UNIVERSIDADE DO VALE DO TAQUARI
CURSO DE ENGENHARIA DE SOFTWARE

# OTIMIZAÇÃO DE APLICAÇÕES WEB PHP: UMA INVESTIGAÇÃO COMPARATIVA ENTRE LARAVEL OCTANE COM SWOOLE E LARAVEL CONVENCIONAL

Guilherme Keunecke Bangemann

Lajeado/RS, novembro de 2023

Guilherme Keunecke Bangemann

# OTIMIZAÇÃO DE APLICAÇÕES WEB PHP: UMA INVESTIGAÇÃO COMPARATIVA ENTRE LARAVEL OCTANE COM SWOOLE E LARAVEL CONVENCIONAL

Projeto de Monografia apresentado na disciplina de Trabalho de Conclusão de Curso II, do curso de Engenharia de Software, da Universidade do Vale do Taquari - Univates, como parte da exigência para a obtenção do título de bacharel em Engenharia de Software.

Orientadora(o): Prof. Dr. Vinícius Meyer.

Lajeado/RS, novembro de 2023

Guilherme Keunecke Bangemann

# OTIMIZAÇÃO DE APLICAÇÕES WEB PHP: UMA INVESTIGAÇÃO COMPARATIVA ENTRE LARAVEL OCTANE COM SWOOLE E LARAVEL CONVENCIONAL

A Banca examinadora abaixo aprova a Monografia apresentada no componente curricular Trabalho de Conclusão de Curso II, do Curso de Engenharia de Software, da Universidade do Vale do Taquari - Univates, como parte da exigência para a obtenção do título de Bacharel em Engenharia de Software:

Profº. Dr. Vinícius Meyer - Orientador
Universidade do Vale do Taquari – Univates

Profº. Dr. Alexandre Stürmer Wolf
Universidade do Vale do Taquari – Univates

Profª. Dra. Maria Claudete Schorr
Universidade do Vale do Taquari - Univates

Lajeado/RS, 13 de dezembro de 2023

## RESUMO

Aprimorar o desempenho das aplicações web é fundamental para garantir uma experiência de usuário de alta qualidade, impulsionando a eficiência e a velocidade das plataformas online. Este estudo propõe uma avaliação abrangente em relação ao desenvolvimento de aplicações web PHP, com foco na otimização, realizando um estudo comparativo entre Laravel Octane com Swoole e Laravel. A motivação para esta pesquisa surgiu da necessidade de entender qual dessas ferramentas oferece a melhor performance na criação de projetos. Para atingir esse objetivo, foram desenvolvidas duas aplicações idênticas em termos de funcionalidade, uma utilizando Laravel Octane com Swoole e a outra baseada apenas no Laravel convencional. O estudo se concentrou na comparação detalhada do desempenho dessas duas abordagens, considerando métricas como tempo de resposta, escalabilidade e consumo de recursos do servidor. O estudo revelou que o Laravel Octane com Swoole é uma solução eficaz para melhorar o desempenho e a escalabilidade em aplicações web PHP, apesar de um maior consumo de recursos do servidor em comparação com o Laravel Convencional. Isso ressalta a necessidade de uma escolha cuidadosa da tecnologia de otimização, considerando as especificidades da aplicação e os recursos disponíveis.



**Palavras-chave:** Aplicações Web, Otimização de aplicações web, Laravel, Swoole, Programação assíncrona.

## ABSTRACT

Enhancing the performance of web applications is crucial to ensure a high-quality user experience, driving efficiency and speed in online platforms. This study proposes a comprehensive evaluation regarding the development of PHP web applications, with a focus on optimization, conducting a comparative study between Laravel Octane with Swoole and conventional Laravel. The motivation for this research arose from the need to understand which of these tools offers the best performance in project creation. To achieve this goal, two identical applications in terms of functionality were developed, one using Laravel Octane with Swoole and the other based on conventional Laravel only. The study focused on a detailed comparison of the performance of these two approaches, considering metrics such as response time, scalability, and server resource consumption. The study revealed that Laravel Octane with Swoole is an effective solution for enhancing performance and scalability in PHP web applications, despite higher server resource consumption compared to Conventional Laravel. This underscores the need for careful technology selection in optimization, considering the specifics of the application and available resources.



**Keywords:** Web Applications, Web Application Optimization, Laravel, Swoole, Asynchronous Programming.

# LISTA DE FIGURAS

# LISTA DE QUADROS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| APIs - | Application Programming Interfaces |
| CDN - | Content Delivery Network |
| CPU - | Central Processing Unit |
| CSS - | Cascading Style Sheets |
| C++ - | C Plus Plus |
| DNS - | Domain Name System |
| Gzip - | GNU zip |
| HTML - | HyperText Markup Language |
| HTTP - | HyperText Transfer Protocol |
| I/O - | Input/Output |
| MVC - | Model-View-Controller |
| PHP - | Hypertext Preprocessor |
| PHP-FPM - | PHP FastCGI Process Manager |
| REST - | Representational State Transfer |
| SMS - | Short Message Service |
| SOAP - | Simple Object Access Protocol |
| SPA - | Single Page Application |
| SQL - | Structured Query Language |
| XML - | eXtensible Markup Language |

# SUMÁRIO

# 1 INTRODUÇÃO

Com o avanço rápido da Internet, a expectativa por uma experiência aprimorada nos sites cresce entre os usuários. Para aperfeiçoar a utilização das páginas e proporcionar uma experiência de interação mais envolvente, as capacidades das páginas web têm aumentado significativamente, incorporando extensos códigos e tecnologias de frameworks. Essa abordagem, embora torne as páginas mais expressivas e ricas, impacta diretamente na eficiência de carregamento no navegador, resultando em atrasos. Diante desse cenário, a otimização web torna-se essencial para melhorar a experiência do usuário, afirmam os autores Cao, Shi e Li (2017).

Segundo Yao e Xia (2016), em um ambiente de alta concorrência, o desempenho de um sistema de aplicação web frequentemente não atende aos requisitos de design e negócios. Embora o amplo uso da publicação de informações e coleta de dados pela internet seja essencial para a gestão da indústria, o aumento do interesse social em informações populares e a alta concorrência resultante de milhares de acessos por segundo podem levar o sistema de aplicação web a funcionar lentamente. Em casos mais críticos, o sistema pode até colapsar e ficar inoperante. Diante desses desafios, há uma necessidade de otimizar o desempenho dos sistemas de aplicação web existentes, garantindo sua operação normal em condições de acesso concorrente elevado.

Nesse contexto, este estudo se configura como uma pesquisa experimental e aplicada, com o objetivo de realizar uma análise comparativa entre as estratégias de desenvolvimento web, especialmente no que diz respeito ao desempenho, entre Laravel convencional e Laravel Octane com Swoole. O foco da pesquisa está centrado

na otimização de aplicações web PHP, buscando identificar qual dessas abordagens proporciona uma performance superior na criação de projetos.

A pesquisa visa desenvolver uma compreensão aprofundada do uso dessas tecnologias, utilizando métricas como tempo de resposta, escalabilidade e consumo de recursos do servidor como validação. Ao comparar diretamente o Laravel Octane com Swoole ao Laravel convencional, pretende-se fornecer informações e resultados que auxiliam na escolha da abordagem mais adequada para otimizar o desempenho de aplicações web PHP.

O estudo se destaca pela sua abordagem prática, proporcionando uma visão mais tangível das implicações dessas tecnologias na eficiência das aplicações web. A relevância deste trabalho reside na contribuição para a tomada de decisões informadas no campo do desenvolvimento web, visando aprimorar a experiência do usuário e a eficácia das aplicações online.

## 1.1 Tema

O tema descrito na pesquisa consiste em investigar, analisar e comparar o uso entre Laravel Octane com Swoole e Laravel convencional, no contexto de otimização de aplicações web PHP.

## 1.2 Delimitação do tema

Delimita-se à pesquisa a realizar uma investigação comparativa entre o Laravel Octane com Swoole e o Laravel convencional, com foco específico na melhoria do desempenho, escalabilidade, consumo de recursos e tempo de resposta em aplicações web PHP. Esta pesquisa visa aprofundar-se nas implicações dessas ferramentas, analisando como contribuem para otimizar o desenvolvimento em ambientes web baseados em PHP.

## 1.3 Problema de pesquisa

Esta pesquisa tem o objetivo de responder a seguinte pergunta: Como o uso de tecnologias como Laravel Octane e Swoole se comparam às abordagens convencionais no desenvolvimento de aplicações web PHP utilizando o framework Laravel, com foco na otimização levando em consideração o desempenho, a escalabilidade, o consumo de recursos e o tempo de resposta?

## 1.4 Hipótese

Acredita-se que a implementação do Laravel Octane em conjunto com o Swoole resultará em melhorias significativas no desempenho, na escalabilidade, no consumo de recursos e no tempo de resposta em comparação com a abordagem convencional do Laravel.

## 1.6 Objetivo geral

O objetivo geral da pesquisa é investigar o uso de tecnologias em favor da otimização de aplicações web PHP, por meio da análise comparativa entre Laravel Octane com Swoole e Laravel convencional, visando aprimorar o desempenho, escalabilidade, eficiência no consumo de recursos e tempo de resposta, em cenários de testes simulados.

## 1.7 Objetivos específicos

- **Avaliar o Desempenho:** Comparar o desempenho das aplicações web desenvolvidas com Laravel Octane e Swoole em relação ao Laravel convencional, medindo métricas como tempo de resposta e carga de servidor;

- **Analisar a Escalabilidade:** Investigar a capacidade de escalabilidade das diferentes configurações de aplicações, considerando o número de requisições simultâneas;

- **Medir o Consumo de Recursos:** Avaliar o consumo de recursos do servidor, incluindo o uso de CPU e memória, nas configurações de Laravel Octane, Swoole e Laravel convencional;

## 1.8 Justificativa

A otimização de aplicações web PHP é uma prioridade constante para desenvolvedores e empresas que buscam aprimorar a qualidade da experiência do usuário. À medida que o uso de aplicações web continua a crescer, a demanda por soluções que ofereçam alta velocidade e eficiência se torna ainda mais crítica.

Compreender quais configurações oferecem o melhor desempenho é fundamental para atender às expectativas dos usuários, bem como para economizar recursos e custos operacionais.

Nesse contexto, o presente estudo se justifica pela curiosidade do pesquisador em explorar soluções que aprimorem o desempenho e a eficiência de aplicações web desenvolvidas em PHP. As experiências prévias do pesquisador em relação ao desenvolvimento de projetos web PHP também motivaram a investigação aprofundada dessas ferramentas, buscando compreender como elas podem impactar positivamente a qualidade e eficiência das aplicações, criando uma necessidade de avaliar e comparar o desempenho do uso do Laravel Octane com Swoole e o Laravel sem qualquer outra extensão (Laravel convencional).

## 1.9 Estrutura

A estrutura deste trabalho está organizada em seis capítulos. O capítulo 1 introduz o tema, fornecendo uma visão geral do assunto abordado na pesquisa. No capítulo 2, é apresentado o referencial teórico, destacando as bases conceituais e tecnológicas essenciais para o desenvolvimento do trabalho. O capítulo 3 trata dos trabalhos relacionados, explorando pesquisas e projetos que têm relevância para o contexto da presente investigação.

No capítulo 4, são detalhados os materiais e métodos utilizados, esclarecendo as razões por trás da escolha das tecnologias e delineando o processo de desenvolvimento adotado. As análises e discussões dos resultados obtidos são apresentadas no capítulo 5, oferecendo uma compreensão aprofundada dos dados coletados. E por fim, o capítulo 6 engloba as conclusões do estudo, as considerações finais sobre os resultados alcançados e aponta possíveis direções para trabalhos futuros, encerrando de maneira abrangente a pesquisa.

# 2 FUNDAMENTAÇÃO TEÓRICA

Para a condução deste estudo sobre a otimização de aplicações web PHP, é essencial estabelecer uma base teórica sólida que permita compreender os conceitos fundamentais e as tecnologias relevantes para o tema em questão. Este capítulo explora as bases teóricas necessárias para a análise comparativa entre Laravel Octane com Swoole e o Laravel convencional. Serão abordados tópicos essenciais, como a arquitetura web, o modelo de design Model-View-Controller (MVC), o paradigma de programação assíncrona e o conceito de corrotinas.

## 2.1 Arquitetura Web

O ambiente web, hoje, utiliza HTML e CSS para apresentar dados aos usuários e a interação é realizada via JavaScript. Essas tecnologias são chamadas de tecnologias “front-end” ou “client-side”. Por outro lado, as tecnologias “back-end” ou “do lado do servidor” referem-se a tecnologias de armazenamento e processamento de dados. A arquitetura de um ambiente web refere-se à estrutura e organização do sistema que suporta e gerencia a interação entre usuários e serviços por meio da internet.

Geralmente, essa arquitetura é projetada para garantir a eficiência, segurança, escalabilidade e manutenibilidade de aplicações web. Abaixo seguem alguns conceitos fundamentais relacionados à arquitetura web:

1. **Cliente-Servidor:**
    - Na arquitetura cliente-servidor, a aplicação é dividida em duas partes principais: o cliente, que é a interface do usuário, e o servidor, que processa as solicitações e fornece os recursos ou dados necessários.

- O cliente e o servidor interagem por meio de solicitações e respostas, geralmente utilizando protocolos de comunicação como HTTP.

2. **Modelo de Três Camadas:**
    - Este modelo divide uma aplicação em três camadas: apresentação, lógica de negócios e armazenamento de dados.
    - A camada de apresentação (front-end) lida com a interface do usuário.
    - A camada de lógica de negócios (back-end) gerencia as regras de negócios e a lógica de processamento.
    - A camada de armazenamento de dados (banco de dados) armazena e recupera informações.

3. **REST (Transferência de Estado Representacional):**
    - REST é um estilo arquitetural que utiliza a natureza sem estado do protocolo HTTP para criar serviços web simples e escaláveis.
    - Os recursos (como dados ou serviços) são identificados por URLs, e as operações (GET, POST, PUT, DELETE) são aplicadas a esses recursos.

4. **SOAP (Protocolo Simples de Acesso a Objetos):**
    - SOAP é um protocolo de comunicação utilizado em serviços web.
    - Ele define uma estrutura de mensagem XML para troca de informações entre sistemas distribuídos.

5. **Microsserviços:**
    - Na arquitetura de microsserviços, uma aplicação é dividida em serviços independentes e autônomos.
    - Cada microsserviço realiza uma função específica e se comunica com outros microsserviços por meio de APIs (Interfaces de Programação de Aplicações).

6. **Single Page Application (SPA):**
    - Uma SPA é uma aplicação web que interage com o usuário carregando dinamicamente as páginas, em vez de recarregar a página inteira.

