aplicação monolítica em uma aplicação dirigida a eventos. Bukhsh, Sinderen e Singh (2015) realizaram um comparativo através estudos da literatura entre as arquiteturas orientada a serviços e dirigida a eventos. Pienwittayasakul e Liu (2014) apresentaram os benefícios da utilização da arquitetura dirigida a eventos em relação a arquitetura orientada a serviços.

Como pode ser visto, pouco tem sido investigado e analisado de forma comparativa entre as arquitetura dirigida a eventos e monolíticas através da perspectiva da performance com utilização de métricas de análise e resultados empíricos. Logo, desenvolvedores de software acabam adotando arquitetura dirigida a eventos sem o suporte de evidências empíricas, apenas confiando na opinião de especialistas (STOPFORD, 2018).

Este trabalho, portanto, apresenta uma análise comparativa, cujo objetivo é investigar os efeitos na performance em decorrência da utilização da arquitetura dirigida a eventos e da arquitetura monolítica. Para isso, foi realizado um estudo empírico exploratório que compara uma aplicação dirigida a eventos com uma aplicação monolítica. Para esta comparação foram utilizadas métricas para a coleta de dados, análises descritivas e estatística e análises dos resultados de cada aplicação. Isso foi feito através de um processo dividido em oito etapas organizados em quatro fases. Os resultados obtidos neste trabalho apontaram que arquitetura monolítica consume recursos computacionais de forma mais eficiente, além de possuir melhores tempos de respostas em comparação à arquitetura dirigida a eventos. Estes resultados beneficiam profissionais com interesse na área de Arquitetura de Software como Arquitetos de Software, Desenvolvedores de Software e Pesquisadores.

Este trabalho é organizado da seguinte forma: A Seção 2 apresenta a fundamentação teórica sobre os principais conceitos necessários para o entendimento deste estudo. Na Seção 3 é realizada uma análise comparativa de trabalhos relacionados. Na Seção 4 a metodologia do estudo é apresentada. A Seção 5 descreve a análise dos resultado obtidos. E por fim, a Seção 6 apresenta as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os principais conceitos essenciais para o entendimento do trabalho desenvolvido.

2.1 Decomposição de aplicações monolíticas

A decomposição é a divisão de uma aplicação com arquitetura monolítica em diversos módulos interdependentes resultando na composição de uma nova aplicação com arquitetura não

monolítica. Essencialmente ambas aplicações são equivalentes, possuindo as mesmas funcionalidades, porém são estruturalmente distintas (URDANGARIN; FARIAS; BARBOSA, 2021).

Em uma aplicação com arquitetura monolítica, todas as funcionalidades são disponibilizadas em uma única unidade. Normalmente, essa unidade é restrita à um único artefato de software executável. Geralmente, as aplicações monolíticas tradicionais são compostas por uma interface da Web, uma camada de domínio e uma camada de acesso à dados. Em uma arquitetura monolítica, essas camadas são combinadas em uma única instância da aplicação (URDANGARIN; FARIAS; BARBOSA, 2021).

Geralmente, as arquiteturas monolíticas são adequadas para aplicações pequenas, mas elas podem se tornar complexas ao mesmo tempo que a aplicação cresce. O que inicialmente era uma aplicação pequena e simples pode ser tornar uma aplicação complexa e difícil de manter, pois todas as funcionalidades estão contidas em uma única unidade. Diversos desafios são encontrados conforme o negócio e carga da aplicação aumentam, por exemplo: dificuldade de dimensionar funcionalidades existentes; complexidade de desenvolver novas funcionalidades; limitação no uso de tecnologias; compartilhamento do trabalho entre equipes; lançamento e implantação de novas versões. Esses desafios podem ser minimizados ao aplicar a decomposição.

2.2 Arquitetura dirigida a eventos

A arquitetura dirigida a eventos é um padrão arquitetural que é composto por componentes desacoplados com propósitos únicos que recebem, processam e transmitam eventos de forma assíncrona (RICHARDS, 2015). A arquitetura dirigida a eventos preconiza a comunicação dos seus componentes através do modelo produtor/consumidor, sendo que os produtores tem a responsabilidade de publicar eventos e os consumidores de assinar e consumir eventos (FALATIUK; SHIROKOPETLEVA; DUDAR, 2019).

Através da comunicação baseada em eventos, é estabelecido uma barreira que garante o baixo acoplamento e isolamento dos componente da arquitetura dirigida a eventos. Através do isolamento é possível gerenciar cada componente de forma independente, possibilitando controlar aspectos de controle de carga, elasticidade, monitoramento, entre outros aspectos de cada componente individualmente (BONER et al., 2014).

2.3 Performance

A performance é uma das características vitais da qualidade do software. Se aprimorarmos a performance da aplicação, serão obtidos efeitos positivos na qualidade do software, além de oferecer melhores experiência de uso. Dito isso, definimos performance como um indicador sobre a quantidade de respostas de uma aplicação executando suas funcionalidades em um determinado intervalo de tempo, e pela capacidade de consumir recursos como CPU e memória de forma apropriada durante o processamento das funcionalidade nas condições estabelecidas

(KAUR; GROVER; DIXIT, 2019). Entende-se como recursos o CPU (processador) e a memória (RAM). A Performance é composta pelas seguintes características: Comportamento em relação ao tempo, utilização de recursos e capacidade.

- **Comportamento em relação ao tempo:** Medidas dos tempos de resposta e processamento e taxas de transferência de uma aplicação durante a execução de uma funcionalidade.
- **Utilização de recursos:** Medidas das quantidades e tipos de recursos utilizados por uma aplicação durante a execução de uma funcionalidade.
- **Capacidade:** Limite máximo de uma aplicação executar uma funcionalidade.

Se o código fonte de uma aplicação for otimizado, terá bons efeitos no que se diz de performance, como a redução do tempo de resposta, aumento da quantidade de respostas, redução do uso de memória e do consumo de rede. (KAUR; GROVER; DIXIT, 2019).

A performance também é conhecida como eficiência e faz parte dos atributos de qualidades descritos pela ISO/IEC 25010 (ISO - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011), são eles: funcionalidade, eficiência, compatibilidade, usabilidade, confiabilidade, segurança, manutenção e portabilidade. Os atributos de qualidade fornecem uma base para medidas e formas de verificação para avaliação da qualidade do software. (PRESSMAN; MAXIM, 2016).

3 TRABALHOS RELACIONADOS

Esta seção apresenta uma análise dos trabalhos relacionados disponíveis na literatura, para isso foi feito uma seleção de trabalhos que abordam o uso da arquitetura monolítica e arquitetura dirigida a eventos. Os trabalhos selecionados são apresentados na Seção 3.1. A análise comparativa e oportunidades de pesquisa são discutidos na Seção 3.2.

3.1 Análise dos Trabalhos Relacionados

Laigner et al. (2020): apresentaram os resultados da reconstrução de uma aplicação big data de monitoramento de frota de caminhões de uma industria de óleo e gás. A arquitetura inicial da aplicação era do tipo monolítica e durante o projeto foi reconstruída utilizando a arquitetura de microsserviços dirigida a eventos. A reconstrução teve como objetivo de facilitar a manutenção da aplicação devido a complexidade e códigos obsoletos que a aplicação possuía. Com a arquitetura de microsserviços dirigida a eventos, foi percebido alguns resultados esperados, como a maior facilidade da manutenção dos serviços e na melhora do controle de falhas dos serviços, mas devido o complexo fluxo de dados gerado pela quantidade de microsserviços, foi percebido como desvantagens pelos autores.

Schipor, Vatavu e Vanderdonckt (2019): realizaram um estudo empírico utilizando a arquitetura de software EUPHORIA[3] desenvolvida pelos autores. Esta nova arquitetura dirigida a eventos tem a finalidade de orquestrar a comunicação entre diferentes dispositivos de um ambiente inteligente. O estudo foi conduzido em três cenários de ambientes inteligentes distintos utilizando a nova arquitetura dirigida a eventos, demonstrando na prática o uso da arquitetura em diferentes tipos de ambientes. Por fim, o resultado do estudo empírico demonstrou que o tempo de processamento de um evento pode ser impactado pelo tamanho das mensagens ou pela complexidade do ambiente inteligente, ou seja, pela quantidade de consumidores conectados.

Tragatschnig, Stevanetic e Zdun (2018): realizaram um estudo empírico sobre como o *change patterns* poderá influenciar a performance nas mudanças de uma aplicação com arquitetura dirigia a eventos. Para isso os autores selecionaram 4 aplicações baseadas em contexto reais da indústria e três conjuntos *change patterns*. Com isso, o experimento foi conduzido com 90 alunos divididos em três grupos, cada grupo ficou encarregado de executar um conjunto de *change patterns* específico nas aplicações previamente selecionadas. Com o resultado do experimento foi possível identificar que *change patterns* com menos operações primitivas exigem menos tempo em relação com os *change patterns* que possuem operação basicamente operações primitivas.

Djogic, Ribic e Donko (2018): apresentaram os benefícios em realizar a reconstrução de uma aplicação com arquitetura monolítica orientada a serviços em uma arquitetura de microsserviços dirigida a eventos, o trabalho foi realizado em uma aplicação real do setor imobiliário. A reconstrução da aplicação em uma nova arquitetura foi realizada devido ao aumento da quantidade de mensagens que são processadas pela aplicação e também para suportar novas integrações com a mesma. Por fim, os autores relataram que a reconstrução utilizando a arquitetura de microsserviços dirigida a eventos possibilitou na solução de diversos problemas que a aplicação monolítica possuíra, além de simplificar alguns processos como a manutenção individual dos componentes da aplicação, publicação somente do que foi alterado, escalabilidade por serviço e aproveitamentos do uso de recursos.

