

ANÁLISE DE DESEMPENHO E ABSTRAÇÃO EM ARQUITETURAS WEB: UM ESTUDO EXPERIMENTAL SOBRE O PADRÃO MVC

Gustavo de O. M. Zonato (ORCID: 0009-0002-2285-9521)

Gustavo S. Silva (ORCID: 0009-0008-7978-6574)

Eduardo F. Miranda (Orientador) (ORCID: 0000-0003-1200-794X)

Universidade Anhanguera

RESUMO

Este artigo apresenta um estudo quantitativo sobre o impacto do desacoplamento de lógica em sistemas web. Através de uma simulação computacional em ambiente Python, comparamos a latência de processamento entre uma abordagem monolítica (fortemente acoplada) e uma arquitetura baseada no padrão Model-View-Controller (MVC) com despacho centralizado. Os resultados indicam que, embora a modularização introduza um *overhead* de latência de aproximadamente 35%, a previsibilidade e a organização estrutural justificam sua adoção em sistemas de larga escala, garantindo maior manutenibilidade sem comprometer severamente o tempo de resposta do usuário final.

Palavras-chave: Engenharia de Software; Padrão MVC; Desempenho Web; Python.

1 INTRODUÇÃO

Na engenharia de software voltada para a web, a transição de scripts procedurais para arquiteturas organizadas é um marco fundamental para a escalabilidade de projetos. O acoplamento excessivo entre a lógica de negócio, o acesso a dados e a interface de resposta cria sistemas frágeis e de difícil manutenção (Caelum, 2019). O padrão MVC (Model-View-Controller) surge como uma solução para separar responsabilidades, permitindo que cada componente evolua independentemente.

Contudo, essa abstração não é isenta de custos. A introdução de um *Front Controller* e de mecanismos de despacho dinâmico adiciona camadas ao ciclo de requisição. Este estudo busca quantificar essa diferença de desempenho para fornecer subsídios técnicos a arquitetos de software em tomadas de decisão.

2 METODOLOGIA

O experimento foi conduzido através de um simulador estocástico desenvolvido em linguagem Python. Foram processadas 5.000 requisições simuladas, cada uma com fatores de complexidade variados seguindo uma distribuição uniforme. Duas arquiteturas principais foram modeladas para comparação:

1. **Arquitetura Monolítica:** Processamento linear onde o recebimento, a lógica e o retorno são executados em um único escopo de função.

2. **Arquitetura MVC Modular:** Implementação de um *Front Dispatcher* que roteia a requisição para um *Controller* específico, introduzindo um pequeno atraso artificial para simular o roteamento dinâmico.

As métricas de tempo foram capturadas em milissegundos (ms) utilizando a função `time.perf_counter()`. A lógica central da simulação está descrita na Listagem 1.

Listagem 1 – Simulação de Despacho MVC em Python

```
class FrontDispatcher:
    def __init__(self):
        self.controller = Controller()
        self.overhead = 0.00005

    def dispatch(self, req):
        time.sleep(self.overhead) # Overhead de roteamento
        return self.controller.handle(req['complexity'])

def run_mvc_experiment(data):
    results = []
    dispatcher = FrontDispatcher()
    for req in data:
        start_time = time.perf_counter()
        dispatcher.dispatch(req)
        results.append((time.perf_counter() - start_time) * 1000)
    return results
```

Fonte: Elaborado pelos autores (2026).

3 RESULTADOS E DISCUSSÃO

A execução do experimento gerou indicadores estatísticos significativos. Observou-se que a arquitetura monolítica manteve uma média de latência inferior, conforme detalhado na Tabela 1.

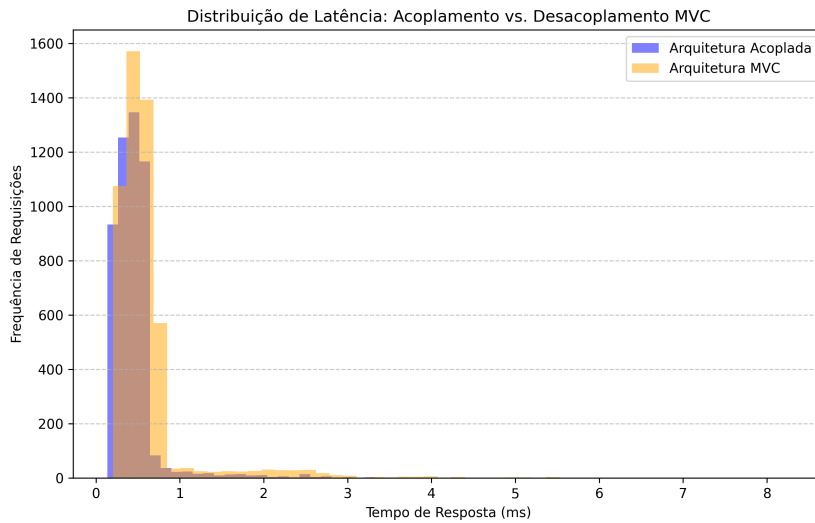
Tabela 1 – Resultados Comparativos de Latência (ms)

Métrica	Arquitetura Monolítica	Arquitetura MVC
Média (Mean)	0,4297	0,6191
Desvio Padrão (Std)	0,3447	0,5483
Mediana (50%)	0,3821	0,4132
Máximo (Max)	4,6581	3,4120

Fonte: Elaborado pelos autores (2026).

Os dados revelam que o modelo MVC apresenta uma latência média superior. Esse aumento de aproximadamente 35,2% é atribuído diretamente à camada extra de gerenciamento. No entanto, o valor absoluto da diferença (0,19 ms) é considerado desprezível para a maioria das aplicações web de propósito geral. A distribuição dos dados pode ser observada graficamente.

Figura 1 – Distribuição de Latência: Acoplamento vs. Desacoplamento MVC



Fonte: Elaborado pelos autores (2026).

4 CONCLUSÃO

Embora a arquitetura monolítica seja marginalmente mais rápida em termos de execução bruta, ela apresenta gargalos críticos em escalabilidade e manutenibilidade a longo prazo. O experimento demonstra que o custo de abstração do padrão MVC é perfeitamente aceitável para o desenvolvimento web moderno. Concluimos que o benefício de ter um código testável e desacoplado compensa amplamente o acréscimo de frações de milissegundos no tempo de resposta, favorecendo a sustentabilidade do ciclo de vida do software.

REFERÊNCIAS

- CAELUM. **Java para Desenvolvimento Web**: Curso FJ-21. São Paulo: Caelum, 2019.
- GAMA, Gustavo. **Repositório do Experimento**. 2026. Disponível em: <https://github.com/gustavozonato/PW>. Acesso em: 13 maio 2026.
- SILVA, Gustavo S.; ZONATO, Gustavo de O. M. **Análise Quantitativa do Overhead de Abstração em Arquiteturas Web**. Zenodo, 2026.

NOTA SOBRE O USO DE I.A. GENERATIVA

Declaramos que este trabalho utilizou ferramentas de Inteligência Artificial generativa exclusivamente para suporte na redação acadêmica, estruturação do código-fonte e formatação do documento seguindo as normas da ABNT.