- Isso proporciona uma experiência mais fluida, pois apenas partes específicas da página são atualizadas conforme necessário.

7. **Websockets:**
    - Websockets oferecem uma comunicação bidirecional em tempo real entre o cliente e o servidor.
    - Essa tecnologia é frequentemente utilizada em aplicativos que exigem atualizações instantâneas, como salas de bate-papo online ou aplicativos colaborativos.

A escolha da arquitetura web depende dos requisitos específicos do projeto, incluindo escalabilidade, desempenho, manutenibilidade e outros fatores. Cada modelo tem suas vantagens e desvantagens, e a seleção adequada depende do contexto e das necessidades do sistema em desenvolvimento.

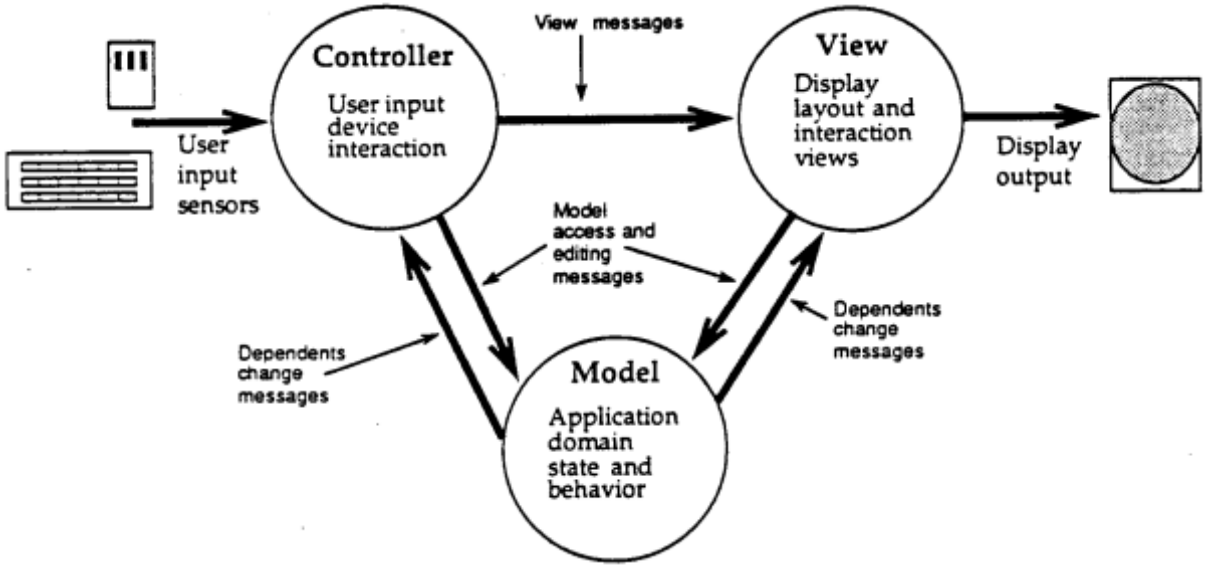
## 2.2 Arquitetura MVC

Segundo Pop e Altar (2014), o padrão de design Model-View-Controller (MVC) foi inicialmente proposto pela primeira vez por Trygve Reenskaug na década de 1970 no Xerox Parc. Segundo ele, "o propósito essencial do MVC é preencher a lacuna entre o modelo mental do usuário humano e o modelo digital existente no computador". Posteriormente, em 1988, o paradigma MVC foi detalhadamente descrito por Krasner e Pope em seu artigo "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80" (Um guia prático para o uso do paradigma de interface de usuário Model-View-Controller em Smalltalk-80".

Para Krasner e Pope (1988), o ciclo de interação padrão ocorre quando o usuário realiza uma ação de entrada. Em seguida, o Controller notifica o Model para que este execute as operações específicas referentes à ação realizada, o que potencialmente, pode alterar o seu estado. Além disso, o Model informa aos seus dependentes (View e Controller) que ocorreu uma mudança, indicando, a natureza dessa alteração. A View pode então consultar o Model sobre seu novo estado e atualizar sua exibição, se necessário. O Controller pode adaptar sua abordagem de

interação com base no novo estado do Model. Esse processo de envio de mensagens é ilustrado na Figura 1.

Figura 1 - Modelo de MVC detalhado por Krasner e Pope



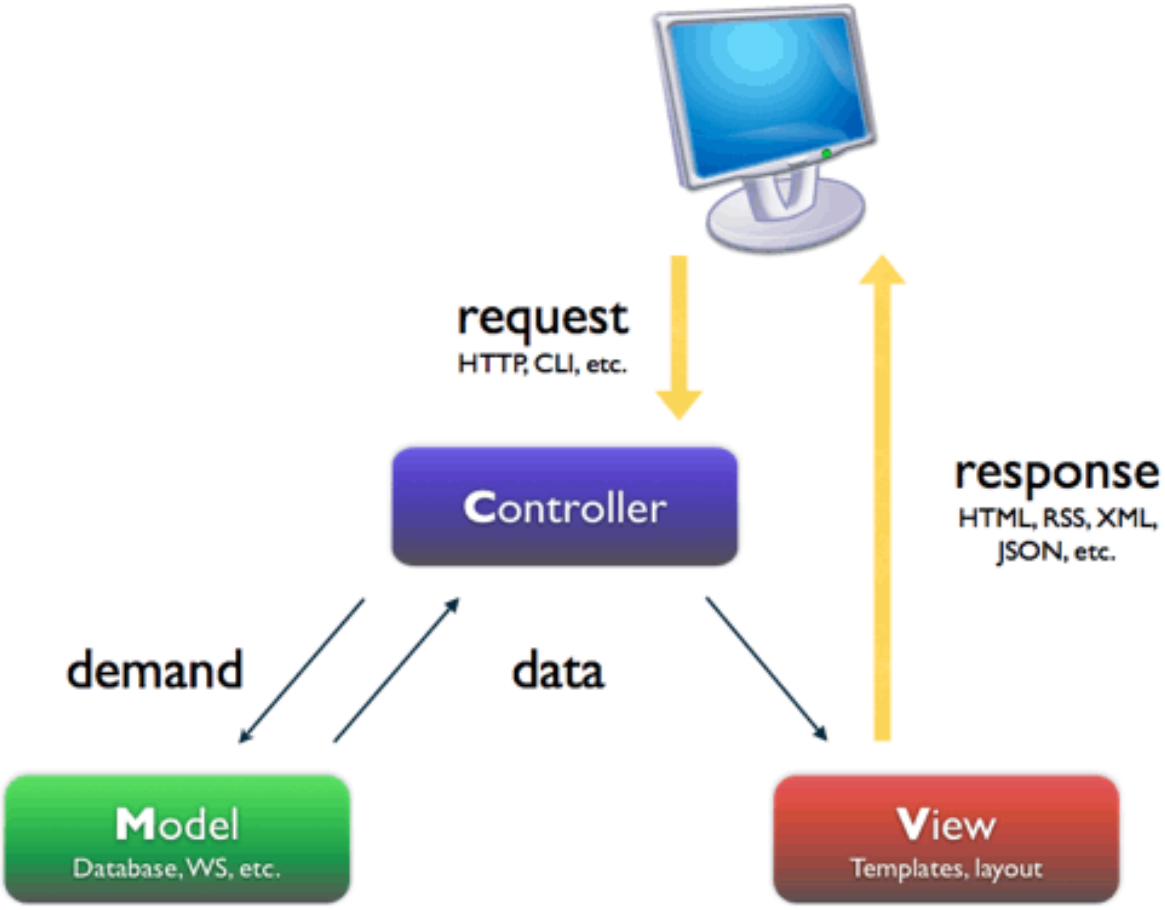

Fonte: Krasner e Pope (1988)

O padrão de design MVC é uma abordagem valiosa para arquitetar sistemas interativos, conforme destacado por Leff e Rayfield (2001). Ainda para Leff e Rayfield (2001), o MVC visa separar as interfaces do usuário dos dados subjacentes representados por essas interfaces. No MVC clássico, a View exibe informações ao usuário, enquanto o Controller processa as interações do usuário, formando a interface do aplicativo. O Model contém as informações representadas pela View e a lógica que altera essas informações em resposta às interações do usuário. A utilização do padrão MVC facilita o desenvolvimento e a manutenção do aplicativo, permitindo alterações significativas na aparência sem afetar estruturas de dados e lógica de negócios. Além disso, possibilita a manutenção de interfaces diversas e diferentes conjuntos de permissões de usuário.

Em contexto mais amplo, o termo "MVC" foi estendido para descrever a maneira como mudanças no Model de um aplicativo são impulsionadas por um Controller, não apenas para processar interações do usuário, mas também para alterar o estado geral do aplicativo em resposta aos eventos criados pela interação do usuário.

Para Pop e Altar (2014), a arquitetura Model-View-Controller (MVC) é um modelo que vai além da linguagem de programação PHP, abrangendo conceitos aplicáveis a diversas linguagens e ambientes de desenvolvimento. O padrão MVC divide as responsabilidades em três funções principais: desenvolvimento, design e integração, cada uma atendida por especialistas em suas áreas. Destaca-se que o MVC é uma escolha vantajosa para o desenvolvimento de aplicações web, pois combina diversas tecnologias geralmente distribuídas em camadas e proporciona interação conveniente para aplicações web entregues como uma série de solicitações e respostas HTTP.

Figura 2 - O padrão MVC




Fonte: https://stackoverflow.com/questions/5966905/what-is-the-right-mvc-diagram-for-a-web-application (2019)

### 2.2.1 Model

De acordo com Pop e Altar (2014), o Model é a parte do sistema responsável por gerenciar todas as tarefas relacionadas aos dados, como validação, estado e controle da sessão e estrutura referente à conexão com o banco de dados. Ele reduz significativamente a complexidade do código que o desenvolvedor precisa escrever. A camada do Model é encarregada da lógica de negócios de uma aplicação, encapsulando métodos para acessar dados (bancos de dados e outros arquivos) e disponibilizando uma biblioteca de classes reutilizável.

Geralmente, um Model é construído com a abstração de dados em mente, incluindo validação e autenticação. Além disso, o Model é composto por classes que definem o domínio de interesse, encapsulando dados armazenados em bancos de dados e código usado para manipular esses dados e aplicar regras de negócios (Pop e Altar, 2014).

### 2.2.2 View

Para Pop e Altar (2014), a camada View é responsável pelo gerenciamento da interface gráfica do usuário, abrangendo formulários, botões, elementos gráficos e todos os outros elementos HTML dentro da aplicação. Ao separar o design da aplicação da lógica, reduzimos significativamente o risco de erros ao realizar alterações na interface da aplicação, como mudanças de logotipo ou tabelas, feitas pelo designer.

Essa camada é comumente associada ao design web ou templates. Ela controla a maneira como os dados são exibidos e como o usuário interage com eles, fornecendo também meios para coletar dados dos usuários. As tecnologias principais usadas nas Views são HTML, CSS e JavaScript. Como regra geral, uma View não deve conter elementos que pertençam à lógica da aplicação. Isso significa que os blocos lógicos devem ser mantidos no mínimo, ou seja, evitar aplicar lógica de programação nas Views (Pop e Altar, 2014).

### 2.2.3 Controller

O Controller, segundo Pop e Altar (2014), é responsável pelo tratamento de eventos, os quais podem ser desencadeados tanto pela interação do usuário com a aplicação quanto por um processo do sistema. Um Controller aceita solicitações e prepara os dados para uma resposta, sendo também responsável por estabelecer o formato dessa resposta. Ele interage com o Model para recuperar os dados necessários e gera a View. Quando uma solicitação chega ao servidor, ela é encaminhada para um método em um Controller com base na URL.

O Controller vincula toda a lógica da aplicação e combina a exibição na View com a funcionalidade no Model. Ele é responsável pela recuperação de dados da View e por estabelecer o caminho de execução para a aplicação. O Controller acessará a funcionalidade do Model e interpretará os dados recebidos para que possam ser exibidos pela View. Além disso, é responsável pelo tratamento de erros. O Controller gerencia a relação entre uma View e um Model, respondendo a solicitações do usuário, interagindo com o Model e decidindo qual View deve ser gerada e exibida (Pop e Altar, 2014).

## 2.3 Programação Assíncrona

Conforme Ganty e Majumdar (2012), a programação assíncrona é uma abordagem comum na programação de sistemas, oferecendo uma maneira eficaz de lidar com interações concorrentes com o ambiente. Nesse estilo de programação, em vez de esperar que operações demoradas sejam concluídas, são realizadas chamadas não bloqueantes para a operação e tarefas de retorno serão executadas posteriormente quando a operação for finalizada.

Para Don, Tomas e Dmitry (2016), a programação é caracterizada por muitas reações simultaneamente pendentes a eventos internos ou externos. Essas reações podem ou não ser processadas em paralelo.

Programação assíncrona é um paradigma de programação que lida com a execução de tarefas de forma não sequencial, permitindo que outras operações

ocorram enquanto uma tarefa está sendo processada. Em vez de esperar até que uma operação seja concluída para iniciar a próxima, a programação assíncrona permite que o programa execute outras tarefas enquanto aguarda o término de uma operação demorada. As principais características da programação assíncrona incluem:

1. **Não Bloqueio:** Em operações assíncronas, o código não é bloqueado durante a espera por uma operação de entrada/saída (I/O) ou outras tarefas demoradas. Isso significa que o programa pode continuar executando outras operações enquanto aguarda a conclusão de uma operação assíncrona.

2. **Callbacks:** Uma técnica comum em programação assíncrona é o uso de callbacks. Em vez de aguardar a conclusão de uma operação, um callback (função) é fornecido para ser executado quando a operação estiver concluída.

3. **Promessas e Async/Await:** Além de callbacks, linguagens modernas frequentemente usam conceitos como promessas (promises) e palavras-chave async/await para facilitar a gestão de operações assíncronas. Isso torna o código mais legível e estruturado.

A programação assíncrona é comumente usada em situações onde a espera por operações de I/O, como leitura de arquivos, chamadas de rede, ou interações com bancos de dados, pode levar algum tempo. Isso permite que o programa continue respondendo a eventos ou execute outras tarefas enquanto aguarda respostas assíncronas.

Linguagens como JavaScript, Python, e outras oferecem suporte robusto para programação assíncrona. Essa abordagem é especialmente valiosa em ambientes onde a eficiência e a responsividade são críticas.