Bukhsh, Sinderen e Singh (2015): com base na literatura, os autores realizaram um estudo comparativo entre a arquitetura orientada a serviços e arquitetura dirigida a eventos, também foi considerado no estudo a arquitetura orientada a serviços dirigidos a eventos. O caso de estudo abordou a utilização de cada arquitetura estudada em uma Aplicação de Gestão de Aprendizagem (Moodle), com isso foi possível identificar quais são as desvantagens de cada arquitetura. Por fim o estudo concluiu que ambas arquiteturas possuem características distintas, principalmente como as comunicações entre os serviços são realizadas, e que a escolha da arquitetura dependerá do negócio da aplicação e como ela deverá reagir às certas situações, como: controle de transações, tipo de comunicação entre serviços, redundância de dados e respostas aos usuário e entre outros fatores.

Pienwittayasakul e Liu (2014): realizaram um estudo comparativo das definições e ca-

[3]http://www.eed.usv.ro/mintviz/resources/Euphoria/

racterísticas das arquiteturas orientada a serviços e dirigida a eventos, também foi analisado o relacionamento das arquiteturas entre si. A comparação entre as arquitetura foi realizada em quatro categorias: como as arquitetura reagem ao negócio da aplicação, como é distribuído cada serviço que compõem a aplicação, como é a comunicação de cada serviço da aplicação, como cada arquitetura gerencia a informação gerada pela aplicação. Por fim, os autores concluíram que a arquitetura orientada a serviços e a arquitetura dirigida a eventos interagem em diversas áreas, como na execução de eventos, onde são executadas simples ou complexas operações de negócio, e também que a arquitetura dirigia a eventos não é apenas uma implementação da arquitetura orientada a serviços, ambas arquiteturas são pares e complementos para o contexto de negócio e TI.

3.2 Análise Comparativa e Oportunidades de Pesquisa

A análise comparativa foi realizada com base em critérios, porque outros trabalhos já publicados (VIEIRA; FARIAS, 2020; RUBERT; FARIAS, 2021) e que utilizaram esta abordagem, mostraram que esta é uma forma efetiva para se gerar uma comparação entre trabalhos e identificar as oportunidades de pesquisa.

Com a realização da análise comparativa, foi possível identificar pontos que se assemelham e diferenciam entre o trabalho atual e os trabalhos relacionados selecionados. Essa comparação foi necessária para a identificação de oportunidades de pesquisa de forma objetiva, e para isso foram selecionados 6 Critérios de Comparação, que são os descritos a seguir:

- **Estudo Exploratório (CC01):** trabalhos que realizaram o estudo de forma exploratória.
- **Atributos de Performance (CC02):** estudos que consideram atributos de performance ou eficiência.
- **Métricas (CC03):** estudos que utilizaram métricas para analisar a eficiência do uso recursos e para avaliar a performance da aplicação.
- **Arquitetura Dirigida a Eventos (CC04):** estudos onde a arquitetura dirigida a eventos fazia parte do tema central de pesquisa.
- **Arquitetura Monolítica (CC05):** estudos onde a arquitetura monolítica foi comparada com outros tipos de arquiteturas.
- **Contexto da Aplicação (CC06):** neste critério são abordados os estudos que efetuaram sua pesquisa em aplicações reais.

O resultado da comparação dos trabalho relacionados e do trabalho proposto com base nos Critérios de Comparação é demonstrado na Tabela 1. Analisando o resultado da comparação, foram identificados as seguintes pontos de oportunidades:

Tabela 1: Análise comparativa dos trabalhos relacionados.

Trabalho Relacionado	Critério de Comparação					
	CC1	CC2	CC3	CC4	CC5	CC6
Trabalho proposto	●	●	●	●	●	●
Laigner et al. (2020)	○	○	○	●	●	●
Schipor, Vatavu e Vanderdonckt (2019)	●	◐	◐	●	○	○
Tragatschnig, Stevanetic e Zdun (2018)	●	○	○	●	○	◐
Djogic, Ribic e Donko (2018)	○	◐	○	●	●	●
Bukhsh, Sinderen e Singh (2015)	○	○	○	●	●	○
Pienwittayasakul e Liu (2014)	○	○	○	●	●	○

● Similar ◐ Parcialmente similar ○ Não similar

Fonte: elaborado pelo autor.

- apenas alguns trabalhos realizaram o estudo de forma empírica sobre a arquitetura dirigida a eventos;
- nenhum trabalho explorou os atributos de performance de aplicações que utilizam a arquitetura dirigida a eventos;
- nenhum trabalho explorou o uso de métricas para avaliar e quantificar o uso da arquitetura dirigida a eventos.

Oportunidade de Pesquisa. Com base nos pontos de oportunidades sinalizados, foi identificada a seguinte oportunidade de pesquisa: execução de estudos exploratórios de forma empírica sobre o impacto da utilização da arquitetura dirigida a eventos na performance das aplicações. Esta oportunidade de pesquisa é explorada nas próximas seções.

4 METODOLOGIA

Esta seção apresenta as decisões tomadas e a condução do estudo experimental e exploratório. Na Seção 4.1 é apresenta o objetivo e as questões de pesquisa. Na Seção 4.2 é introduz a formulação das hipóteses com base nas questões de pesquisa. Na Seção 4.3 é descreve o contexto e domínio da aplicação utilizada no estudo. Na Seção 4.4 é apresenta as variáveis quantitativas consideradas. Na Seção 4.5 é demonstra o processo de experimento. Por fim, na Seção 4.6 é apresenta os procedimentos de análise.

4.1 Objetivo e Questões de Pesquisa

Este estudo essencialmente procura avaliar os efeitos causados pela utilização da arquitetura dirigida a eventos e arquitetura monolítica na performance. Estes impactos serão investigados em aplicações reais de forma a gerar conhecimento empírico. O objetivo desse estudo é apresentado e baseado no modelo GQM, tal modelo mostrou-se eficaz na elaboração de objetivos, que pode ser visto nos seguintes trabalhos: (FARIAS; GARCIA; LUCENA, 2013) e (FARIAS et al., 2014). O resultado gerado pelo modelo é apresentado abaixo:

analisar *estilos arquiteturais*
com a finalidade de *investigar seus efeitos*
no que diz respeito a *performance*
do ponto de vista de *arquiteto de software*
no contexto *de desenvolvimento de software*

Particularmente, este estudo se concentra em avaliar o impacto na performance em decorrência da utilização da arquitetura dirigida a eventos e arquitetura monolítica. Com isso, o estudo se concentra na seguinte questão de pesquisa:

- **QP01:** Será que a performance da arquitetura dirigida a eventos é superior à performance da arquitetura monolítica?

4.2 Formulação da Hipótese

Especula-se que a utilização da arquitetura dirigida a eventos em relação à arquitetura monolítica traz benefícios no que se diz respeito à performance; uma vez que, a arquitetura dirigida a eventos preconiza a separação das funcionalidades modularizadas em subsistemas, diferentemente da arquitetura monolítica, que possui um único sistema (URDANGARIN; FARIAS; BARBOSA, 2021).

Portanto, acredita-se que a modularização pode trazer benefícios de performance, devido à eficiência relacionada à quantidade de recursos que uma aplicação consume e dos códigos necessários para a execução de uma operação em certa condições (KAUR; GROVER; DIXIT, 2019), acredita-se que a modularização pode consumir menos recursos e isso refletindo em uma melhor performance. Esta hipótese avalia se a performance da arquitetura dirigida a eventos é diferente da performance da arquitetura monolítica. A hipótese é descrita da seguinte forma:

> **Hipótese Nula 1, $\mathbf{H}_{1-0}$:** A performance da arquitetura dirigida a eventos é igual à performance da arquitetura monolítica.
> **$\mathbf{H}_{1-0}$:** Performance(arquitetura dirigida a eventos) = Performance(arquitetura monolítica)
> **Hipótese Alternativa 1, $\mathbf{H}_{1-1}$:** A performance da arquitetura dirigida a eventos é diferente da performance da arquitetura monolítica.
> **$\mathbf{H}_{1-1}$:** Performance(arquitetura dirigida a eventos) <> Performance(arquitetura monolítica)

A hipótese H1 foi refinada em um conjunto de 6 sub hipóteses, cada sub hipótese é representada por uma métrica de performance, tal qual é apresentada como uma variável dependente na Seção 4.4. A formulação das sub hipóteses poder ser visto na Tabela 2..

Ao realizar os testes de hipóteses, estarão sendo produzidos conhecimentos empíricos sobre os impactos na performance em recorrência do uso da arquitetura dirigida a eventos e da arquitetura monolítica.

Tabela 2: Hipóteses analisadas

Hipótese Nula		Hipótese Alternativa	
H_{1-0}	Performance(de) = Performance(mo)	H_{1-1}	Performance(de) <> Performance(mo)
H_{2-0}	Consumo de CPU(de) = Consumo de CPU(mo)	H_{2-1}	Consumo de CPU(de) <> Consumo de CPU(mo)
H_{3-0}	Memória RAM(de) = Memória RAM(mo)	H_{3-1}	Memória RAM(de) <> Memória RAM(mo)
H_{4-0}	Pacotes Recebidos(de) = Pacote Recebidos(mo)	H_{4-1}	Pacotes Recebidos(de) <> Pacotes Recebidos(mo)
H_{5-0}	Pacotes Enviados(de) = Pacotes Enviados(mo)	H_{5-1}	Pacotes Enviados(de) <> Pacotes Enviados(mo)
H_{6-0}	Tempo de resposta(de) = Tempo de resposta(mo)	H_{6-1}	Tempo de resposta(de) <> Tempo de resposta(mo)
H_{7-0}	Vazão(de) = Vazão(mo)	H_{7-1}	Vazão(de) <> Vazão(mo)

de Arquitetura Dirigida a eventos, *mo* Arquitetura Monolítica
Fonte: elaborado pelo autor.

4.3 Aplicação Alvo

Para a execução do estudo experimental e exploratório, foi definido que a variável independente (Seção 4.4) deverá possuir os seguintes possíveis valores: arquitetura dirigida a eventos e arquitetura monolítica. Então para cada arquitetura foi selecionada uma aplicação alvo para a execução de testes e coleta de dados de métricas selecionadas para a coleta de dados através de ferramentas de monitoramento. Os dados coletados serão analisados e submetidos à testes de hipóteses.

Cada aplicação selecionada essencialmente correspondem à mesma aplicação possuindo as mesmas funcionalidades desenvolvidas, se diferenciando apenas no estilo arquitetural, ou seja, a aplicação de arquitetura dirigida a eventos representa a decomposição da aplicação de arquitetura monolítica. A aplicação alvo possui as seguintes funcionalidades de um site de comercio eletrônico: catálogo de produtos, carrinho de compras, fechamento de pedidos, histórico de pedidos, cadastro de clientes e autenticação. As funcionalidades são descritas na Tabela 3.

Tabela 3: Funcionalidades da aplicação alvo.

Funcionalidade	**Descrição**
Catálogo de Produtos	Responsável em fornecer a relação de produtos disponíveis para a comercialização pela aplicação
Carrinho de Compras	Responsável por armazenar temporariamente os dados e quantidade de produtos selecionados para compras pelos clientes
Fechamento de Pedidos	Responsável pelo fechamento e faturamento dos Pedidos dos clientes
Histórico de Pedidos	Responsável em fornecer a relação dos pedidos efetuados pelos clientes
Cadastro de Clientes	Responsável em disponibilizar os recursos necessários para o cadastro de clientes na aplicação
Autenticação	Responsável em disponibilizar os recursos necessários para autenticar os clientes previamente cadastrados

Fonte: elaborado pelo autor.

As aplicações alvo foram desenvolvidas e atualnebte são mantidas pela organização *Dotnet Fundation*[4], tal grupo também é responsável por fomentar a plataforma *.Net*[5] na comunidade de desenvolvedores. O versionamento dos arquivos fontes das aplicações é feito na plataforma *GitHub*[6], sendo a aplicação monolítica localizada no repositório eShopOnWeb[7] e a aplicação dirigida a eventos é localizada no repositório eShopOnContainers[8]. Para o desenvolvimento das aplicações foi utilizada a linguagem de programação C# 9.0 e no framework .NET 5.0.

A Tabela 4 apresenta as características de cada aplicação no que se diz respeito ao tamanho e quantidade do código fonte gerado durante o desenvolvimento.

Tabela 4: Características da aplicação alvo

Arquitetura	**Linguagem**	**Framework**	**Arquivos**	**Linhas de código**	**Linhas de comentários**	**Linhas em branco**	**Total**
D.EV.	C# 9.0	.NET 5.0	379	16183	485	3333	20001
MONO	C# 9.0	.NET 5.0	158	5585	110	1129	6824

Fonte: elaborado pelo autor.

4.4 Variáveis do Estudo e Método de Quantificação

Variável independente. A variável independente da hipótese formulada é o tipo de arquitetura, que assume dois possíveis valores: "*arquitetura dirigida a eventos*"e "*arquitetura monolítica*". Cada valor da variável independente representa um estilo arquitetural utilizado pela aplicação alvo (ver Seção 4.3) e que serão submetidas à testes para a coleta de dados através da variável dependente de performance.

Variável dependente. A variável dependente da hipótese é a performance. A performance quantifica o consumo de recursos computacionais e dados referente ao tempo de processamento. A performance é composta pelas seguintes métricas: consumo de CPU, consumo de memória RAM, tempo de resposta, vazão ou respostas por minuto e tráfico de pacotes enviados e recebidos pela rede. As métricas de performance serão coletadas através de uma ferramenta de monitoramento de execução de aplicações, a ferramenta escolhida para o estudo foi a *New Relic*[9].

A Tabela 5 apresenta a descrição de cada métrica agrupada por atributos.

[4]https://dotnetfoundation.org
[5]https://dotnet.microsoft.com
[6]https://github.com/
[7]https://github.com/dotnet-architecture/eShopOnWeb
[8]https://github.com/dotnet-architecture/eShopOnContainers
[9]https://newrelic.com

Tabela 5: Métricas utilizadas para analisar QP01.

Atributos	**Métricas**	**Definições**
CPU	Utilização de CPU	Percentual de CPU (de todos os tipos, system, user, I/O), utilizado pelo processo.
Memória	Memória RAM	Total de memória RAM utilizada um processo.
Tempo	Tempo de resposta	Duração média da execução das transações do processo
	Vazão (Respostas por minuto)	Quantidade de transações por minuto do processo
Rede	Total de Pacotes Enviados	Total de pacotes de rede transmitidos pelo processo
	Total de Pacotes Recebidos	Total de pacotes de rede recebidos pelo processo

Fonte: elaborado pelo autor.

4.5 Processo Experimental

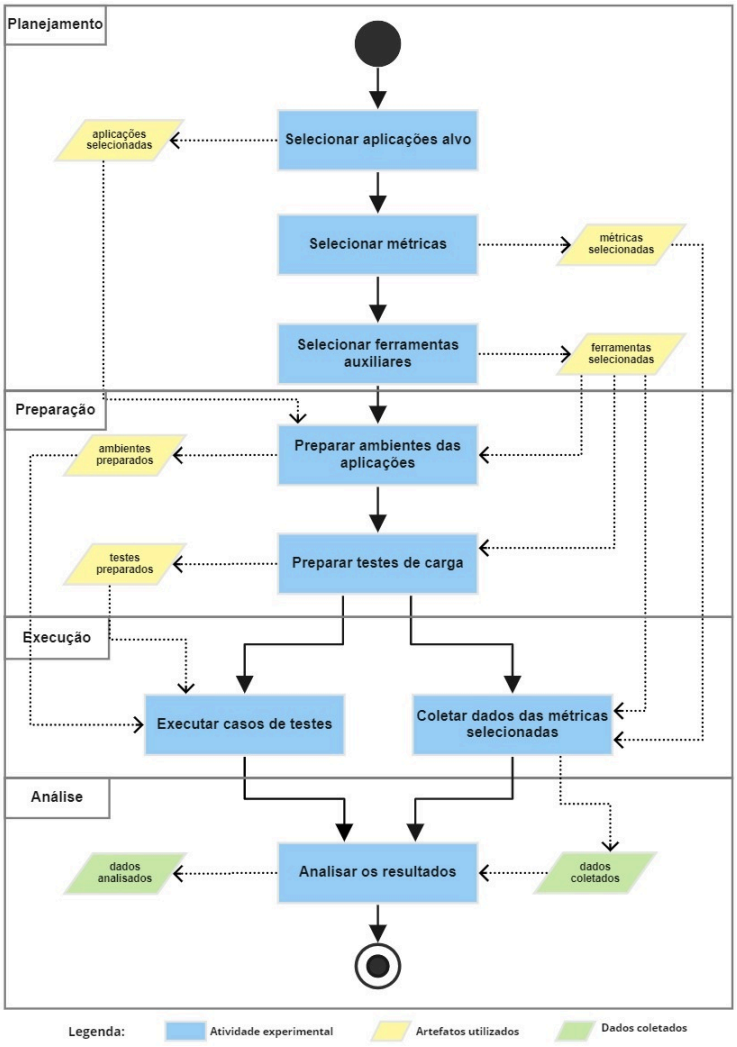
O Processo experimental deste estudo é dividido em oito etapas organizados em quatro fases (ver Figura 1). A fase de planejamento consiste nos passos de seleção de aplicações alvo, seleção de métricas e de ferramentas de apoio do processo experimental, como ferramentas de monitoramento e testes de carga; a fase de preparação consiste na configuração das ferramentas de coleta de dados para as métricas selecionadas e na configuração das ferramentas para os testes de carga das aplicações selecionadas; a fase de execução consiste nos passos de execução de testes de carga das aplicações e coleta de dados das métricas selecionadas; a fase de análise consiste nos passos de análise dos dados coletados. Os passos que compões o processo experimental são descritos a seguir.

Passo 1: Selecionar aplicações alvo. Este passo teve como objetivo selecionar as aplicações alvo para atender as duas arquiteturas definidas pela variável independente, ou seja, selecionar uma aplicação monolítica e uma aplicação dirigida a eventos que representa a decomposição da aplicação monolítica.

Passo 2: Selecionar métricas. Este passo teve como objetivo selecionar as métricas que possibilitem analisar os aspectos de performance de cada aplicação alvo selecionada. As métricas selecionadas corresponderam à variável dependente de performance.

Passo 3: Selecionar ferramentas auxiliares. Este passo teve como objetivo selecionar as ferramentas auxiliares para a execução dos cenários de testes e para a coleta de dados das métricas selecionadas. Para a coleta de dados das métricas selecionadas durante o processamento dos testes das aplicações, foi selecionado a ferramenta de monitoramento *New Relic*, pois além dos agentes que realizam a captura e coleta dos dados de processamento das aplicações alvo, a ferramenta também disponibiliza um portal para a visualização e monitoramento em tempo real das aplicações durante o processamento, além de armazenar os dados coletados para futuras análises. Para a execução dos casos de testes em cada aplicação, teve a necessidade em selecionar uma ferramenta para a realização de testes de carga em aplicações de interface web,

Figura 1: Processo experimental



Fonte: elaborado pelo autor

para isso foi selecionado a ferramenta *Apache JMeter*[10].