## 2.4 Corrotinas

O conceito de coroutines (corrotinas) foi introduzido no início da década de 1960 como um meio de estruturar um compilador, afirmam Anton e Thiemann (2010), e constitui uma das propostas mais antigas de uma abstração geral de controle,

conforme Moura e Ierusalimschy (2004). Para Anton e Thiemann (2010), as corrotinas receberam muita atenção na comunidade de programação e foram integradas a várias linguagens de programação, como Simula 67, BETA, CLU, Modula-2, Python e Lua. A adequação das corrotinas para expressar vários comportamentos de controle úteis foi amplamente explorada em contextos diversos, incluindo simulação, inteligência artificial, programação concorrente, processamento de texto e manipulação de estruturas de dados (Moura e Ierusalimschy, 2004).

Já para Elizarov et al (2021), apesar das corrotinas serem utilizadas de uma forma ou de outra por mais de 50 anos, ainda não existe uma definição universal do que é uma corrotina. Quando se fala em corrotinas, geralmente se refere a algo semelhante a uma função suspensível: uma função que pode ser suspensa e retomada, preservando o estado entre as suspensões. Essas propriedades foram destacadas como características das corrotinas já em 1980.

Segundo Anton e Thiemann (2010), uma corrotina é uma construção de programação situada entre uma função e uma thread. Pode ser invocada como uma função, mas antes de retornar um valor (se retornar), pode se suspender arbitrariamente várias vezes para retornar resultados intermediários e, em seguida, ser retomada com novas entradas. Uma corrotina não é executada simultaneamente com o restante do programa, mas assume o controle até decidir suspender sua execução voluntariamente para retornar o controle para quem a chamou ou passar o controle para outra corrotina. As corrotinas estão intimamente relacionadas à programação cooperativa, mas acrescentam valor porque são capazes de passar valores para dentro e para fora da corrotina e permitem a troca explícita de controle.

A programação concorrente é essencial para lidar com atividades independentes, tradicionalmente implementada por meio de threads com memória compartilhada. No entanto, essa abordagem enfrenta desafios como troca de mensagens e sincronização, resultando em um alto consumo de recursos. As corrotinas surgem como uma solução mais simples e leve para a concorrência.

No contexto de aplicativos que atendem a dezenas ou centenas de milhares de clientes simultaneamente, a concorrência é crucial para mitigar a latência das

operações I/O. As corrotinas proporcionam uma forma eficaz de aplicar concorrência, permitindo a passagem de valores entre atividades independentes. Os canais são utilizados como mecanismo para a comunicação entre corrotinas, possibilitando a transferência de valores.

Embora algumas vezes chamadas de "threads leves" ou "green threads", é importante destacar que as corrotinas não são threads no sentido tradicional. Todas as operações ocorrem no modo do usuário, sem a necessidade de envolvimento direto do kernel, resultando em custos de criação e consumo de recursos significativamente reduzidos. Comparar corrotinas com threads é útil para compreender as situações ideais para sua utilização. O Swoole, por exemplo, cria uma corrotina para cada requisição recebida, seguindo uma abordagem também adotada pelo Go. Um artigo futuro será dedicado exclusivamente às corrotinas (Tedesco, 2019).

# 3 TRABALHOS RELACIONADOS

Neste capítulo, exploramos quatro trabalhos relacionados ao tema da pesquisa, com foco na otimização de aplicações web. A seleção desses trabalhos foi conduzida manualmente, envolvendo a análise dos temas para identificar aqueles mais alinhados com os objetivos desta pesquisa. Os trabalhos mencionados foram cuidadosamente escolhidos a partir de pesquisas realizadas no Google Acadêmico e no Scopus, utilizando como termos principais de busca 'Otimização' e 'Aplicações web'. Adicionalmente, para uma comparação mais detalhada com o escopo da pesquisa, foram também utilizados termos específicos como 'Laravel', 'PHP', 'Swoole' e 'Laravel Octane'. Embora não tenham sido encontrados muitos trabalhos diretamente relacionados ao assunto, os selecionados demonstraram ser de grande valia para o desenvolvimento e entendimento do tema abordado nesta pesquisa.

## 3.1 Research and Practice of Swoole Asynchronous Multithreading Design Method

Yang, Shan e Chen (2018) apresentam a implementação do Swoole, testando o desempenho do servidor baseado nele e destaca a capacidade de processamento concorrente. O artigo explora a aplicação prática do Swoole em um caso de uso de sala de bate-papo em tempo real, demonstrando sua eficácia na resolução de problemas de concorrência.

A parte prática do estudo envolve a construção de um servidor local, implementação de envio assíncrono de SMS, criação de eventos ao vivo e cenas de chat, e teste de concorrência do servidor Swoole. Os resultados dos testes indicam

que o PHP+Swoole pode substituir linguagens de programação complexas, como C++ e Java, em cenários de alta concorrência, superando as limitações tradicionais do PHP em programação de rede.

Os testes de estresse mostram que o servidor Swoole pode suportar uma considerável carga concorrente, embora o aumento no número de usuários e concorrência resulte em um aumento significativo no tempo de resposta médio. A implementação do Swoole redefiniu a capacidade do PHP, permitindo o desenvolvimento de funções de comunicação de rede que anteriormente não eram viáveis. O estudo conclui que o Swoole tem o potencial de melhorar a eficiência de desenvolvimento, expandindo o escopo de aplicação do PHP em ambientes de alta concorrência.

### 3.2 Analysis and Research on the Performance Optimization of Web Application System in High Concurrency Environment

Yao e Xia (2016) apresentam estratégias e esquemas de otimização de desempenho em sistemas de aplicativos web de alta concorrência. O estudo introduz indicadores de teste de desempenho, métodos de teste de desempenho e estratégias de otimização de desempenho. Com base nisso, o artigo explica o esquema de otimização a partir de três aspectos: acesso do navegador, servidor e armazenamento de dados, com foco na estrutura geral do sistema de aplicativo web de alta concorrência.

O trabalho realiza uma validação dos resultados, demonstrando que o tempo de resposta do sistema é reduzido em cerca de 30%, o número de concorrência do sistema é melhorado em 4 vezes, a taxa de utilização de recursos do servidor é melhorada em 60%, e a otimização de desempenho do sistema alcança bons resultados.

Um exemplo prático utilizado no estudo é um serviço de rede de informações para detecção e supervisão do consumo de combustível de veículos de transporte rodoviário. O sistema lida com um grande número de usuários (9105727), com picos de acesso concorrente chegando a 10000 pessoas. O artigo descreve as otimizações

realizadas em diferentes partes do sistema para lidar com o grande volume de dados e alto acesso concorrente, incluindo a utilização de páginas estáticas, junção de recursos CSS e imagens, caching e compressão de classes estáticas, uso de servidores proxy reversos, adição de servidores de cache Memcached, separação de leitura e gravação no banco de dados, substituição de discos por unidades de estado sólido e a implementação de um array de discos RAID5.

Os resultados dos testes após a otimização mostram que o tempo de resposta médio do sistema foi reduzido em cerca de 30%, a capacidade de carga foi melhorada 4 vezes, a taxa de utilização de recursos do servidor foi melhorada em 60%, evidenciando um impacto positivo na otimização de desempenho em ambientes de alta concorrência. O estudo conclui que a otimização de sistemas de aplicativos web de alta concorrência é significante.

### 3.3 The Solution of Web Font-end Performance Optimization

Cao, Shi e Li (2017) exploram a otimização de desempenho de sites, focando especialmente na otimização do front-end. O trabalho destaca que a otimização de sites é dividida em duas abordagens: otimização de front-end e otimização de back-end. A pesquisa concentra-se na otimização de front-end, argumentando que apenas 10% a 20% do tempo é gasto na obtenção de documentos HTML do servidor da Web para os navegadores front-end, enquanto 80% a 90% do tempo é usado na resposta ao front-end.

O artigo propõe métodos de otimização em cinco aspectos: redução de solicitações HTTP, otimização de elementos de página, otimização de cache do cliente, otimização de DNS e tecnologia de compressão. Algumas estratégias específicas incluem a tecnologia CSS Sprites para reduzir solicitações HTTP, a junção dos códigos JavaScripts e CSS, centrada em apenas um arquivo, para reduzir o número de solicitações, e a otimização de elementos HTML, CSS e JavaScript.

Além disso, o trabalho aborda a otimização do cache do cliente, sugerindo o uso de Content Delivery Network (CDN), a adição de cabeçalhos Expires e o uso de Etags para confirmar a validade do componente armazenado em cache. O texto

também discute a otimização de cookies, recomendando a remoção de cookies desnecessários, minimização de seu tamanho e configuração de tempos de expiração apropriados. A otimização do DNS é explorada para reduzir o tempo de pesquisa DNS, destacando a importância de minimizar o número de pesquisas DNS e sugerindo a colocação de todos os recursos de conteúdo sob o mesmo domínio.

E por fim, a tecnologia de compressão é apresentada, com ênfase especial na compressão Gzip para reduzir o tamanho da página e melhorar o desempenho do aplicativo da web.

Em resumo, o trabalho fornece uma visão abrangente das estratégias de otimização de desempenho para sites, com foco específico no front-end, e oferece uma análise detalhada dos métodos propostos.

### 3.4 Introdução ao Swoole, framework PHP assíncrono baseado em corrotinas

Tedesco (2019), faz uma abordagem sobre o Swoole, destacando sua relevância e eficácia no desenvolvimento concorrente com alto desempenho e uso eficiente de recursos. Explora também a arquitetura tradicional com PHP-FPM e os desafios enfrentados em cenários de alta concorrência, ressaltando a solução proporcionada pelo modelo de I/O não bloqueante do Swoole. O papel central das corrotinas no framework é destacado, apresentando-as como uma abordagem leve e eficiente para lidar com a concorrência. A implementação do Swoole é comparada com outras tecnologias, evidenciando sua capacidade de lidar com milhares de conexões simultâneas.

O trabalho também aborda a comparação entre execução síncrona (tradicional) e assíncrona usando Swoole em PHP. O principal foco está na análise do desempenho e eficiência das abordagens, evidenciando as diferenças em termos de tempo de execução e retorno de resultados. Ele destaca a otimização do tempo de execução por meio do uso de corrotinas e operações não bloqueantes com Swoole. A comparação enfatiza como a abordagem assíncrona pode ser mais eficiente em determinados cenários, evitando bloqueios de I/O e melhorando a escalabilidade.

A pesquisa apresenta duas implementações: uma síncrona, utilizando a função sleep para simular bloqueio de I/O, e outra assíncrona, empregando corrotinas do Swoole. A estratégia assíncrona demonstra uma execução mais eficiente em termos de tempo, evitando a espera bloqueante e, consequentemente, melhorando o desempenho. Além disso, o trabalho realiza um benchmarking comparativo entre servidores com retorno "hello world" em diferentes plataformas (PHP embutido, NodeJS, Go e Swoole), destacando o desempenho impressionante do Swoole, especialmente em cenários de alto volume de requisições.

O artigo conclui que o Swoole, além de proporcionar uma abordagem assíncrona eficiente, suporta aplicações multithreading. A contextualização do benchmarking em exemplos simples é reconhecida, com a ressalva de que o impacto pode variar em aplicações mais complexas com uso intensivo de I/O.

### 3.5 Comparativo entre os trabalhos relacionados

Com base nas análises detalhadas dos trabalhos relacionados apresentados nas seções anteriores, é possível realizar um comparativo abrangente entre essas propostas e a pesquisa em questão. O Quadro 1 destaca as principais características e abordagens adotadas em cada trabalho, permitindo uma visão comparativa entre eles.

Quadro 1 - Trabalhos relacionados

| **Aspecto** | **Trabalho 1** | **Trabalho 2** | **Trabalho 3** | **Trabalho 4** |
|---|---|---|---|---|
| **Autores (Ano)** | Yang, Shan e Chen (2018) | Yao e Xia (2016) | Cao, Shi e Li (2017) | Tedesco (2019) |
| **Foco Principal** | Otimização no tempo de resposta em um ambiente de alta concorrência. | Otimização do desempenho em ambientes de alta concorrência em | Otimização do front-end, com ênfase em redução de | Análise de desempenho e eficiência em relação a execuções |

| | | | | |
|---|---|---|---|---|
| | | sistemas de aplicação web. | solicitações HTTP. | síncronas e assíncronas |
| **Áreas de Otimização** | Back-end aplicando o Swoole para otimização em tempo real. | Acesso do navegador, servidor e armazenamento de dados. | Front-end, cache do cliente e DNS. | Tempo de execução e de resposta |
| **Principais Estratégias** | Implementação do Swoole em um cenário prático de sala de bate-papo. | Uso de páginas estáticas, junção de recursos CSS e imagens, caching e técnicas como Memcached e RAID5. | Uso de CSS Sprites, otimização de JS e CSS, uso de CDN, Expires, Etag e compressão Gzip. | Uso de corrotinas do Swoole |

Fonte: Do autor (2023)

Analisando os trabalhos relacionados em conjunto, observa-se uma semelhança em relação a otimização de desempenho em ambientes web de alta concorrência. Os estudos destacam a importância de estratégias que visam melhorar a eficiência em diferentes camadas da arquitetura de sistemas, incluindo o acesso do navegador, o servidor e o armazenamento de dados. No geral, há uma ênfase compartilhada na redução do tempo de resposta do sistema, aumento da capacidade de carga e utilização mais eficiente dos recursos do servidor.

O primeiro trabalho apresenta uma aplicação prática do Swoole em um cenário de sala de bate-papo em tempo real, demonstrando a eficácia dessa tecnologia em situações dinâmicas e concorrentes. O segundo trabalho foca em otimizações globais e estratégias gerais. Já o terceiro trabalho, se concentra nas otimizações do front-end, destacando a relevância de melhorias na resposta ao cliente, como a redução de solicitações HTTP, implementação de cache no cliente, otimização DNS e uso de compressão Gzip. E por fim, o quarto trabalho demonstra uma análise comparativa entre execuções síncronas e assíncronas, utilizando o Swoole para otimizar o tempo de execução e de resposta.

Os estudos fornecem um panorama abrangente de estratégias e tecnologias utilizadas para otimizar o desempenho de sistemas web e contribuem para a compreensão de práticas eficazes na melhoria do desempenho de aplicações web em ambientes PHP. Os estudos que empregam o Swoole para otimizar aplicações web PHP são especialmente relevantes para a proposta de pesquisa, que compara o Laravel Octane com Swoole e o Laravel convencional.

Neste estudo, a análise comparativa entre Laravel Octane com Swoole e Laravel convencional oferece um novo panorama sobre otimização de aplicações web PHP. Enquanto os trabalhos de Yang, Shan e Chen (2018) e Tedesco (2019) também focam no Swoole, evidenciando sua eficácia em ambientes de alta concorrência, a pesquisa atual se destaca ao comparar diretamente duas configurações distintas de Laravel, proporcionando insights específicos sobre desempenho, escalabilidade e consumo de recursos. Diferentemente do estudo de Yao e Xia (2016), que aborda a otimização de desempenho em um contexto mais amplo, e do trabalho de Cao, Shi e Li (2017), centrado na otimização do front-end, esta pesquisa fornece uma compreensão mais aprofundada das particularidades do PHP, destacando aspectos específicos importantes na escolha de tecnologias para aplicações web. Assim, ao integrar e expandir os conhecimentos destes estudos, a pesquisa atual sublinha a complexidade e a relevância da escolha tecnológica adequada para otimização em ambientes PHP específicos.

# 4 MATERIAIS E MÉTODOS

Neste capítulo, é abordado a classificação da pesquisa, detalhando sua natureza, abordagem e objetivos, estabelecendo uma sólida base para o estudo. Em seguida, serão apresentadas as tecnologias escolhidas, justificando cada seleção. A seção de desenvolvimento delineia a arquitetura, requisitos e casos de uso, proporcionando uma visão clara do que será desenvolvido. Este capítulo visa fornecer uma compreensão abrangente dos materiais, métodos e diretrizes que orientarão a execução deste trabalho.

## 4.1 Pesquisa enquanto aos Métodos Científicos

O método empregado nesta pesquisa é o dedutivo, seguindo a abordagem proposta por Marconi e Lakatos (2003). O método dedutivo busca definir o conteúdo dos fatos por meio da validação das premissas relacionadas a uma determinada questão. Em conformidade com essa abordagem, se as premissas forem consideradas verdadeiras, a conclusão final também é considerada verdadeira.

Com isso, esta pesquisa utiliza o método dedutivo para realizar uma investigação estruturada e lógica, contribuindo para a obtenção de conclusões sólidas e coerentes no contexto da presente pesquisa sobre otimização de aplicações web.

Conforme Filho e Filho (2015), a pesquisa qualitativa parte de um pressuposto em que há uma relação dinâmica entre o mundo real e o autor, pesquisador, entre o mundo objetivo e a subjetividade de quem observa, impossibilitando a tradução em números. O conhecimento dos aspectos e informações relacionadas ao tema, são

essenciais nos processos desta pesquisa. Já a pesquisa quantitativa, ela se inicia a partir da coleta de informações que possibilitam a tradução em números para uma abordagem que permite a análise e classificação destes dados. São utilizados recursos e técnicas estatísticas para a sua execução.

Desse modo, são utilizados estes dois modos de abordagem no presente estudo. A pesquisa qualitativa contribuirá para uma compreensão mais contextualizada, enquanto a pesquisa quantitativa proporcionará dados numéricos que podem ser analisados de maneira estatística, resultando em uma compreensão mais completa e robusta do fenômeno em questão.

Esta pesquisa se baseia na metodologia de pesquisa exploratória. A pesquisa exploratória possui o objetivo de propiciar uma maior adaptação ao problema, tornando-o mais claro. Acaba sendo bastante flexível, pois é desejado a consideração de diversos aspectos relacionados ao assunto. O levantamento dos elementos pode ser feito a partir do levantamento bibliográfico, de entrevistas com pessoas que tiveram experiência prática com o assunto ou da análise de exemplos que estimulem a compreensão (Gil, 2022).

Com isso, a pesquisa se baseia em diversos aspectos e fatores relacionados ao tema. Qualquer questão será levada em conta para ter uma melhor análise e compreensão do assunto.

Em relação aos procedimentos técnicos ou meio de investigação, este estudo segue os conceitos da pesquisa bibliográfica e levantamento. Segundo Ramos (2009), a pesquisa bibliográfica pode ser definida como o estudo de obras publicadas relacionadas ao assunto que está sendo estudado. Ou seja, um conjunto de conhecimentos fundamentais para o andamento da pesquisa. A pesquisa bibliográfica é essencial para qualquer pesquisa científica. O método de pesquisa de levantamento pode ser considerado como a análise de uma amostragem, ou seja, análise a partir da coleta de dados relacionados ao estudo pesquisado (Filho e Filho, 2015).

Para este estudo, além da realização da coleta de dados a partir de testes realizados obtendo assim, informações sobre desempenho, uso de recursos e tempo

de resposta, foram realizados estudos e análises em outras bibliografias, como por exemplo artigos, que estão relacionados com o tema estudado.

## 4.2 Tecnologias

Nesta seção são apresentadas e justificadas as tecnologias selecionadas para o desenvolvimento da pesquisa, visando estabelecer uma base sólida para os experimentos subsequentes, proporcionando uma compreensão clara das ferramentas e frameworks utilizados.

### 4.2.1 PHP: Hypertext Preprocessor

Segundo Laaziri et al (2019), a linguagem de programação PHP é considerada uma das linguagens mais amplamente utilizadas no desenvolvimento de aplicações web, pois oferece grande flexibilidade, é fácil de usar e de aprender. Possui recursos intuitivos, execução escalável e eficiente, código aberto, compatibilidade entre plataformas e suporte para SQL.

### 4.2.2 Frameworks PHP

Laarizi et al (2019) afirmam que, os frameworks PHP desempenham um papel crucial no desenvolvimento web, oferecendo eficiência, confiabilidade e escalabilidade. Essas ferramentas agilizam o processo de desenvolvimento, reduzem o esforço de codificação repetitiva e auxiliam na conformidade com padrões de codificação, resultando em código mais robusto e facilmente mantido. Os frameworks PHP são baseados no modelo de design Model, View, Controller (MVC), que separa os componentes de uma aplicação para melhor modularização e organização. Isso simplifica o design arquitetônico, aumentando a flexibilidade e a reutilização do código. Embora existam muitas opções, como Symfony, CodeIgniter, Laravel, Cake PHP e Yii, a escolha do framework ideal permanece um desafio. Os desenvolvedores devem considerar cuidadosamente as características de suporte e as necessidades de seus projetos para tomar a decisão mais informada.

### 4.2.3 Laravel

O framework Laravel, conforme Olanrewaju, Islam e Ali (2015), é um framework de aplicação web PHP de código aberto e gratuito, com foco em desenvolvimento MVC (Model-View-Controller). Ele oferece uma estrutura com uma variedade de recursos e facilidades para desenvolvedores de aplicações web em PHP, facilitando o processo de desenvolvimento e aumentando a eficiência.

### 4.2.4 Laravel Octane

Laravel Octane é uma extensão do framework Laravel projetada para otimizar o desempenho e a escalabilidade de aplicativos web PHP. Ele permite a execução em paralelo de tarefas, integrando-se com a extensão Swoole, um servidor assíncrono. Ideal para aplicativos com alta demanda, o Octane oferece respostas mais rápidas e maior eficiência, tornando-o uma escolha valiosa para lidar com tráfego intenso e melhorar a experiência do usuário em aplicativos web.

Conforme RunCloud Team (2022), O Laravel Octane é um pacote open-source que melhora o desempenho de sua aplicação Laravel por meio de programação PHP stateful. O pacote inclui dependências, bibliotecas e compatibilidade com servidores de alto desempenho, como RoadRunner e Swoole. O Laravel Octane roda sobre o Laravel e PHP na versão 8.0 ou superior. Ele é uma adição ao Laravel padrão e não representa uma linguagem separada. Em termos de desempenho, enquanto aplicativos Laravel padrão podem processar até 500 solicitações por segundo, o Laravel Octane facilmente ultrapassa 2.000 solicitações. O Laravel Octane opera de maneira diferente em relação aos servidores, tendo seu próprio servidor através do Swoole e RoadRunner, ao qual o Nginx e o Apache redirecionam o tráfego recebido.

No entanto, uma diferença crucial entre o Laravel padrão e o Laravel Octane está no uso do PHP. O Laravel Octane transforma seu stack PHP em um framework PHP semi-stateful, ou seja, ele atua em conjunto com o usuário ou cliente, lembrando ativamente os dados durante toda a sessão, ao contrário do PHP stateless, que possui servidores independentes das solicitações do usuário ou cliente. Para o Laravel Octane ser mais rápido, adotou-se um framework stateful, visto que,

tradicionalmente, cada solicitação cria um novo processo PHP, resultando em um alto tempo de inicialização. O Laravel Octane compartilha o mesmo framework iniciado entre as solicitações, evitando a necessidade de inicializar um novo framework para cada solicitação, o que contribui significativamente para seu desempenho superior (RunCloud Team, 2022).

RunCloud Team (2022) ainda afirmam que, além dos benefícios de desempenho, o Laravel Octane oferece recursos como Octane Workers, que cria vários workers desde o início para lidar com solicitações imediatamente. Também permite tarefas concorrentes através de task-workers, melhorando ainda mais o desempenho. Esses recursos são específicos para o Swoole e são detalhadamente explorados nos itens seguintes, especificamente nos itens 1 e 2.

1. **Octane Workers (workers)**

Segundo RunCloud Team (2022), vários workers são criados desde o início da execução, tornando mais rápido o atendimento imediato de solicitações pelo servidor à medida que são recebidas.

É possível especificar a quantidade inicial de workers. Ter vários workers é benéfico, pois se um worker estiver ocupado com uma tarefa, o outro pode começar a atender outras solicitações do usuário sem atrasos. É assim que Swoole aumenta significativamente o desempenho (RunCloud Team, 2022). A Figura 3 retrata um exemplo de inicialização do Laravel Octane com 4 workers iniciais.

Figura 3 – Código para iniciar o Octane com o parâmetro workers

```
php artisan octane:start --workers=4
```

Fonte: Do Autor, adaptado de RunCloud Team (2022)

2. **Task-workers**

O Laravel Octane também permite a execução de tarefas concorrentes, o que significa que essas tarefas serão realizadas simultaneamente, não uma após a outra. Isso pode ser feito por meio de task-workers, conforme exemplo na Figura 3.

Figura 4 – Código para iniciar o Octane com os parâmetros workers e task-workers

```
php artisan octane:start --workers=4 --task-workers=6
```

Fonte: Do Autor, adaptado de RunCloud Team (2022)

Por ser uma extensão do framework Laravel e possuir Swoole integrado, essa tecnologia foi escolhida para realizar execuções simultâneas, além de possuir uma documentação robusta e ser própria do Laravel, com o objetivo de otimizar aplicações web que utilizam o Laravel como framework.

### 4.2.5 Swoole

Para Yang, Shan e Chen (2018), o Swoole é uma extensão do PHP implementado em C, que funciona como um framework subjacente baseado em comunicação via sockets. Ele possibilita uma abordagem assíncrona, onde várias operações podem ser executadas simultaneamente, mesmo em um único processador, o que o torna uma ferramenta valiosa para otimizar o desempenho e a eficiência em aplicações web baseadas em PHP.

Conforme RunCloud Team (2022), o Swoole possui limitações com ferramentas de monitoramento como Xdebug e pode ter problemas com outras ferramentas de monitoramento.

Para Miladev (2023), o Swoole é uma extensão PHP que oferece um conjunto robusto de funcionalidades para o desenvolvimento de aplicações de rede de alto desempenho. Construída sobre um modelo de I/O de rede assíncrono e orientado a eventos, permite lidar com muitas conexões de rede com uso baixo de recursos. É

ideal para aplicações concorrentes e em tempo real, como servidores de chat e jogos, oferecendo alta taxa de transferência e baixa latência. Suas características incluem os seguintes itens:

1. **I/O de Rede Assíncrona**: Permite o manuseio simultâneo de múltiplas conexões de rede sem bloquear a execução principal, favorecendo alta concorrência e reduzindo sobrecarga.

2. **Corrotinas**: Suporta concorrência baseada em corrotinas, mais leve e eficiente que threads tradicionais, facilitando a escrita e manutenção de código PHP concorrente.

3. **Suporte a Servidores WebSocket e HTTP**: Facilita o desenvolvimento de aplicações web em tempo real que podem gerenciar muitas conexões.

4. **Suporte a Timer e Loop de Eventos**: Inclui um sistema de timer e loop de eventos para agendar tarefas e eventos para momentos específicos ou em resposta a gatilhos específicos.

A escolha do Swoole como extensão foi definida por conta da possibilidade de realizar execuções simultâneas, além de estar integrado com o Laravel Octane, o que facilitou o uso da ferramenta. E outra justificativa é pelo fato de ser uma extensão do PHP.

### 4.2.6 HTML e CSS

Conforme MDN Web Docs (2023), HTML (Hypertext Markup Language - Linguagem de Marcação de HiperTexto) representa a base fundamental da web, delineando tanto o significado quanto a estrutura do conteúdo da web. Já o CSS (Cascading Style Sheets - Folhas de Estilo em Cascata), para MDN Web Docs (2022), é uma linguagem de formatação utilizada para definir como um documento, seja em HTML ou XML, é apresentado. Ele determina a aparência dos elementos na tela, em impressões, em leituras em voz alta ou em outras formatos de mídia.

### 4.2.7 Javascript e jQuery

MDN Web Docs (2022) afirma que o JavaScript é uma linguagem de programação leve e orientada a objetos, amplamente conhecida como linguagem de script para páginas web, mas também utilizada em diversos outros ambientes fora do navegador, como Apache CouchDB e Adobe Acrobat. É uma linguagem dinâmica, baseada em protótipos e suporta vários estilos de programação, incluindo orientação a objetos, imperativa e declarativa.

Flanagan (2011) destaca o jQuery como uma das bibliotecas Javascript mais populares e amplamente utilizadas, conhecida por sua estabilidade. Além disso, o jQuery apresenta utilitários Ajax que facilitam a realização de solicitações HTTP dinâmicas, bem como funções utilitárias de propósito geral para trabalhar com objetos e arrays.

O uso do Javascript e da biblioteca jQuery nesta pesquisa visa otimizar o desenvolvimento, aproveitando as vantagens oferecidas pela praticidade e versatilidade dessas tecnologias. A escolha do jQuery também se deve à facilidade de seleção e manipulação de elementos em documentos HTML.

### 4.2.8 Ajax

De acordo Flanagan (2011), Ajax é um termo que descreve uma arquitetura para aplicações web que destaca o uso de HTTP com scripts. O principal recurso de uma aplicação Ajax é sua capacidade de utilizar HTTP com scripts para iniciar a troca de dados com um servidor web sem recarregar as páginas. Isso resulta em aplicações web responsivas que se assemelham mais a aplicativos de desktop tradicionais, pois evitam os recarregamentos de página comuns nos estágios iniciais da web.

A escolha de incorporar a tecnologia Ajax neste projeto se justifica pela sua capacidade de aprimorar a experiência do usuário, além da facilidade do uso na implementação, possibilitando a atualização dinâmica de conteúdo sem a necessidade de recarregar a página. Essa abordagem visa proporcionar uma

navegação mais fluida e interativa, contribuindo para a eficiência e agilidade na interação com a aplicação web desenvolvida em PHP.