Passo 4: Preparar ambientes das aplicações. Neste passo teve a finalidade de efetuar a instalação e configuração das aplicações alvo selecionadas no ambiente de testes. Para a instalação e configuração de cada aplicação foi necessário efetuar a compilação de cada aplicação para a geração dos programas executáveis.

Para a aplicação monolítica foi compilado e gerado apenas um programa executável responsável pelo processamento da própria aplicação, também foi necessário a configuração do banco de dados *MS Sql Server*[11], tal qual é necessário para o armazenamento das informações da aplicação. Para a instalação e execução do banco de dados foi utilizado contêineres através da plataforma *Docker*[12].

Para a aplicação dirigida a eventos foi compilado e gerado 9 programas executáveis respon-

[10]https://jmeter.apache.org
[11]https://www.microsoft.com/pt-br/sql-server/
[12]https://www.docker.com

sáveis pelo processamento dos módulos da aplicação. Também foi necessário a configuração dos bancos de dados *MS Sql Server* e *Redis*[13], o primeiro para a persistências definitiva dos dados e o segundo para o armazenamento temporário de dados. Para o tráfico e distribuição das mensagens emitidas pelos módulos da aplicação, foi configurado o gerenciador de mensagens *RabbitMq*[14]. Para a execução dos bancos de dados e do gerenciador de mensagens também foi utilizado o recurso de contêineres utilizando a plataforma *Docker*.

Também neste passo, foram instalados e configurados os agentes da ferramenta de monitoramento para cada aplicação alvo preparada.

Passo 5: Preparar testes de carga. Este passo teve a finalidade de elaborar os cenários de testes de carga para cada aplicação alvo na ferramente de teste *Apache JMeter*. Como a aplicação dirigida a eventos é uma versão decomposta da aplicação monolítica, os cenários de testes de cada aplicação apresentaram essencialmente as mesmas etapas, são elas: 1) acessar a página inicial; 2) efetuar login no site de comércio eletrônico; 3) acessar o catálogo de produtos; 4) adicionar três produtos ao carrinho de compras; 5) acessar o carinho de compras; 6) confirmar o carinho de compras; 7) confirmar dados de pagamento; 8) visualizar histórico de pedidos; 9) efetuar logout no site de comércio eletrônico.

Conforme apresentado na Tabela 6, foi preparado três execuções para cada cenário de teste, cada execução do cenário de teste simula o acesso simultâneo de 10, 25 e 50 usuários, desta forma poderá ser observado o comportamento de cada aplicação ao reagir com o aumento do processamento simultâneos.

Tabela 6: Cenários de teste de carga

Cenário	Descrição
1	Teste de carga com 10 Usuários simultâneos
2	Teste de carga com 25 Usuários simultâneos
3	Teste de carga com 50 Usuários simultâneos

Fonte: elaborado pelo autor.

Passo 6: Executar casos de testes. Neste passo teve como objetivo executar os casos de testes de cada aplicação alvo selecionada. Com a execução dos casos de testes será possível coletar dados referente à execução da aplicação pelas ferramentas de monitoramento. A execução dos casos de testes ocorreu de forma manual e automatizada através da ferramenta de teste *Apache JMeter*.

Passo 7: Coletar dados das métricas selecionadas. Neste passo teve a finalidade de coletar os dados capturados pelas ferramenta de monitoramento *New Relic* durante a execução dos casos de testes. Para cada execução de teste de carga, foi coletado dados mas métricas selecionadas agrupadas por métrica e cenário de teste.

Passo 8: Analisar os resultados. Neste passo teve como objetivos em analisar os dados de forma descritiva e identificar sua distribuição, executar os testes das hipóteses utilizando os

[13]https://redis.io
[14]https://www.rabbitmq.com

dados coletados. Por fim apresentar a discussão dos resultados de acordo com a perspectiva das variáveis dependentes.

4.6 Procedimento de Análise

Análise descritiva. Foi realizada a análise descritiva para analisar a distribuição, dispersões, tendências como médias e medianas dos dados coletados de cada métricas selecionada para cada tipo de arquitetura definida pela variável independente.

Análise estatística. Realizamos a análise estatística para o teste de hipóteses. O nível de significância para os testes das hipóteses foi $\alpha = 0{,}05$. As análises estatística foram realizadas para testar as hipóteses de cada métrica selecionada e em cada cenário de teste executado. Para testar a hipótese H1 e suas sub hipóteses, foi aplicado o Teste não pareado Mann-Whitney para cada conjunto de dados coletados pelas métricas selecionadas. Como resultado, cada métrica será analisada e comparada entre os tipos de arquitetura e para cada cenário de teste executado. Também foi aplicado o Teste de Shapiro-Wilk para auxiliar na análise de distribuição dos dados e na escolha do teste estatístico.

5 RESULTADOS

Esta seção analisa os dados obtidos pelo processo experimental descrito na Seção 4. As descobertas são derivadas de processamentos numéricos dos dados coletados e representações gráficas dos aspetos dos resultados obtidos. Na seção 5.1 é apresentado a análise descritiva dos dados coletados. Na seção 5.2 apresenta os dados obtidos através dos testes de hipóteses.

5.1 Estatística descritiva

Esta seção descreve os aspectos dos dados coletados respectivos à performance dos tipos arquiteturais estudadas neste trabalho. Para isto, foi aplicado a análise descritiva para analisar a distribuição dos dados, tendências (médias, medianas e etc) e as dispersões dos conjuntos de dados através do desvio padrão.

Os dados estatísticos foram calculados com base em 120 composições para cada métrica selecionada (ver Tabela 5), ou seja, com 60 composições aplicadas à arquitetura dirigida a eventos e 60 composições aplicadas à arquitetura monolítica. Cada composição representa um minuto numa linha de tempo contínua de 60 minutos. Cada métrica analisada possui um conjunto de 60 composições de dados para cada cenário de teste de carga e para cada tipo de arquitetura.

Ao realizar a análise do resultado da estatística descritiva, foi possível observar que a arquitetura monolítica possuiu mais efeitos positivos sobre a arquitetura dirigida a eventos do que a arquitetura dirigida a eventos sobre a arquitetura monolítica, pois a arquitetura monolítica

apresentou melhores valores em relação ao consumo de recursos e em relação aos tempos de processamentos durante o teste empírico.

Este resultado é sustentado pelas seguintes observações para cada métrica (ver também a Figura 2):

Percentual de CPU. As médias de uso de CPU para cada cenário da arquitetura monolítica foram de 9,11%, 24,81% e 38,14% e para a arquitetura dirigida a eventos foram de 2,93%, 7,94% e 17,54%. Com esses valores é possível visualizar que o uso de CPU pela arquitetura monolítica foi entre duas e três vezes maior que a arquitetura dirigida a eventos, mesmo considerando o desvio padrão de cada cenário.

Tabela 7: Estatística Descritiva - Percentual de CPU

Cenário	Arquit.	N	Mínimo	25	Média	75	Máximo	Mediana	DP
1	D.EV	60	1,47	2,90	2,96	4,43	13,10	4,33	2,67
	MONO	60	4,90	8,14	9,11	9,95	14,90	9,33	2,40
2	D.EV	60	5,20	6,62	7,94	9,19	11,70	7,98	1,52
	MONO	60	21,20	22,88	24,81	26,60	35,00	25,13	3,09
3	D.EV	60	3,90	15,89	17,54	18,52	59,30	17,78	5,93
	MONO	60	32,20	36,48	38,14	39,95	48,60	38,65	2,92

D.EV Arquitetura Dirigida a eventos, *MONO* Arquitetura Monolítica, *DP* Desvio Padrão

Memória RAM. As médias de consumo de memória Ram para cada cenário da arquitetura monolítica foram de 347,05mb, 349,95mb e 344,57mb e para a arquitetura dirigida a eventos foram de 1723,03mb, 1721,55mb e 1788,81mb. Com esses valores é possível constatar que a arquitetura dirigida a eventos consumiu uma maior quantidade de memória RAM que a arquitetura monolítica, a diferença de consumo de memória foi de cinco vezes maior.

Tabela 8: Estatística Descritiva - Memória RAM

Cenário	Arquit.	N	Mínimo	25	Média	75	Máximo	Mediana	DP
1	D.EV	60	1656,24	1683,07	1723,03	1889,70	1941,85	1772,41	104,02
	MONO	60	335,99	340,78	347,05	391,19	684,29	378,96	67,12
2	D.EV	60	1672,69	1701,61	1721,55	1732,82	1750,75	1717,59	21,01
	MONO	60	338,26	346,04	349,95	352,48	392,29	351,79	9,86
3	D.EV	60	1726,06	1764,84	1788,81	1856,11	2049,38	1811,47	74,82
	MONO	60	328,51	339,12	344,57	348,19	357,42	343,85	6,15

D.EV Arquitetura Dirigida a eventos, *MONO* Arquitetura Monolítica, *DP* Desvio Padrão

Tempo de Resposta. As médias do tempo de resposta para cada cenário da arquitetura monolítica foram de 13,31, 13,13 e 14,54 milissegundos e para a arquitetura dirigida a eventos as médias foram de 43,08, 16,68 e 16,13 milissegundos. Através destes valores é possível notar que as médias são muito próximas entre os tipos de arquiteturas, mesmo considerando o desvio padrão os valores médios se mantém muitos próximos sem grandes diferenças entre si.