### 4.2.9 Ferramentas para análise de testes de carga

Nesta seção, são exploradas duas ferramentas fundamentais para testes de carga em aplicações web: 'Wrk: Benchmarking Tool' e 'Htop: Comando Linux'. Estas ferramentas são detalhadas nos itens 1 e 2.

1. **Wrk: Benchmarking Tool**

De acordo com Silva (2018), wrk é uma ferramenta moderna de benchmarking HTTP capaz de gerar uma carga significativa quando executada em uma única CPU com diversos núcleos.

Figura 5 – Exemplo de benchmark de um ponto de extremidade HTTP

```
wrk -t12 -c400 -d30s --latency http://127.0.0.1:8080/index.html
```

Fonte: Do Autor, adaptado de Silva (2018)

Conforme exemplo na Figura 5, o comando executa um benchmark por 30 segundos, usando 12 threads e mantendo abertas 400 conexões HTTP. O retorno do comando pode ser observado na Figura 6:

Figura 6 – Retorno de teste de benchmark via wrk

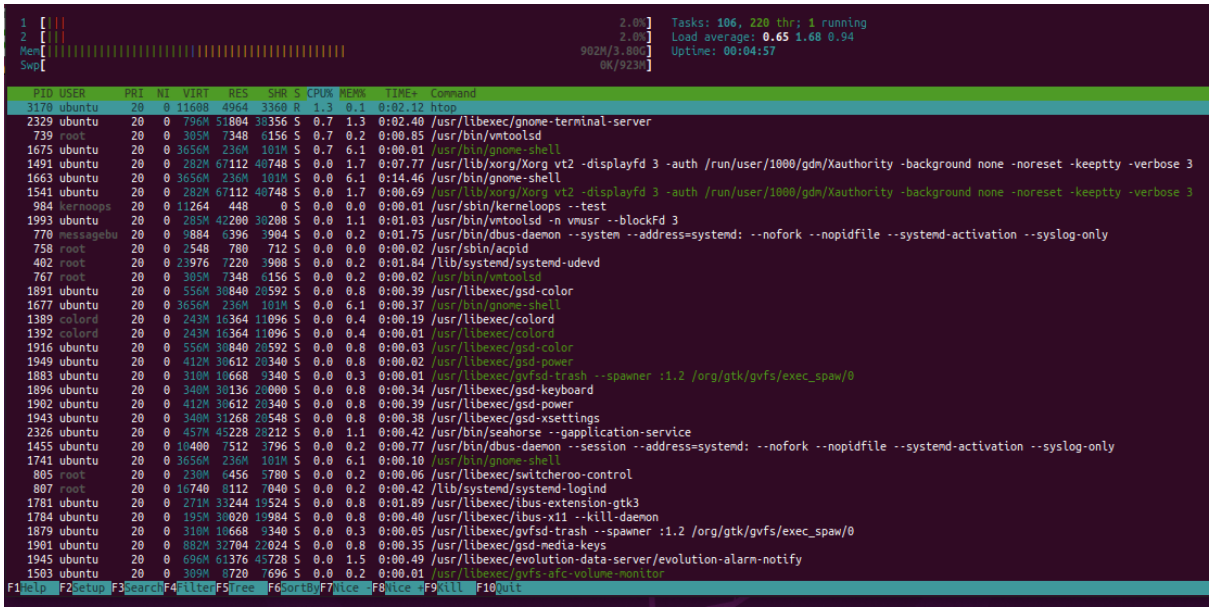
```
Running 30s test @ http://localhost:8080/index.html
  12 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   635.91us    0.89ms  12.92ms   93.69%
    Req/Sec    56.20k     8.07k   62.00k    86.54%
Latency Distribution
  50% 250.00us
  75% 491.00us
  90% 700.00us
  99% 5.80ms
22464657 requests in 30.00s, 17.76GB read
Requests/sec: 748868.53
Transfer/sec:    606.33MB
```

Fonte: Do Autor, adaptado de Silva (2018)

## 2. Htop: Comando Linux

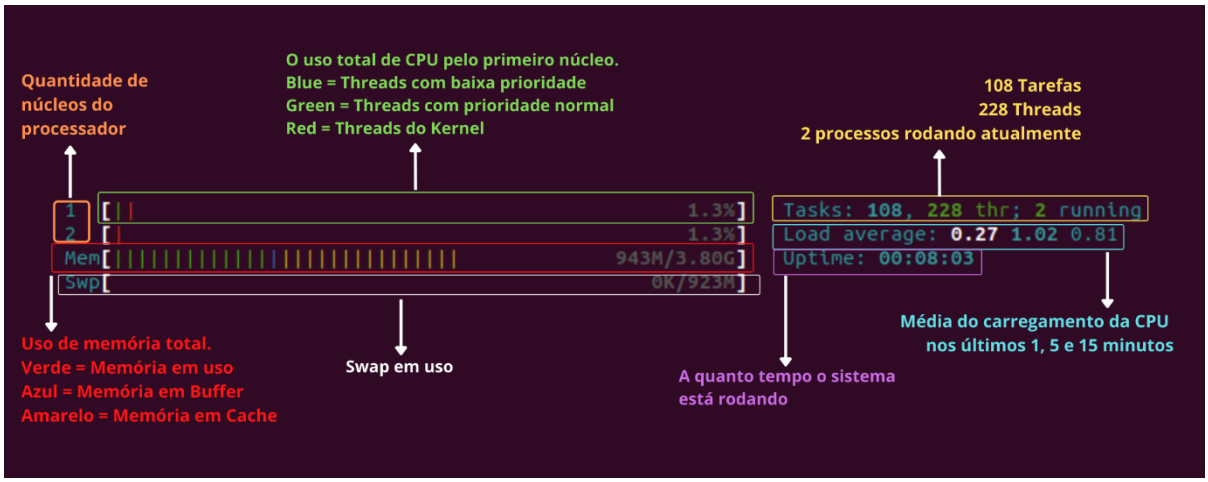
Conforme Souza (2022), o comando htop é uma ferramenta de linha de comando desenvolvida para proporcionar ao usuário uma monitoração interativa e em tempo real dos recursos do seu sistema operacional Linux. Para Kumar (2022), permite rolar vertical e horizontalmente para visualizar todos os processos em execução no sistema, exibir a árvore de processos, selecionar e agir em vários processos ao mesmo tempo. Ele ajuda a identificar qual processo está usando mais CPU, quanto de memória cada processo está consumindo, entre outras informações, conforme ilustrado nas figuras Figura 7 e Figura 8. Também é possível usar o comando htop para alterar as prioridades da CPU e melhorar o desempenho do computador.

Figura 7 – Interface exibida ao executar htop




Fonte: Souza (2022)

Figura 8 – Parte superior detalhada na interface do htop




Fonte: Souza (2022)

### 4.3 Desenvolvimento

Nesta seção, são descritos os detalhes de cada etapa do desenvolvimento da pesquisa.

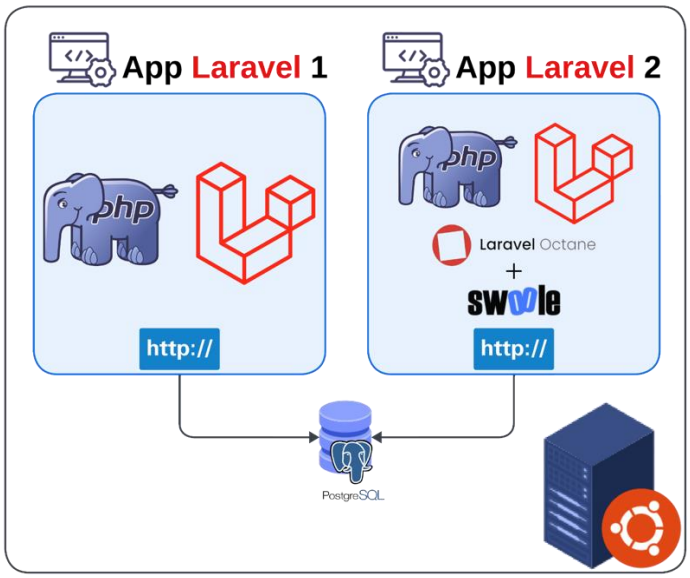
#### 4.3.1 Descrição do projeto

Para a condução dos testes comparativos, foram estabelecidas duas aplicações web PHP com o framework Laravel distintas em um único servidor. A escolha de um servidor único foi estratégica, visando garantir configurações idênticas para ambas as aplicações, proporcionando uma avaliação precisa e equitativa dos dados comparativos. Em uma das aplicações, foram integrados o Laravel Octane e a extensão Swoole, enquanto no segundo foi mantida a configuração padrão do Laravel, representando assim a versão convencional. Ambas as aplicações foram uniformizadas em termos de estrutura, utilizando páginas web idênticas em HTML, CSS e JS. Além disso, as rotas e demais configurações foram mantidas consistentes entre as aplicações para garantir uma base de comparação justa.

### 4.3.2 Arquitetura do projeto desenvolvido

A arquitetura do projeto foi construída de uma forma a obter os resultados a serem analisados e comparados. A Figura 9 proporciona uma visão abrangente da arquitetura implementada, evidenciando a coexistência de ambas as aplicações em um único servidor. Nessa representação visual, destacam-se também, as diferenças de cada aplicação.

Figura 9 – Arquitetura do projeto para a realização dos testes

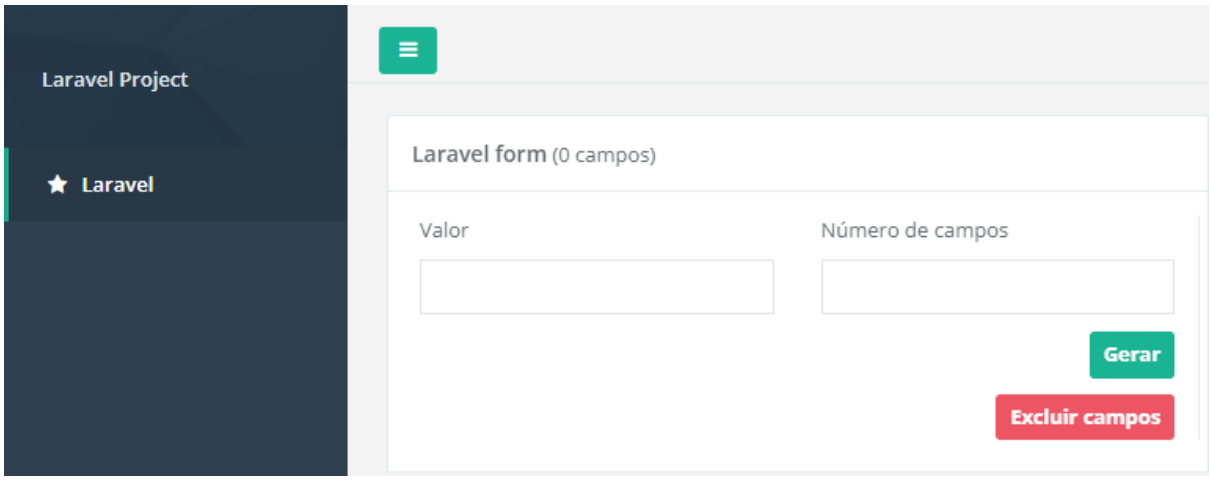



Fonte: Do Autor (2023).

### 4.3.3 Representação visual das aplicações

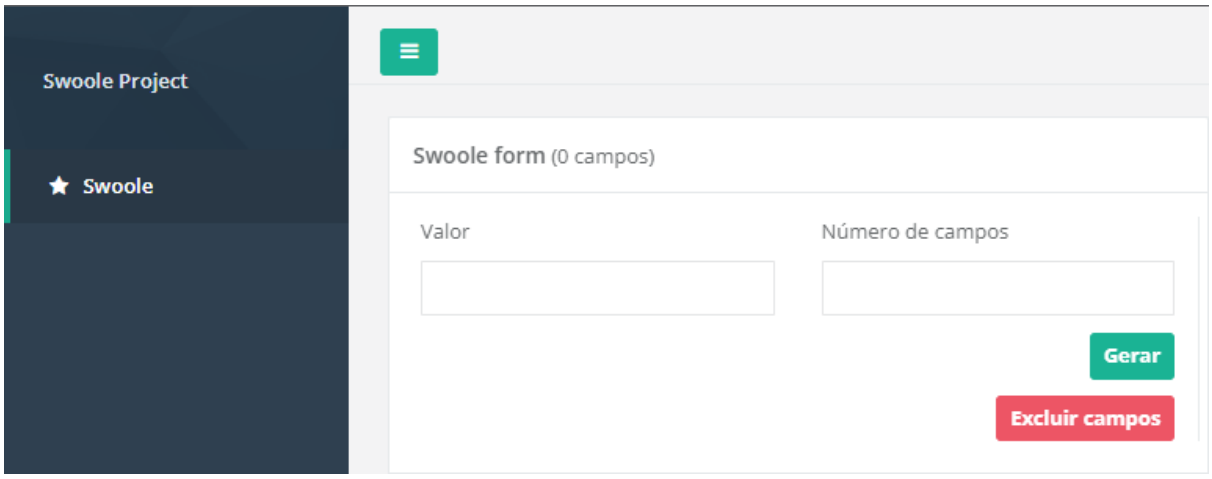
Para proporcionar uma visualização clara e distintiva de cada aplicação, foi desenvolvido um formulário exclusivo para cada uma, incluindo um identificador posicionado no canto superior esquerdo e no cabeçalho do formulário, conforme ilustrado nas Figuras 10 e 11. Este identificador destaca a aplicação específica a ser testada. Na aplicação correspondente ao Laravel convencional, foi atribuído o identificador “Laravel”, enquanto na representação da aplicação do Laravel Octane com Swoole, cada formulário foi identificado como “Swoole”.

Figura 10 – Formulário da aplicação Laravel




Fonte: Do Autor (2023).

Figura 11 – Formulário da aplicação Laravel Octane com Swoole




Fonte: Do Autor (2023).

### 4.3.4 Etapas do Desenvolvimento das Aplicações

O desenvolvimento das aplicações foi conduzido de maneira sistemática, seguindo uma série de etapas definidas. A seguir, apresentamos um passo a passo detalhado do processo:

1. **Aquisição e Configuração do Servidor**
    - Inicialmente, foi solicitado um servidor para o setor Laboratório de Automação da Univates, com a especificação de que deveria ser um servidor Ubuntu Server 22.04 LTS.

- Configurações do servidor: 1 CPU com 2 núcleos, 4 GB RAM e 30 GB de armazenamento.
- Após a entrega do servidor, foram realizadas as configurações iniciais necessárias para prepará-lo para o ambiente de desenvolvimento.

2. **Configuração das Aplicações**
    - Seguindo a documentação oficial do Laravel, foram iniciados dois projetos separados para as aplicações convencional e Laravel Octane com Swoole.

3. **Instalação e Configuração do Banco de Dados PostgreSQL**
    - Foi instalado e configurado o sistema de gerenciamento de banco de dados PostgreSQL no servidor.
    - As bases de dados necessárias para cada aplicação foram criadas no PostgreSQL.

4. **Desenvolvimento das Aplicações Laravel Convencional e Octane com Swoole**
    - Utilizando o conceito de migrations, foram criadas as estruturas de banco de dados necessárias para cada aplicação.
    - As Models, Views e Controllers foram desenvolvidos para ambas as aplicações, conforme as necessidades específicas de cada uma.
    - Rotas foram configuradas para permitir o correto direcionamento das requisições HTTP.

5. **Ajustes no Formulário**
    - O formulário foi projetado com dois campos: um campo de texto identificado como “Valor” e um campo numérico identificado como “Número de campos”.
    - A lógica de envio do formulário foi implementada através do botão "Gerar".
    - Utilizando JavaScript, as informações do formulário foram enviadas via Ajax para o back-end.

6. **Implementação do Backend para salvar no Banco de Dados**
    - No back-end, no Controller de cada aplicação, o código foi desenvolvido para tratar os dados recebidos e efetuar a inserção no banco de dados.
    - Na aplicação convencional, a inserção ocorre sequencialmente, enquanto na aplicação do Laravel Octane com Swoole, a inserção é realizada de forma paralela.

7. **Ajustes na Aplicação Laravel Octane com Swoole**
    - Seguindo a documentação do Laravel Octane, as extensões necessárias do Laravel Octane e do Swoole foram configuradas na aplicação correspondente.
    - Isso possibilitou a execução de operações de forma paralela, melhorando o desempenho da aplicação em cenários de carga elevada.

8. **Processo de Envio do Formulário**
    - O envio do formulário é acionado pelo botão "Gerar", executando um código JavaScript que realiza o envio assíncrono das informações via Ajax.
    - Os dados são processados no back-end, no Controller, adaptado para salvar no banco de dados, levando em consideração as especificidades de cada aplicação.

Essas etapas delineiam o caminho percorrido desde a configuração inicial do servidor até a implementação e otimização das aplicações, proporcionando um panorama abrangente do desenvolvimento realizado.

# 5 TESTES E ANÁLISE DOS RESULTADOS

Nos testes conduzidos para otimizar aplicações web, foram desenvolvidos dois projetos utilizando Laravel Octane com Swoole e Laravel convencional. Esta seção explora os resultados desses projetos, focando na dinâmica de inserção de dados e nos intervalos de tempo associados ao armazenamento dessas informações no banco de dados. Essas análises foram realizadas sob diferentes condições de carga de trabalho, proporcionando informações sobre o desempenho relativo das duas abordagens.

## 5.1 Metodologia dos Testes

Como mencionado no capítulo anterior, a metodologia dos testes consistiu na criação de um formulário padrão para ambos os projetos, garantindo uniformidade nos dados de entrada. Antes do envio do formulário, é realizada uma lógica via JavaScript para montar um array referente ao número de campos de texto a serem enviados, conforme detalhado na Figura 12.

Figura 12 – Lógica para montar um array referente ao número de campos de texto a serem enviados

```
// Obter as informações dos campo "Valor" e "Número de campos"
var valor = $('[name="valor"]').val();
var numero_campos = $('[name="numero_campos"]').val();

// Criar um array com o valor repetido
var valoresRepetidos = new Array(parseInt(numero_campos)).fill(valor);
```

Fonte: Do Autor (2023)

Na Figura 12 é observada a lógica JavaScript, onde são capturadas as informações referentes aos campos “Valor” e “Número de campos” e em seguida criado um array do “Valor” x (vezes) “Número de campos”. Para Shoemaker (2018), a função “Array.fill” serve para popular um array em JavaScript, onde são passados os parâmetros referentes ao tamanho do array a ser gerado com base na string repassada.

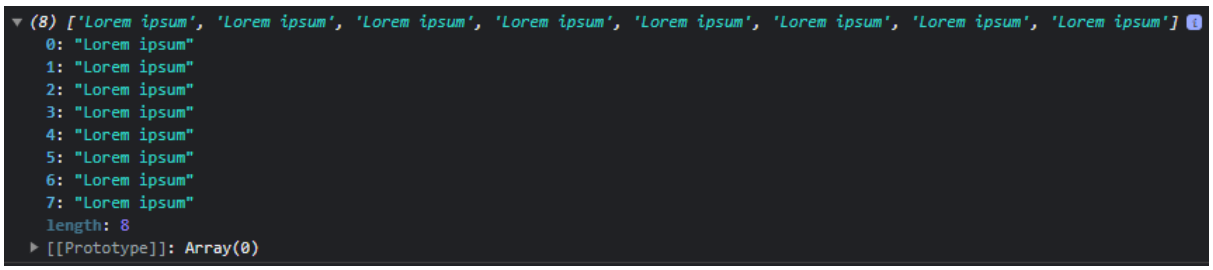
### 5.2 Estrutura do Formulário e Dados Gerados

O formulário foi estruturado da seguinte forma:

- Campo de texto para inserção de uma string.
- Campo numérico para indicar o número de vezes que a string será repetida.

A lógica de montagem do array é baseada no número inteiro inserido no campo "Número de campos". Por exemplo, se o campo de texto, identificado como "Valor", contiver a informação " Lorem ipsum" e o campo "Número de campos" estiver preenchido com o número "8", será multiplicado o conteúdo do campo de texto pelo valor inserido, criando assim, um array com 8 posições, todas preenchidas com a string "Lorem ipsum". O retorno pode ser observado na Figura 13.

Figura 13 – Array com 8 posições contendo a mesma string




Fonte: Do Autor (2023)

### 5.3 Execução dos Testes

Em seguida, foram realizados testes de inserção com uma carga crescente de dados, variando o número de registros (10, 100, 1000, 2000, 3500, 5000). Durante

cada teste, o tempo de armazenamento no banco de dados foi registrado, medindo o intervalo entre antes do início da inserção do primeiro registro e após a conclusão da inserção do último registro.

### 5.4 Resultados Obtidos

Nesta seção podem ser observados os resultados obtidos e comparados.

#### 5.4.1 Comparação visual de códigos

Nas Figuras 14 e 15, são apresentados trechos de código que refletem a ação de armazenar valores dos campos no banco de dados nas aplicações Laravel Convencional e Laravel Octane com Swoole. A diferença na construção do código é notável: no Laravel Convencional, há uma iteração padrão para armazenar os campos individualmente, enquanto no Laravel Octane a abordagem é diferente. Neste último, os registros são divididos em grupos de 1000, e a iteração ocorre de forma paralela, possibilitando salvar 1000 registros no banco de dados simultaneamente. Essa divergência destaca a eficiência do Octane em lidar com operações concorrentes, otimizando o processo de armazenamento de dados.

Figura 14 – Trecho de código de inserção do Laravel convencional

```
try {
    $campos = $request->input('valoresRepetidos');

    foreach ($campos as $campo) {
        Campo::create([
            'valor' => $campo
        ]);
    }
}
```

Fonte: Do Autor (2023)

Figura 15 – Trecho de código de inserção do Laravel Octane com Swoole

```php
try {
    $campos = $request->input('valoresRepetidos');

    $tasks = [];

    // Divide os campos em grupos de 1000
    $gruposDeCampos = array_chunk($campos, 1000);

    // Executa cada grupo de tarefas em paralelo
    foreach ($gruposDeCampos as $grupoDeCampos) {
        $tasks = [];

        foreach ($grupoDeCampos as $campo) {
            $tasks[] = function () use ($campo) {
                Campo::create([
                    'valor' => $campo
                ]);
            };
        }

        Octane::concurrently($tasks);
    }
}
```

Fonte: Do Autor (2023)

### 5.4.2 Tabelas de Resultados dos testes realizados

No Quadro 2 pode ser observado uma comparação entre o número de registros inseridos de cada aplicação. Para este teste foi feita a seguinte configuração na aplicação contendo o Laravel Octane:

- Campos em grupos: 1000.
- Tamanho da string de cada campo: 26.
- Workers: 4.
- Task-workers: 4.

Quadro 2 – Quadro comparativo entre as duas aplicações com base no número de registros e no tempo para armazenar no banco de dados

| **Número de Registros** | **Laravel Convencional**<br><br>**Tempo (ms)** | **Laravel Octane com Swoole**<br><br>**Tempo (ms)** |
|---|---|---|
| 10 | 0.027 | 0.035 |
| 100 | 0.255 | 0.285 |
| 1000 | 2.385 | 2.027 |
| 2000 | 4.672 | 3.026 |
| 3500 | 8.046 | 6.872 |
| 5000 | 11.951 | 9.926 |
| 7000 | 13.432 | 11.413 |
| 10000 | 24.670 | 13.689 |
| 20000 | 47.513 | 26.802 |
| 32500 | 76.795 | 45.905 |

Fonte: Do Autor (2023)

No Quadro 3 pode ser observado uma comparação entre o número de registros inseridos na aplicação do Laravel Octane com Swoole, realizando diferentes configurações.

1. **Configuração 1**
    - Campos em grupos: 1000.
    - Tamanho da string de cada campo: 26.
    - Workers: 4.
    - Task-workers: 4.

2. **Configuração 2**
    - Campos em grupos: 500.
    - Tamanho da string de cada campo: 26.
    - Workers: 6.
    - Task-workers: 6.

3. **Configuração 3**
    - Campos em grupos: 1000.
    - Tamanho da string de cada campo: 26.
    - Workers: 8.
    - Task-workers: 8.

4. **Configuração 3**
    - Campos em grupos: 1000.
    - Tamanho da string de cada campo: 71.
    - Workers: 8.
    - Task-workers: 8.

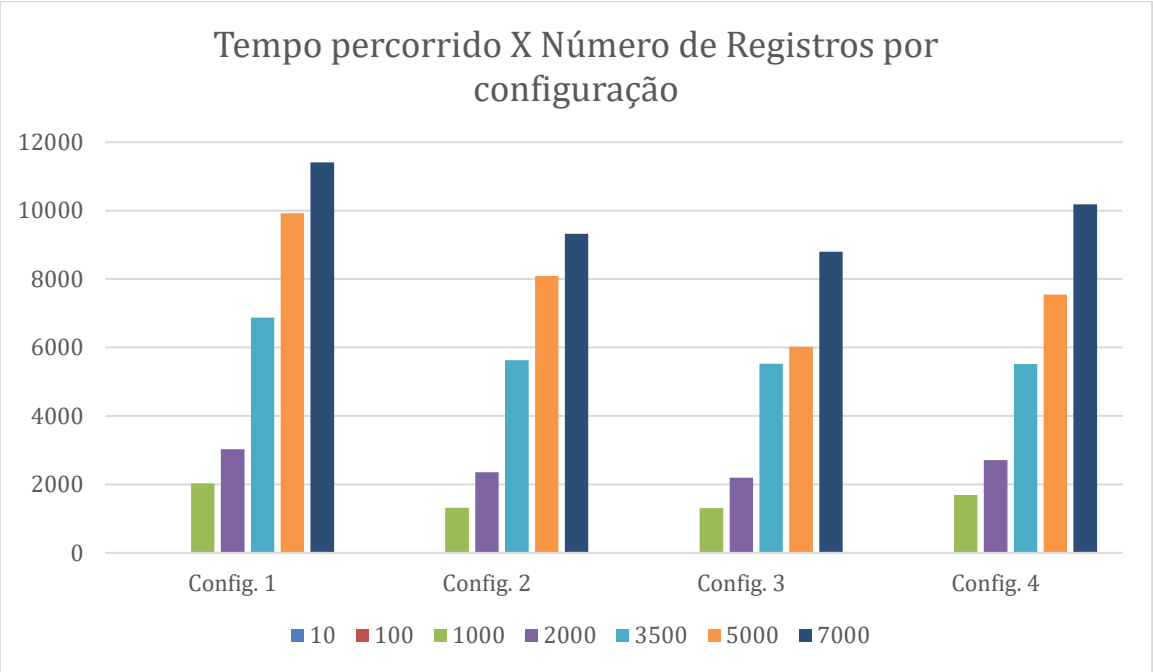
Quadro 3 – Quadro comparativo entre as diferentes configurações da aplicação Laravel Octane com Swoole

| **Número de Registros** | **Config. 1**<br><br>**Tempo (ms)** | **Config. 2**<br><br>**Tempo (ms)** | **Config. 3**<br><br>**Tempo (ms)** | **Config. 4**<br><br>**Tempo (ms)** |
|---|---|---|---|---|
| 10 | 0.035 | 0.031 | 0.028 | 0.028 |
| 100 | 0.285 | 0.221 | 0.208 | 0.229 |
| 1000 | 2.027 | 1.322 | 1.309 | 1.694 |
| 2000 | 3.026 | 2.354 | 2.200 | 2.708 |
| 3500 | 6.872 | 5.632 | 5.527 | 5.518 |
| 5000 | 9.926 | 8.091 | 6.021 | 7.542 |
| 7000 | 11.413 | 9.327 | 8.798 | 10.182 |

Fonte: Do Autor (2023)

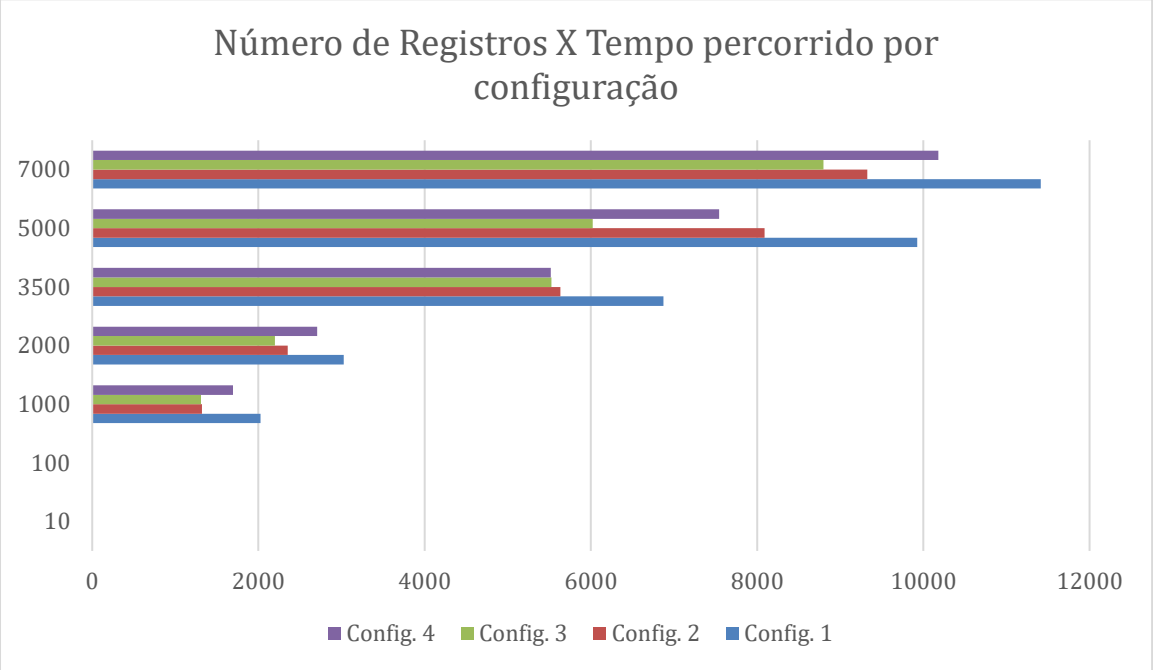
Na Figura 16 pode ser visualizado em formato de gráfico os valores da tabela, onde Y é o tempo em milissegundos percorrido e X as configurações com a quantidade de registro de cada uma.