Vazão. A média de respostas por minuto em cada cenário observado da arquitetura monolítica foram de 801,5, 1592 e 2399 requisições por minuto e para a arquitetura dirigida a eventos obteve as médias 2350, 5161 e 5713 requisições por minuto, através destes valores, é possível observar que a arquitetura dirigida a eventos obteve as maiores médias, isso indica que a arquitetura dirigida eventos gerou uma quantidade superior de conexões entre cliente e servidor para

Tabela 9: Estatística Descritiva - Tempo de resposta (ms)

Cenário	Arquit.	N	Mínimo	25	Média	75	Máximo	Mediana	DP
1	D.EV	60	18,51	35,83	43,08	43,08	49,33	37,67	10,86
	MONO	60	13,20	13,20	13,31	14,51	14,51	13,84	0,66
2	D.EV	60	16,22	16,68	16,68	16,68	17,09	16,68	0,08
	MONO	60	13,13	13,13	13,13	13,16	13,16	13,14	0,02
3	D.EV	60	16,13	16,13	16,13	22,80	22,80	18,71	3,25
	MONO	60	14,54	14,54	14,54	14,54	15,74	14,56	0,15

D.EV Arquitetura Dirigida a eventos, *MONO* Arquitetura Monolítica, *DP* Desvio Padrão

atender processamento das funcionalidades da aplicação alvo.

Tabela 10: Estatística Descritiva - Vazão (rpm)

Cenário	Arquit.	N	Mínimo	25	Média	75	Máximo	Mediana	DP
1	D.EV	60	2350,00	2350,00	2350,00	2501,00	2505,00	2398,87	71,01
	MONO	60	789,00	789,00	801,50	803,00	803,00	796,18	7,02
2	D.EV	60	5161,00	5161,00	5161,00	5161,00	5637,00	5169,97	61,84
	MONO	60	1592,00	1592,00	1592,00	1611,00	1611,00	1598,63	9,12
3	D.EV	60	3638,00	3638,00	5713,00	5713,00	5713,00	4908,40	1012,55
	MONO	60	1859,00	2399,00	2399,00	2399,00	2399,00	2390,00	69,71

D.EV Arquitetura Dirigida a eventos, *MONO* Arquitetura Monolítica, *DP* Desvio Padrão

Pacotes recebidos por segundo. A média de pacotes recebidos por segundo em cada cenário de teste da arquitetura monolítica foram de 30,89, 60,70 e 86,17 de pacotes por segundo e para a arquitetura dirigida a eventos foram de 77,09, 152,01, 224,54 pacotes por segundo, através destes valores é possível observar que a arquitetura dirigida a eventos obteve as maiores médias e também percebe-se que a quantidade de pacotes recebidos está diretamente relacionado a quantidade de requisições por minutos (Vazão).

Tabela 11: Estatística Descritiva - Pacotes recebidos por segundo

Cenário	Arquit.	N	Mínimo	25	Média	75	Máximo	Mediana	DP
1	D.EV	60	67,75	73,62	77,09	80,25	83,68	76,78	4,12
	MONO	60	27,22	30,01	30,89	32,58	36,67	31,19	2,13
2	D.EV	60	141,50	148,06	152,01	158,08	175,38	152,82	6,46
	MONO	60	56,44	59,31	60,70	62,24	65,80	60,82	2,31
3	D.EV	60	162,36	217,58	224,54	230,95	479,72	228,44	35,09
	MONO	60	80,55	84,17	86,04	88,49	93,49	86,24	2,91

D.EV Arquitetura Dirigida a eventos, *MONO* Arquitetura Monolítica, *DP* Desvio Padrão

Pacotes enviados por segundo. A média de pacotes enviados por segundo em cada cenário de teste da arquitetura monolítica foram de 21,48, 42,28 e 60,98 de pacotes por segundo e para a arquitetura dirigida a eventos foram de 53,69, 104,60 e 153,66 pacotes por segundo, através destes valores, assim como a média de pacotes recebidos por segundo é possível observar que a arquitetura dirigida a eventos obteve as maiores médias, e também percebe-se que a quantidade de pacotes enviados também está diretamente a quantidade de requisições por minutos (Vazão).

Por fim, é observado os efeitos da utilização da arquitetura monolítica e dirigida a eventos através de dois aspectos: consumo de recursos computacionais e em relação ao tempo e quantidade de respostas. O detalhamento de cada aspecto observado durante a estatística descritiva é apresentado a seguir.

Tabela 12: Estatística Descritiva - Pacotes enviados por segundo

Cenário	Arquit.	N	Mínimo	25	Média	75	Máximo	Mediana	DP
1	D.EV	60	47,46	51,47	53,69	55,69	58,19	53,45	2,70
	MONO	60	18,93	20,63	21,48	22,36	24,72	21,53	1,33
2	D.EV	60	97,29	101,91	104,60	108,12	123,16	105,10	4,57
	MONO	60	39,53	41,27	42,28	43,41	45,92	42,31	1,53
3	D.EV	60	104,27	149,54	153,66	157,76	310,19	155,93	21,99
	MONO	60	57,29	59,80	60,98	62,49	65,23	61,10	1,91

D.EV Arquitetura Dirigida a eventos, *MONO* Arquitetura Monolítica, *DP* Desvio Padrão

Aspecto 1: Consumo de recursos computacionais. Sobre o consumo de recursos, a média do consumo de CPU foi maior na arquitetura monolítica do que a arquitetura dirigida a eventos, porém esta diferença é menor em comparação da média de consumo de memória que é inferior na arquitetura monolítica em relação a arquitetura dirigida a eventos. Os resultados apresentaram o consumo de CPU duas vezes e meia maior na arquitetura monolítica e para o consumo de memória cinco vezes maior na arquitetura dirigida a eventos. Através desde aspecto, é identificado que a arquitetura monolítica consome menos recursos em comparação à arquitetura dirigida a eventos.

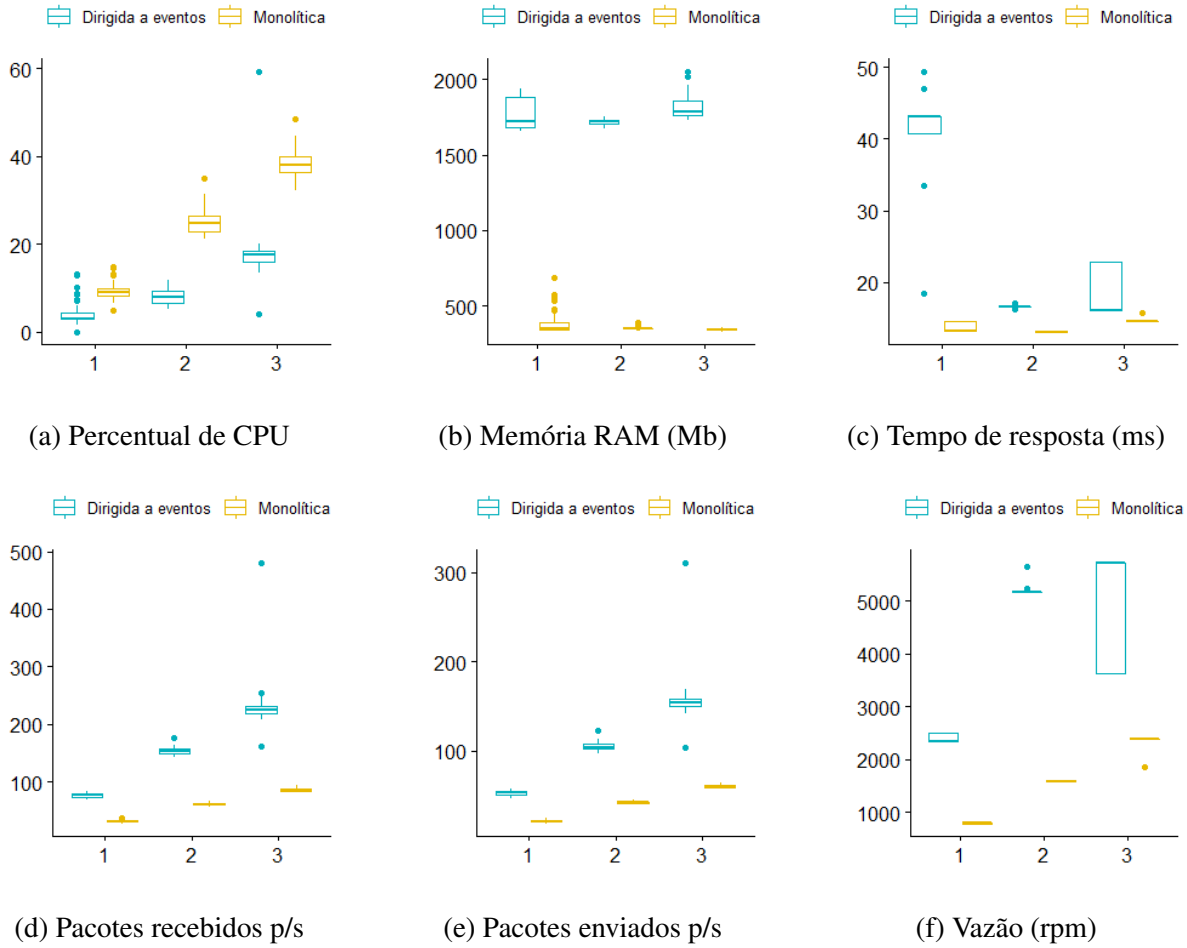
Aspecto 2: Tempo e quantidade de respostas. Sobre o desempenho, as médias do tempo de resposta da arquitetura monolítica foram superiores as médias da arquitetura dirigida a eventos, porém a diferença das médias em comparação de uma arquitetura com a outra apresentam valores muito próximos, entretanto, a dispersão dos tempos de resposta da arquitetura dirigida a eventos foi maior. Sobre a vazão, a média de respostas por minuto na arquitetura dirigida a eventos foi superior que média da arquitetura monolítica, e assim como a média de tempo de resposta, a arquitetura dirigida a eventos possui a dispersão dos dados maior do que a arquitetura monolítica. Dito isto, foi identificado que a arquitetura monolítica apresentou melhores tempos de respostas em comparação a arquitetura dirigida a eventos.