Figura 16 – Gráfico Tempo percorrido X Número de Registros




Fonte: Do Autor (2023)

Figura 17 - Gráfico Número de Registros X Tempo percorrido




Fonte: Do Autor (2023)

A análise dos resultados dos testes com as configurações específicas do Laravel Octane revela uma performance notável, conforme analisado no Quadro 3 e

nas Figuras 16 e 17, especialmente em comparação com o Laravel Convencional. Aqui estão alguns pontos destacados:

1. **Divisão de Campos em Grupos de 1000:**
    - Ao dividir os campos em grupos de 1000 para serem inseridos de forma paralela, observa-se uma estratégia eficiente para lidar com grandes volumes de dados. Isso contribui para a capacidade do Octane de realizar operações de forma assíncrona, otimizando o tempo de resposta.

2. **Relação de Desempenho a Partir de 1000 Registros:**
    - A partir de 1000 registros, torna-se evidente que o Laravel Octane demonstra um desempenho superior em termos de tempo de resposta em comparação com o Laravel Convencional. Mesmo com o tempo de espera de 3 segundos entre os grupos de campos, o Octane consegue manter um tempo de salvamento menor, indicando sua capacidade de processar operações de forma mais eficiente.

3. **Observação da Assincronicidade:**
    - A abordagem assíncrona do Laravel Octane, potencializada pela integração com o Swoole, destaca-se na execução paralela de operações. Isso é especialmente benéfico em situações de carga mais intensa, onde a capacidade de realizar várias operações simultaneamente resulta em tempos de resposta mais rápidos.

4. **Configuração de Workers e Task-workers (4 cada):**
    - A configuração de 4 workers e 4 task-workers parece ser eficaz, proporcionando um equilíbrio entre paralelismo e uso eficiente de recursos. Essa configuração contribui para a melhoria do desempenho, permitindo que múltiplas tarefas sejam executadas de maneira concorrente.

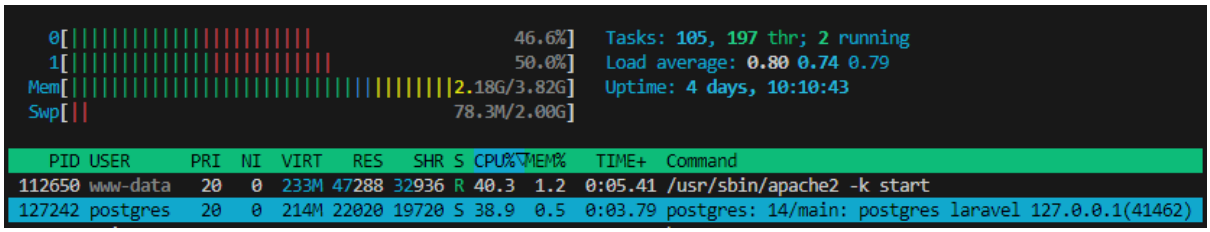
Em resumo, os resultados destacam a eficácia do Laravel Octane em situações de inserção de dados, especialmente em cenários com muitos registros. A estratégia

de divisão em grupos, combinada com a abordagem assíncrona e as configurações específicas, resulta em um desempenho notavelmente superior em comparação com o Laravel Convencional.

### 5.4.3 Análise Detalhada do uso de recursos da máquina

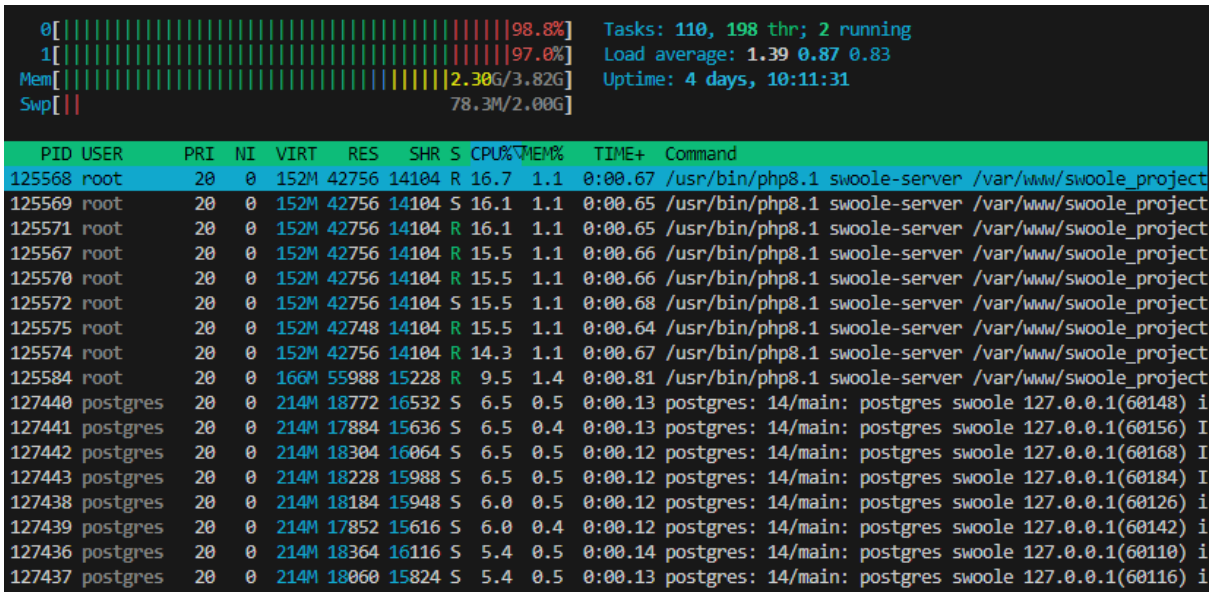
Análise detalhada referente ao uso do comando htop para analisar as métricas de uso de CPU e memória durante a inserção de cargas de registros através do formulário desenvolvido, conforme as figuras disponibilizadas nessa seção.

Figura 18 – Uso de CPU no Laravel Convencional envio de 10.000 registros



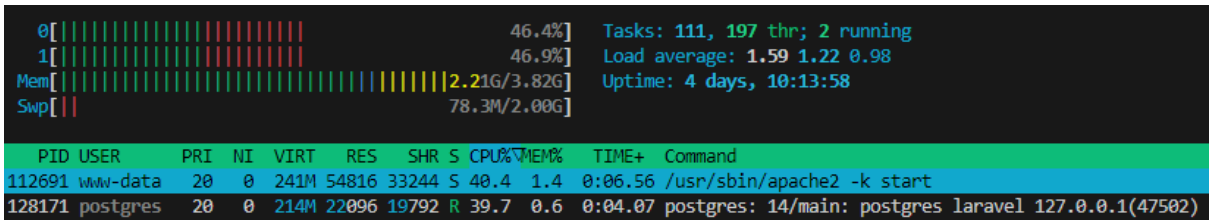

Fonte: Do Autor (2023)

Figura 19 – Uso de CPU no Laravel Octane com Swoole envio de 10.000 registros



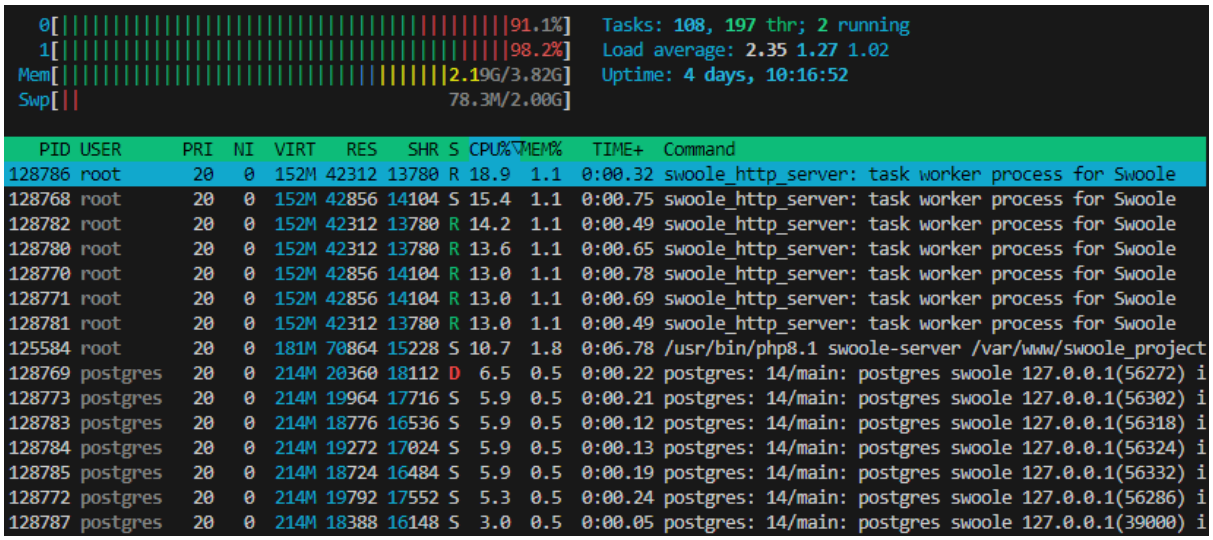

Fonte: Do Autor (2023)

Figura 20 – Uso de CPU no Laravel Convencional envio de 20.000 registros




Fonte: Do Autor (2023)

Figura 21 – Uso de CPU no Laravel Octane com Swoole envio de 20.000 registros




Fonte: Do Autor (2023)

A investigação do desempenho da aplicação durante a inserção de registros revelou insights cruciais ao acompanhar o utilitário Htop, conforme nas figuras. Notavelmente, independentemente da carga de dados, a aplicação Laravel Convencional demonstrou uma distribuição equitativa de uso de CPU, variando entre 40% e 50% para cada núcleo do servidor. Em contraste, a aplicação Laravel Octane com Swoole apresentou uma média persistente acima dos 90% de uso em cada núcleo.

Esse comportamento distinto é atribuído à natureza do Swoole, que instancia vários processos em paralelo para lidar com operações de forma assíncrona e paralela. O aumento significativo no uso de CPU observado na aplicação Octane reflete a eficiência do Swoole em otimizar o processamento concorrente, maximizando a utilização dos recursos disponíveis.

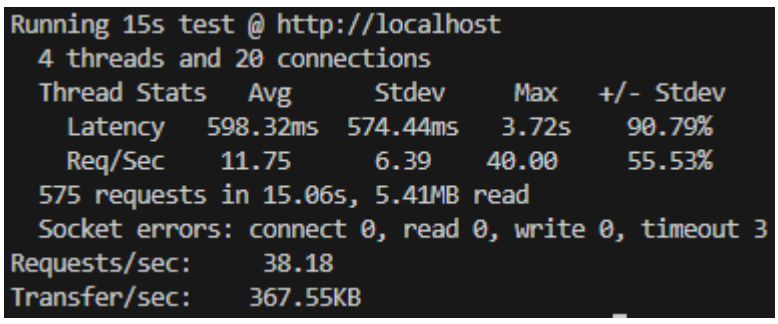
Vale destacar que o uso de memória não apresentou mudanças significativas, conforme observado nas figuras referentes às duas aplicações. Este aspecto é relevante, pois indica uma estabilidade no consumo de memória, independentemente do modelo de aplicação utilizado. A consistência nesse aspecto sugere que, mesmo com variações na carga de dados, a gestão de memória permaneceu eficiente em ambos os cenários, contribuindo para a estabilidade operacional das aplicações.

A análise detalhada do Htop proporcionou uma visão aprofundada do impacto das tecnologias adotadas no desempenho do servidor, fundamentando considerações críticas para a escolha entre Laravel Convencional e Laravel Octane com Swoole. Esses resultados não apenas destacam a eficácia do Swoole em cenários de alta concorrência, mas também evidenciam a importância de uma abordagem contextualizada ao considerar o uso de recursos em diferentes contextos de aplicação.

### 5.4.4 Testes de benchmarking com Wrk

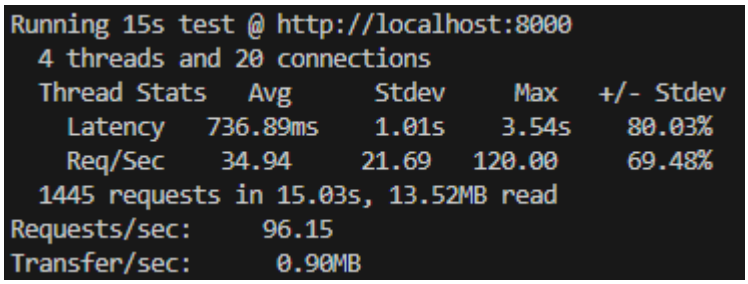
Utilizando a ferramenta de benchmarking wrk, são exploradas métricas essenciais, como taxas de requisição e latência. Esses testes proporcionam algumas informações sobre a eficiência e capacidade de resposta do sistema em diferentes cenários, contribuindo para uma avaliação do desempenho do Laravel Octane com Swoole em comparação com o Laravel convencional.