5.2 Teste de Hipóteses

Foram realizados testes estatísticos com os dados coletadas através das métricas de performance para avaliar se os mesmos são estatisticamente significantes. Para isto, foi hipotetizado que a arquitetura dirigida a eventos possui a performance superior que a arquitetura monolítica, ou seja, consome menos recursos computacionais e possui melhores tempos de respostas. Para testar as diferenças entre as médias das métricas de performance foram executados testes unilaterais, considerando o nível de significância em 0,05 (p value $<= 0{,}05$), como mencionado anteriormente.

Foi aplicado o teste de Shapiro-Wilk para avaliar a distribuição dos dados coletados e os mesmos apresentaram uma distribuição diferente da normal (p value $<= 0{,}05$) para todos os conjuntos de dados dos cenários de testes executados. Então, devido a distribuição dos dados, não foi possível aplicar o teste estatístico T-Teste, neste caso, o teste não paramétrico Mann-Whitney foi utilizado como o teste estatístico principal.

Figura 2: Diagrama de box-plot.



(a) Percentual de CPU (b) Memória RAM (Mb) (c) Tempo de resposta (ms)

(d) Pacotes recebidos p/s (e) Pacotes enviados p/s (f) Vazão (rpm)

Foi testado a hipótese H1 e suas sub-hipóteses (ver Tabela 2) para avaliar a QP1 nos cenários de testes (ver Tabela 6) executados. A Tabela 13 contém os valores da comparação independente entre a arquitetura dirigida a eventos e a arquitetura monolítica em cada cenário de teste. Os valores p em negrito representam resultados estatisticamente significantes, isto é, p *value* $<$ 0,05, estes valores indicam a rejeição da respectiva hipótese nula.

A principal descoberta é a comprovação que a arquitetura monolítica obteve maiores valores médios de processamento de CPU que a arquitetura dirigida a eventos, porém em contraponto também foi comprovado que a arquitetura dirigida a eventos obteve maiores valores de consumo de mais memória que a arquitetura monolítica, sendo a diferença muito superior do que a diferença do consumo de CPU.

Também foi comprovado que a arquitetura monolítica obteve melhores médias de tempo de respostas em relação a arquitetura dirigida a eventos, entretanto a arquitetura dirigida a eventos obteve melhores médias na vazão de respostas por minutos, porém a arquitetura dirigida a eventos obteve valores médios de recebimento e envios de pacotes de dados superiores, e isto pode comprovar o aumento da vazão de respostas por minutos, pois a arquitetura dirigida a eventos necessita de um barramento virtual para comunicação a comunicação dos módulos.

Tabela 13: Teste Estatístico Mann-Whitney

Cenário	Estatísticas	CPU	Memória RAM	Tempo de resposta	Vazão (Throughput)	Pacotes recebidos	Pacotes enviados
1	*U'*	3.282	3.600	3.600	3.600	3.600	3.600
	U	318	0	0	0	0	0
	Z	7,776	9,445	9,445	9,445	9,445	9,445
	p-value	**0,001**	**0,001**	**0,001**	**0,001**	**0,001**	**0,001**
	Rank sum D.EV	2148	5430	5430	5430	5430	5430
	Rank sum MONO	5112	1830	1830	1830	1830	1830
	Mean Rank D.EV	35,8	90,5	90,5	90,5	90,5	90,5
	Mean Rank MONO	85,2	30,5	30,5	30,5	30,5	30,5
2	*U'*	3.600	3.600	3.600	3.600	3.600	3.600
	U	0	0	0	0	0	0
	Z	9,445	9,445	9,445	9,445	9,445	9,445
	p-value	**0,001**	**0,001**	**0,001**	**0,001**	**0,001**	**0,001**
	Rank sum D.EV	1830	5430	5430	5430	5430	5430
	Rank sum MONO	5430	1830	1830	1830	1830	1830
	Mean Rank D.EV	30,5	90,5	90,5	90,5	90,5	90,5
	Mean Rank MONO	90,5	30,5	30,5	30,5	30,5	30,5
3	*U'*	3.540	3.600	3.600	3.600	3.600	3.600
	U	60	0	0	0	0	0
	Z	9,130	9,445	9,445	9,445	9,445	9,445
	p-value	**0,001**	**0,001**	**0,001**	**0,001**	**0,001**	**0,001**
	Rank sum D.EV	1890	5430	5430	5430	5430	5430
	Rank sum MONO	5370	1830	1830	1830	1830	1830
	Mean Rank D.EV	31,5	90,5	90,5	90,5	90,5	90,5
	Mean Rank MONO	89,5	30,5	30,5	30,5	30,5	30,5

D.EV Arquitetura Dirigida a eventos, *MONO* Arquitetura Monolítica
Fonte: elaborado pelo autor.

Desta forma, é rejeitado a hipótese nula e este resultado é sustentado pelas seguintes observações:

Percentual de CPU. Foi hipotetizado que a arquitetura dirigida a eventos possui um consumo de processamento de CPU superior ou igual do que a arquitetura monolítica, porém ao analisar os resultados do teste estatístico, é possível observar que o *p-value* é inferior à *z* e inferior ao nível de significância 0,05. Portanto, a hipótese nula de nenhuma diferença de consumo de CPU pela arquitetura dirigida a eventos e arquitetura monolítica pode ser rejeitada. Ou seja, há evidências suficientes para comprovar que as diferenças de consumo de CPU entre as arquitetura dirigida a eventos e arquitetura monolítica são estatisticamente diferentes. A Tabela 13 retrata o rank médio de consumo de CPU da arquitetura monolítica é maior que a arquitetura dirigida a eventos.

> O teste estatístico *Mann-Whitney U* foi executado para determinar se havia diferenças no consumo de CPU entre uma aplicação de arquitetura dirigida a eventos e outra aplicação com arquitetura monolítica em três cenário de testes. As distribuições dos valores de consumo de CPU para as arquiteturas dirigida a eventos e monolítica foram semelhantes, conforme avaliado por inspeção visual. O consumo médio de CPU foi estatisticamente significativamente maior na aplicação de arquitetura monolítica para todos os três cenários de testes (9,33, 25,13 e 38,65) do que na aplicação de arquitetura dirigida a eventos (4,33, 7,98 e 17,78), U = 3282, 3600 e 3540, z = 7,776, 9,445 e 9,130, p = 0,001, 0,001 e 0,001.

Memória RAM. Foi hipotetizado que a arquitetura dirigida a eventos possui um consumo de Memória RAM superior ou igual do que a arquitetura monolítica, porém ao analisar os resultados do teste estatístico, é possível observar que que o *p-value* é inferior à *z* e inferior ao nível

de significância 0,05. Portanto, a hipótese nula de nenhuma diferença de consumo de Memória RAM pela arquitetura dirigida a eventos e arquitetura monolítica pode ser rejeitada. Ou seja, há evidências suficientes para comprovar que as diferenças de consumo de Memória RAM entre as arquitetura dirigida a eventos e arquitetura monolítica são estatisticamente diferentes. A Tabela 13 retrata o rank médio de consumo de Memória RAM da arquitetura dirigida a eventos é maior que a arquitetura monolítica.

O teste estatístico *Mann-Whitney U* foi executado para determinar se havia diferenças no consumo de Memória RAM entre uma aplicação de arquitetura dirigida a eventos e outra aplicação com arquitetura monolítica em três cenário de testes. As distribuições dos valores de consumo de Memória RAM para as arquiteturas dirigida a eventos e monolítica foram semelhantes, conforme avaliado por inspeção visual. O consumo médio de Memória RAM foi estatisticamente significativamente maior na aplicação de arquitetura dirigida a eventos para todos os três cenários de testes (1772,41, 1717,59 e 1811,47) do que na aplicação de arquitetura monolítica (378,96, 351,79 e 343,85), U = 3600, 3600 e 3600, z = 9,445, 9,445 e 9,445, p = 0,001, 0,001 e 0,001.

Tempo de resposta. Foi hipotetizado que a arquitetura dirigida a eventos possui o Tempo de Resposta inferior ou igual do que a arquitetura monolítica, porém ao analisar os resultados do teste estatístico, é possível observar que que o *p-value* é inferior à *z* e inferior ao nível de significância 0,05. Portanto, a hipótese nula de nenhuma diferença do Tempo de Resposta pela arquitetura dirigida a eventos e arquitetura monolítica pode ser rejeitada. Ou seja, há evidências suficientes para comprovar que as diferenças de Tempo de Resposta entre as arquitetura dirigida a eventos e arquitetura monolítica são estatisticamente diferentes. A Tabela 13 retrata o rank médio de Tempo de Resposta da arquitetura dirigida a eventos é maior que a arquitetura monolítica.

O teste estatístico *Mann-Whitney U* foi executado para determinar se havia diferenças no Tempo de Respostas entre uma aplicação de arquitetura dirigida a eventos e outra aplicação com arquitetura monolítica em três cenário de testes. As distribuições dos valores dos Tempos de Respostas para as arquiteturas dirigida a eventos e monolítica foram semelhantes, conforme avaliado por inspeção visual. O Tempo de Resposta médio foi estatisticamente significativamente maior na aplicação de arquitetura dirigida a eventos para todos os três cenários de testes (37,67, 16,68 e 18,71) do que na aplicação de arquitetura monolítica (13,84, 13,14 e 14,56), U = 3600, 3600 e 3600, z = 9,445, 9,445 e 9,445, p = 0,001, 0,001 e 0,001.

Vazão. Foi hipotetizado que a arquitetura dirigida a eventos possui uma taxa de vazão de repostas por minutos superior ou igual do que a arquitetura monolítica, porém ao analisar os resultados do teste estatístico, é possível observar que que o *p-value* é inferior à *z* e inferior ao nível de significância 0,05. Portanto, a hipótese nula de nenhuma diferença da Vazão pela arquitetura dirigida a eventos e arquitetura monolítica pode ser rejeitada. Ou seja, há evidências suficientes para comprovar que as diferenças de Vazão entre as arquitetura dirigida a eventos e arquitetura monolítica são estatisticamente diferentes. A Tabela 13 retrata o rank médio da taxa de vazão da arquitetura dirigida a eventos é maior que a arquitetura monolítica.

O teste estatístico *Mann-Whitney U* foi executado para determinar se havia diferenças na Vazão de Respostas por Minuto entre uma aplicação de arquitetura dirigida a eventos e outra aplicação com arquitetura monolítica em três cenário de testes. As distribuições dos valores das Vazões de Respostas por Minuto para as arquiteturas dirigida a eventos e monolítica foram semelhantes, conforme avaliado por inspeção visual. A Vazão de Respostas por Minuto média foi estatisticamente significativamente maior na aplicação de arquitetura dirigida a eventos para todos os três cenários de testes (2398,87, 5169,97 e 4908,40) do que na aplicação de arquitetura monolítica (796,18, 1598,63 e 2390,00), U = 3600, 3600 e 3600, z = 9,445, 9,445 e 9,445, p = 0,001, 0,001 e 0,001.

Pacotes recebidos por segundo. Foi hipotetizado que a arquitetura dirigida a eventos possui uma frequência de recebimento de pacotes por segundo superior ou igual do que a arquitetura monolítica, porém ao analisar os resultados do teste estatístico, é possível observar que que o *p-value* é inferior à *z* e inferior ao nível de significância 0,05. Portanto, a hipótese nula de nenhuma diferença de Pacotes recebidos por segundo pela arquitetura dirigida a eventos e arquitetura monolítica pode ser rejeitada. Ou seja, há evidências suficientes para comprovar que as diferenças da quantidade de Pacotes recebidos por segundo entre as arquitetura dirigida a eventos e arquitetura monolítica são estatisticamente diferentes. A Tabela 13 retrata o rank médio da taxa de vazão da arquitetura dirigida a eventos é maior que a arquitetura monolítica.

O teste estatístico *Mann-Whitney U* foi executado para determinar se havia diferenças no Recebimento de Pacotes por Segundo entre uma aplicação de arquitetura dirigida a eventos e outra aplicação com arquitetura monolítica em três cenário de testes. As distribuições dos valores de Pacotes Recebidos por Segundo para as arquiteturas dirigida a eventos e monolítica foram semelhantes, conforme avaliado por inspeção visual. A média de Pacotes Recebidos por Segundo foi estatisticamente significativamente maior na aplicação de arquitetura dirigida a eventos para todos os três cenários de testes (76,78, 152,82 e 228,44) do que na aplicação de arquitetura monolítica (31,19, 60,82 e 86,24), U = 3600, 3600 e 3600, z = 9,445, 9,445 e 9,445, p = 0,001, 0,001 e 0,001.

Pacotes enviados por segundo. Foi hipotetizado que a arquitetura dirigida a eventos possui uma frequência de envio de pacotes por segundo superior ou igual do que a arquitetura monolítica, porém ao analisar os resultados do teste estatístico, é possível observar que que o *p-value* é inferior à *z* e inferior ao nível de significância 0,05. Portanto, a hipótese nula de nenhuma diferença de Pacotes enviados por segundo pela arquitetura dirigida a eventos e arquitetura monolítica pode ser rejeitada. Ou seja, há evidências suficientes para comprovar que as diferenças da quantidade de Pacotes enviados por segundo entre as arquitetura dirigida a eventos e arquitetura monolítica são estatisticamente diferentes. A Tabela 13 retrata o rank médio da taxa de vazão da arquitetura dirigida a eventos é maior que a arquitetura monolítica.

O teste estatístico *Mann-Whitney U* foi executado para determinar se havia diferenças no Envio de Pacotes por Segundo entre uma aplicação de arquitetura dirigida a eventos e outra aplicação com arquitetura monolítica em três cenário de testes. As distribuições dos valores de Pacotes Enviados por Segundo para as arquiteturas dirigida a eventos e monolítica foram semelhantes, conforme avaliado por inspeção visual. A média de Pacotes Enviados por Segundo foi estatisticamente significativamente maior na aplicação de arquitetura dirigida a eventos para todos os três cenários de testes (53,45, 105,10 e 155,93) do que na aplicação de arquitetura monolítica (21,53, 42,31 e 61,10), U = 3600, 3600 e 3600, z = 9,445, 9,445 e 9,445, p = 0,001, 0,001 e 0,001.

5.3 Discussão

Baseado nas observações dos resultados obtidos pela análise descritiva e pelos testes de hipóteses, identificados os seguintes assuntos para discussão:

Consumo de CPU. A arquitetura monolítica apresentou maior consumo de CPU do que a arquitetura dirigida a eventos, conforme a intensidade dos testes de carga foi aumentando, foi possível observar o aumento do consumo de CPU em ambas as arquiteturas, porém a arquitetura monolítica se manteve com o consumo superior em todos os cenários de testes. A aplicação monolítica possui apenas um módulo responsável pelo processamento de todas as funcionalidades da aplicação, diferente da arquitetura dirigida a eventos que possui diversos módulos, cada um responsável pelo processamento de uma funcionalidade ou parte de uma funcionalidade, então especulamos que devido à esta característica da arquitetura monolítica, há uma sobrecarga de processamento pela aplicação causando o alto consumo de CPU. Este pode ser um fato importante na contribuição das tomadas de decisões de reconstruções de aplicações monolíticas, pois em aplicações reais o alto consumo poderá ser limitador do processamento e reduzindo a performance da aplicação. Estes resultados podem contribuir juntamente com os estudos comparativos como os trabalhos realizados por Pienwittayasakul e Liu (2014) e Bukhsh, Sinderen e Singh (2015), pois estes trabalhos realizaram com base na literatura um levantamento de vantagens e desvantagem da utilização de cada arquitetura, porém sem evidência empírica.

Consumo de Memória. A arquitetura dirigida a eventos apresentou maior consumo de Memória RAM em comparação com a arquitetura monolítica e obtendo os maiores valores de consumo em todos os cenários de testes aplicados. Diferente do consumo de CPU, os valores de consumo de memória RAM não apresentaram um aumento muito significativo conforme o aumento da carga durante os testes executados. A aplicação com arquitetura dirigida a eventos possui 9 sub aplicações para o processamento de todas as funcionalidade da aplicação alvo (ver seção 4.3), diferente da arquitetura monolítica que é composta por apenas uma aplicação. Especulamos que o aumento do consumo memória RAM pela arquitetura dirigida a eventos é devido a quantidade de objetos necessários para a execução de cada sub aplicação, como vimos na Tabela 4 a aplicação dirigida eventos possui uma quantidade maior de arquivos ou classes implementadas que a aplicação monolítica. Também devemos considerar que cada aplicação possui um consumo de memória para alocação dos recursos necessários para inicialização e

execução da própria aplicação, se supomos que para iniciar uma aplicação são necessários 50mb de memória RAM, então as 9 sub aplicação da aplicação dirigida a eventos tem um consumo mínimo de 450mb, sendo um valor superior ao consumo total de memória RAM da aplicação monolítica.

Tempo de resposta. A arquitetura monolítica obteve menores tempos de resposta que a arquitetura dirigida a eventos, ou seja, obteve os melhores valores e devido a isto acaba sendo mais performático. O que difere entre as arquiteturas é a forma de comunicação entre os módulos da aplicação, na arquitetura monolítica todos os módulos estão na mesma aplicação, diferente da arquitetura monolítica, onde os módulos são sub aplicações que se comunicam através de um barramento. Acredita-se que este barramento de comunicação entre os módulos da arquitetura dirigida a eventos pode impactar no tempo de respostas da aplicação, porém para confirmar esta afirmação, será necessário aplicar novos testes e analisar novas métricas que ajudem a evidência esse comportamento.

Vazão. A arquitetura dirigida a eventos obteve melhores valores de vazão de respostas por minuto do que a arquitetura monolítica. Apesar da aplicação de arquitetura dirigida a eventos ter obtido valores de tempo de respostas menos performático, a mesma apresentou valores superiores referentes a quantidade de respostas por minuto do que a aplicação de arquitetura monolítica. Um indício sobre o aumento da quantidade de respostas por minuto é a métrica também estar contabilizando as respostas da comunicação entre os módulos da aplicação dirigida a eventos e não somente as respostas que foram enviadas para o cliente.

5.4 Desafios e Implicações

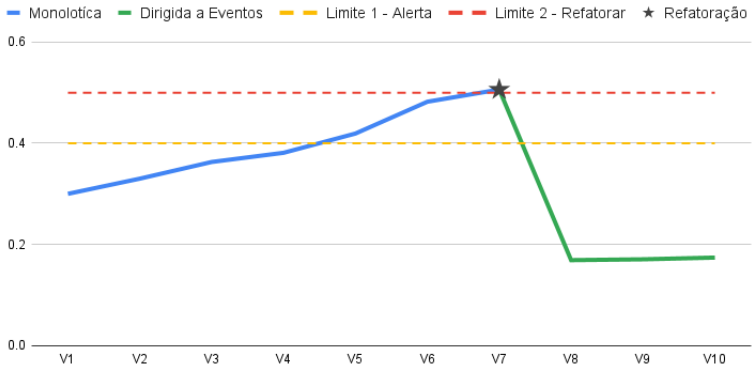
Este seção apresenta os desafios e implicações que foram derivados a partir da análise dos resultados dos dados coletados.