Figura 22 – 1° teste com a ferramenta wrk na aplicação do Laravel convencional



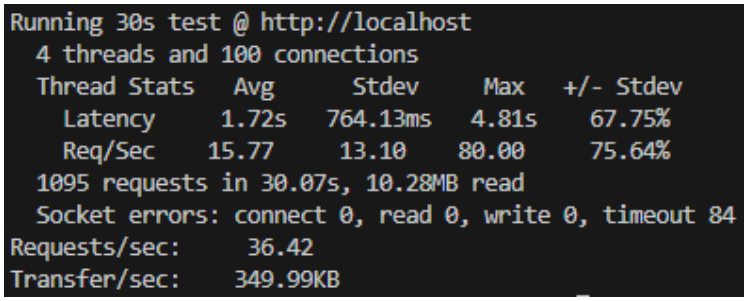

Fonte: Do Autor (2023)

Figura 23 – 1° teste com a ferramenta wrk na aplicação do Laravel Octane



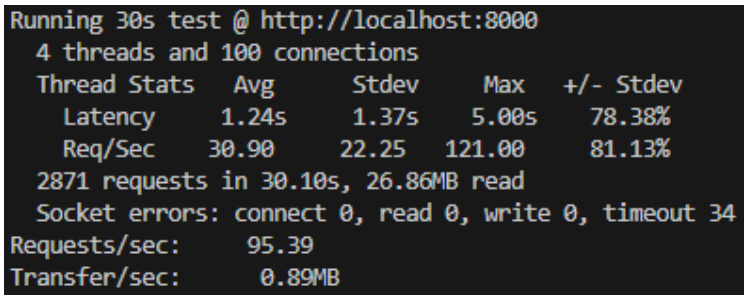

Fonte: Do Autor (2023)

Figura 24 – 2° teste com a ferramenta wrk na aplicação do Laravel convencional



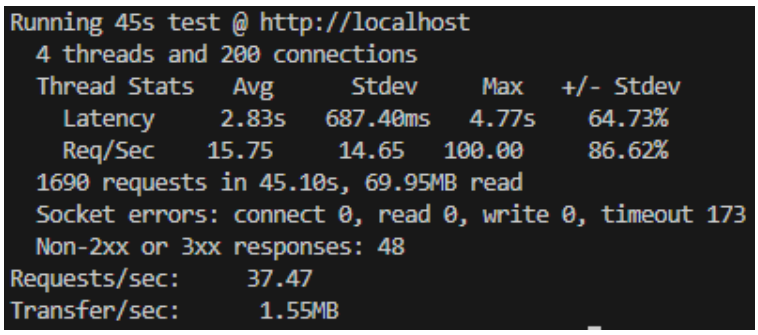

Fonte: Do Autor (2023)

Figura 25 – 2° teste com a ferramenta wrk na aplicação do Laravel Octane



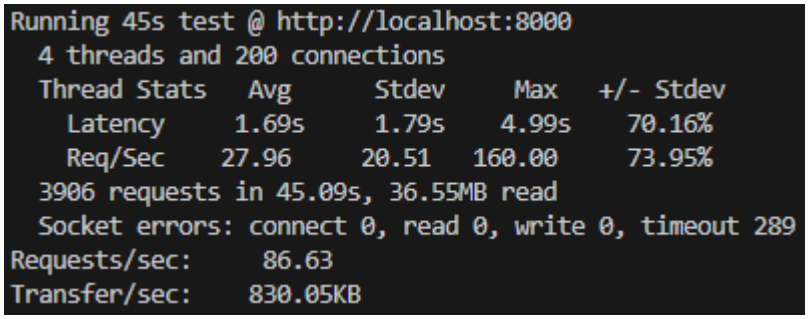

Fonte: Do Autor (2023)

Figura 26 – 3° teste com a ferramenta wrk na aplicação do Laravel convencional




Fonte: Do Autor (2023)

Figura 27 – 3° teste com a ferramenta wrk na aplicação do Laravel Octane




Fonte: Do Autor (2023)

Quadro 4 - Quadro de Desempenho: Laravel Convencional vs. Laravel Octane com Swoole em diferentes configurações

| **Teste** | **Informações** | **Laravel Convencional** | **Laravel Octane com Swoole** |
|---|---|---|---|
| **1° Teste** | **Conexões (-c)** | 20 | 20 |
| | **Req. Totais** | 575 | 1445 |
| | **Req/Seg** | 38.18/s | 96.15/s |
| | **Timeout** | 3 | 0 |
| | **Latência Média** | 598.32ms | 736.89ms |
| | | | |
| **2° Teste** | **Conexões (-c)** | 100 | 100 |
| | **Req. Totais** | 1095 | 2871 |
| | **Req/Seg** | 37.38/s | 95.39/s |
| | **Timeout** | 85 | 34 |
| | **Latência Média** | 1.68s | 1.24s |
| | | | |
| **3° Teste** | **Conexões (-c)** | 200 | 200 |
| | **Req. Totais** | 1690 | 3906 |
| | **Req/Seg** | 36.03/s | 86.63/s |
| | **Timeout** | 180 | 289 |
| | **Latência Média** | 2.92s | 1.69s |

Fonte: Do Autor (2023)

Ao analisar o Quadro 4, construído a partir das Figuras 22, 23, 24, 25, 26 e 27, pode-se concluir que em todas as configurações, o Laravel Octane com Swoole demonstra uma taxa de requisições por segundo significativamente superior em comparação com o Laravel Convencional. Observa-se que, à medida que é aumentado o número de conexões e o tempo de teste, o número de requisições também aumenta. Esse aumento na carga de trabalho pode impactar negativamente o desempenho, resultando em um maior número de timeouts. No caso específico do Laravel Octane com Swoole, é interessante notar que, mesmo com um aumento na carga, ele geralmente mantém uma latência menor em comparação com o Laravel Convencional.

O aumento no número de timeouts em configurações mais exigentes pode ser atribuído à sobrecarga do sistema, onde o servidor pode não conseguir lidar eficientemente com todas as conexões simultâneas, resultando em algumas requisições expirando antes de serem concluídas.

# 6 CONSIDERAÇÕES FINAIS

Com base nas análises e descobertas apresentadas neste estudo sobre a otimização de aplicações web PHP, concentrada em uma análise comparativa entre o Laravel Convencional e o Laravel Octane com Swoole, é evidente que a escolha correta das tecnologias desempenha um papel crucial na melhoria do desempenho das aplicações. A busca contínua por aprimoramento no desempenho de aplicações web PHP é essencial para atender às demandas crescentes da era digital. Os resultados obtidos fornecem percepções significativas sobre as vantagens e considerações ao adotar diferentes abordagens tecnológicas.

Ao explorar a literatura existente e realizar testes práticos, foi possível contextualizar a importância da otimização no cenário PHP. A implementação do Laravel Octane com Swoole revelou-se uma estratégia promissora, proporcionando eficiência notável no processamento assíncrono e na gestão de tarefas concorrentes. Essa abordagem resultou em tempos de resposta mais rápidos e escalabilidade aprimorada em comparação com a abordagem convencional. Contudo, conforme detalhado na seção "Análise Detalhada do Uso de Recursos da Máquina", observou-se uma desvantagem notável: o aumento significativo no uso de CPU. Enquanto a aplicação convencional manteve uma média de utilização de CPU entre 40% a 50%, a implementação com Swoole manteve uma média acima dos 90% de uso em cada núcleo do servidor. Assim, embora o Laravel Octane com Swoole ofereça vantagens em determinados aspectos, é importante considerar essa elevação no consumo de CPU como um ponto importante a ser considerado ao escolher entre as abordagens de implementação.

Este contexto ressalta a natureza dinâmica da otimização de aplicações web PHP, que requer estratégias flexíveis, testes cuidadosos e uma compreensão profunda das particularidades de cada tecnologia estudada.

Contudo, é crucial observar que, em cenários de carga extrema, alguns desafios foram identificados, incluindo um aumento proporcional no número de timeouts no Laravel Octane. Essa observação enfatiza a necessidade de considerar cuidadosamente as configurações e os requisitos específicos da aplicação, além dos recursos do servidor, ao escolher entre as opções disponíveis.

Assim como nos exemplos apresentados, este estudo reconhece a importância da colaboração com profissionais da área, neste caso, desenvolvedores e especialistas em otimização de desempenho. O engajamento ativo desses profissionais foi fundamental para entender as nuances das tecnologias aplicadas e garantir a eficácia das estratégias adotadas.

É importante considerar esses resultados no contexto das necessidades das aplicações. A escolha entre o Laravel Convencional e o Laravel Octane com Swoole deve levar em conta diversos fatores, incluindo características específicas da aplicação, os recursos disponíveis no servidor, os requisitos de desempenho desejados e as demandas esperadas de tráfego. A otimização do número de conexões simultâneas e a afinidade com os recursos do servidor podem ser áreas de foco para melhorar ainda mais o desempenho em cenários de carga mais pesada.

Considerando as limitações do estudo, é importante ressaltar que os resultados obtidos estão vinculados às especificidades dos recursos do servidor utilizado nas análises. A máquina virtual em questão, dotada de 1 CPU com 2 núcleos, 4GB de RAM e 30GB de armazenamento, pode influenciar diretamente no desempenho observado. Cabe mencionar que a escolha desses recursos pode impactar a capacidade de lidar com cargas extremas, e variações nos resultados podem ocorrer em ambientes com configurações distintas. Assim, para uma compreensão mais abrangente e aplicável em diferentes contextos, seria benéfico realizar testes em servidores com características diversas, considerando uma gama mais ampla de cenários e demandas específicas de diferentes aplicações. Essa

consideração ressalta a importância de adaptar as conclusões às condições específicas de cada ambiente de implementação.

Em relação aos objetivos estabelecidos e aos resultados alcançados, é evidente que o objetivo geral deste estudo, foi atingido. O trabalho demonstrou melhorias significativas no desempenho e na escalabilidade com o uso do Laravel Octane com Swoole, cumprindo com os objetivos específicos de avaliar o desempenho e analisar a escalabilidade. Além disso, foi possível medir e comparar o consumo de recursos do servidor entre as duas configurações, abordando o terceiro objetivo específico.

Em síntese, este estudo enfatiza a centralidade da otimização para alcançar padrões superiores de desempenho em ambientes web PHP. A busca contínua por aprimoramento no desempenho de aplicações web PHP é essencial para atender às demandas crescentes da era digital.

Como direção para trabalhos futuros, sugere-se uma análise mais aprofundada das configurações específicas que impactam o aumento de timeouts no Laravel Octane em cargas extremas, bem como a exploração de técnicas avançadas de ajuste para otimizar ainda mais o desempenho em diferentes cenários de aplicação.

# REFERÊNCIAS


ANTON, Konrad; THIEMANN, Peter. **Typing coroutines**. In: International Symposium on Trends in Functional Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 16-30.

CAO, Bangzhong; SHI, Minyong; LI, Chunfang. **The solution of web font-end performance optimization**. In: 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI). IEEE, 2017. p. 1-5.

ELIZAROV, Roman et al. **Kotlin coroutines: design and implementation**. In: Proceedings of the 2021 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. 2021. p. 68-84.

FILHO, Milton Cordeiro Farias; FILHO, Emílio J. M. Arruda. **Planejamento da pesquisa científica**. 2. ed. – São Paulo: Atlas, 2015.

FLANAGAN, David; NOVAK, Gregor M. **JavaScript: The Definitive Guide, Sixth Edition**. 2011.

KRASNER, Glenn E.; POPE, Stephen T. **A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80**, 1988. Disponível em: https://ics.uci.edu/~redmiles/ics227-SQ04/papers/KrasnerPope88.pdf. Acesso em: 22 nov. 2023

KUMAR, Rahul. **htop Command in Linux**. Tecadmin, 2022. Disponível em: https://tecadmin.net/htop-command-in-linux/#:~:text=The%20htop%20command%20is%20like,run%20strace%20on%20any%20process. Acesso em: 23 nov 2023.

LAAZIRI, Majida et al. **A Comparative study of PHP frameworks performance**. Procedia Manufacturing, v. 32, p. 864-871, 2019. Disponível em: https://www.sciencedirect.com/science/article/pii/S2351978919303312?pes=vor. Acesso em: 6 out. 2023.

LEFF, Avraham; RAYFIELD, James T. **Web-application development using the model/view/controller design pattern**. In: Proceedings fifth ieee international enterprise distributed object computing conference. IEEE, 2001. p. 118-127.

MOURA, Ana Lúcia De; IERUSALIMSCHY, Roberto. **Revisiting coroutines**. PUC-RioInf.MCC15/04 June, 2004. Disponível em: https://www.inf.puc-rio.br/~roberto/docs/MCC15-04.pdf. Acesso em: 22 nov 2023.

MILADEV. **Php swoole**. Medium, 2023. Disponível em: https://medium.com/@miladev95/php-swoole-4476df565e33. Acesso em: 19 dez. 2023.

OLANREWAJU, Rashidah F.; ISLAM, Thouhedul; ALI, Nor'ashikin. **An empirical study of the evolution of PHP MVC framework**. In: Advanced Computer and Communication Engineering Technology: Proceedings of the 1st International Conference on Communication and Computer Engineering. Springer International Publishing, 2015. p. 399-410. Disponível em: https://sci-hub.se/10.1007/978-3-319-07674-4_40. Acesso em: 6 out. 2023.

POP, Dragos-Paul; ALTAR, Adam. **Designing an MVC model for rapid web application development**. Procedia Engineering, v. 69, p. 1172-1179, 2014.

RUNCLOUD TEAM. **Laravel Octane – What It Is, Why It Matters & How To Take Advantage Of It**. RunCloud, 2022. Disponível em: https://blog.runcloud.io/laravel-octane/. Acesso em 20 nov 2023.

SHOEMAKER, Sophia. **4 Ways to Populate an Array in JavaScript**. Medium, 2018. Disponível em: https://medium.com/@wisecobbler/4-ways-to-populate-an-array-in-javascript-836952aea79f. Acesso em: 23 nov 2023.

SILVA, Felipe Dutra Tine. **Intelligent benchmark with wrk**. Medium, 2018. Disponível em: https://medium.com/@felipedutratine/intelligent-benchmark-with-wrk-163986c1587f. Acesso em: 23 nov 2023

SOUZA, Vinicius. **O comando HTOP no Linux 2022**. Blog Ironlinux, 2022. Disponível em: https://blog.ironlinux.com.br/o-comando-htop-no-linux/. Acesso em: 23 nov 2023.

TEDESCO, Kennedy. **Introdução ao Swoole, framework PHP assíncrono baseado em corrotinas**. Treinaweb, 2019. Disponível em: https://www.treinaweb.com.br/blog/introducao-ao-swoole-framework-php-assincrono-baseado-em-corrotinas/. Acesso em: 22 nov 2023.

YANG, Jiajie; SHAN, Zhihao; CHEN, Zhengming. **Research and practice of swoole asynchronous multithreading design method**. In: 2018 IEEE 4th International Conference on Computer and Communications (ICCC). IEEE, 2018. p. 2163-2169. Disponível em: https://sci-hub.se/10.1109/compcomm.2018.8780934. Acesso em: 6 out. 2023.

YAO, Yu; XIA, Jie. **Analysis and Research on the performance Optimization of Web Application system in high Concurrency Environment**. In: 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference. IEEE, 2016. p. 321-326.