Desafio 1: Efeitos da modularização na performance. A modularização é manifestação da separação de interesses, ou seja, é a divisão ou organização do software em módulos que são integrados em entre si para atender uma ou mais funcionalidades (BUCCHIARONE et al., 2018). Dito isto, em uma futura pesquisa sugere-se avaliar os efeitos causados pela modulação na performance das arquiteturas dirigidas a eventos e monolítica, pois a comunicação entre os módulos da arquitetura dirigida a eventos é feita através de um barramento de eventos, diferentemente da arquitetura monolítica, que , a modularização pode ser medida através de alguns grupos de métricas, são elas: separação de interesses, acoplamento, coesão e tamanho (GARCIA et al., 2006).

Desafio 2: Analisar performance durante o desenvolvimento. Diferente dos testes e dados coletados executados para este trabalho, no qual cada aplicação alvo selecionada possui todas as funcionalidades já implementas, recomenda-se aplicar testes semelhantes para a coleta de dados de performance em cada versão da aplicação dirigida a eventos e monolítica, pois dessa forma será possível acompanhar e analisar o uso de recursos durante o ciclo de vida da

aplicação.

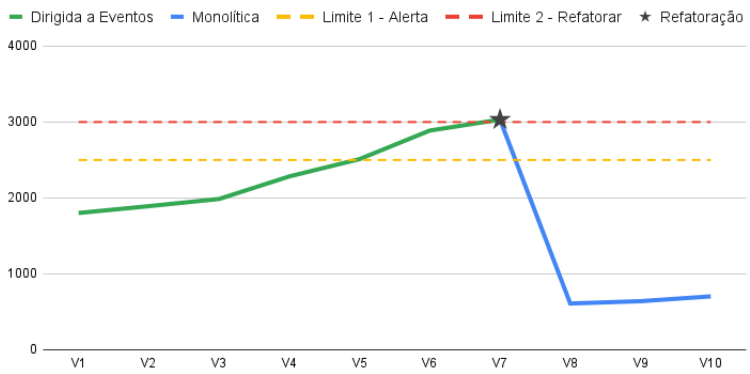
Figura 3: Implicação CPU



Fonte: elaborado pelo autor

Implicação 1: Indicadores de refatoração ou decomposição. Os resultados implicam em buscar fatores que possibilitam indicar o momento que uma aplicação monolítica deverá ser decomposta em uma aplicação dirigida a eventos, ou vice-versa. Como visto anteriormente, a arquitetura monolítica consume uma porcentagem maior de CPU em comparação a arquitetura dirigida a eventos, então especulamos que este pode ser um fator para ser avaliado durante a decisão de decomposição. Para a memória RAM, que diferentemente do consumo de CPU, na arquitetura dirigida o consumo é maior em comparação com a arquitetura monolítica, também é especulado que este pode ser outro fator para a decisão da refatoração da arquitetura dirigida a eventos em arquitetura monolítica.

Figura 4: Implicação Memória



Fonte: elaborado pelo autor

Para exemplificar o uso de indicadores de refatoração ou decomposição, a Figura 3 e a Figura 4 demonstram o ciclo evolutivo de uma determinada aplicação através de versões lançadas ao longo do tempo. Para cada nova versão da aplicação, serão aplicados testes exploratórios para avaliar as mudanças do consumo de CPU e memória RAM em decorrência as alterações

aplicadas durante o desenvolvimento. Os resultados obtidos serão analisados e comparados com indicadores que alertam sobre o aumento do consumo de recursos ou sobre a necessidade de refatoração da aplicação devido ao excesso consumo.

6 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou um estudo empírico para avaliar a performance da arquitetura dirigida e da arquitetura monolítica. Durante o experimento empírico uma aplicação com arquitetura dirigida a eventos e outra com arquitetura monolítica foram submetidas à testes para a coleta de dados através de métricas previamente selecionadas. Foram analisados o consumo de CPU e memória RAM, quantidade e tempos de resposta e a quantidade de pacotes transferidos e recebidos pela rede de cada aplicação durante os testes para a observação dos efeitos de cada arquitetura. Analisando os resultados, pode-se verificar em quais aspectos cada arquitetura possui maior performance.

A partir dos resultados dos dados obtidos neste estudo, observou-se que aplicações com arquitetura monolíticas possuem baixo consumo de recursos como o CPU e memória RAM em comparação com aplicações de arquitetura dirigida a eventos, porém a aplicação de arquitetura dirigida a eventos possuem melhores quantidades e tempos de respostas. Através desse resultados, este estudo contribui com conhecimentos empíricos para auxiliar a tomada de decisão da escolha da arquitetura adequada para o desenvolvimento do software e também na decomposição de aplicações monolíticas em aplicações dirigida a eventos com evidências sobre a execução de uma aplicação utilizando duas arquitetura distintas.

Para tornar este trabalho ter uma abordagem mais completa, alguns trabalhos futuros poderão serão realizados: (1) considerar novas métricas para avaliar outros aspectos relacionados a performance; (2) considerar métricas de modularização para analisar possíveis efeitos na performance; (3) coletar dados de novas aplicações de outros contextos e/ou linguagens e frameworks utilizados no desenvolvimento. Este trabalho é um estudo inicial dos efeitos na performance causados pela arquitetura dirigida a eventos, servindo de suporte para novos estudos mais aprofundados relacionados tema.

REFERÊNCIAS

BONER, J. et al. **The reactive manifesto**. Disponível em: <https://www.reactivemanifesto.org>. Acesso em: 18 novembro 2021.

BUCCHIARONE, A. et al. From monolithic to microservices: an experience report from the banking domain. **IEEE Software (Volume: 35, Issue: 3, May/June 2018)**, [S.l.], p. 50–55, June 2018.

BUKHSH, Z. A.; SINDEREN, M. van; SINGH, P. M. Soa and eda: a comparative study: similarities, differences and conceptual guidelines on their usage. **2015 12th International**

Joint Conference on e-Business and Telecommunications (ICETE), Colmar, France, July 2015.

DJOGIC, E.; RIBIC, S.; DONKO, D. Monolithic to microservices redesign of event driven integration platform. **MIPRO 2018, May 21-25, 2018, Opatija Croatia**, [S.l.], p. 1411–1414, May 2018.

FALATIUK, H.; SHIROKOPETLEVA, M.; DUDAR, Z. Investigation of architecture and technology stack for e-archive system. **2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC ST)**, [S.l.], October 2019.

FARIAS, K. et al. Evaluating the effort of composing design models: a controlled experiment. **Softw Syst Model (2015) 14:1349–1365**, [S.l.], p. 14:1349—-1365, May 2014.

FARIAS, K.; GARCIA, A.; LUCENA, C. Effects of stability on model composition effort: an exploratory study. **Softw Syst Model (2014) 13:1473–1494**, [S.l.], p. 13:1473—-1494, Jan. 2013.

GARCIA, A. et al. Modularizing design patterns with aspects: a quantitative study. **Rashid A., Aksit M. (eds) Transactions on Aspect-Oriented Software Development I. Lecture Notes in Computer Science, vol 3880**, [S.l.], p. 36–74, 2006.

ISO - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 25010**:2011: systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models. 2011.

KAUR, A.; GROVER, P. S.; DIXIT, A. Performance efficiency assessment for software systems. **Hoda M., Chauhan N., Quadri S., Srivastava P. (eds) Software Engineering. Advances in Intelligent Systems and Computing, vol 731**, Singapore, p. 83–92, 2019.

LAIGNER, R. et al. From a monolithic big data system to a microservices event-driven architecture. **2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**, New Portoroz, Slovenia, p. 213–220, Aug. 2020.

PIENWITTAYASAKUL, C.; LIU, Y. Comparative study on service-oriented architecture and event-driven architecture. **Proceedings of the International conference on Computing Technology and Information Management, Dubai, UAE, 2014**, Islamic Azad University, UAE Branch, Academic City Campus, Dubai, UAE, p. 397–405, Apr. 2014.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH Editora Ltda, 2016.

RICHARDS, M. **Software architecture patterns**. [S.l.]: O'Reilly Media, Inc., 2015.

RUBERT, M.; FARIAS, K. On the effects of continuous delivery on code quality: a case study in industry. **Computer Standards and Interfaces**, [S.l.], p. 103588, 2021.

SCHIPOR, O.-A.; VATAVU, R.-D.; VANDERDONCKT, J. Euphoria: a scalable, event-driven architecture for designing interactions across heterogeneous devices in smart environments. **Information and Software Technology Volume 109, May 2019, Pages 43-59**, [S.l.], p. 43–59, May 2019.

STOPFORD, B. **Designing event-driven systems**. 1. ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., 2018.

TRAGATSCHNIG, S.; STEVANETIC, S.; ZDUN, U. Supporting the evolution of event-driven service-oriented architectures using change patterns. **Information and Software Technology Volume 100, August 2018, Pages 133-146**, [S.l.], p. 133–146, Aug. 2018.

URDANGARIN, R. G.; FARIAS, K.; BARBOSA, J. Mon4aware: a multi-objective and context-aware approach to decompose monolithic applications. **XVII Brazilian Symposium on Information Systems (SBSI 2021), June 7–10, 2021, Uberlândia, Brazil**, [S.l.], June 2021.

VIEIRA, R. D.; FARIAS, K. Usage of psychophysiological data as an improvement in the context of software engineering: a systematic mapping study. **SBSI'20: XVI Brazilian Symposium on Information Systems**, [S.l.], p. 1–8, Nov. 2020.