TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

Analisis Perbandingan Performa Backend REST API Node.js, .NET, dan Laravel Octane Menggunakan Load Testing

Djoko Handoko[1,*], Galet Guntoro Setiaji[1], Ahmad Rifa'i[2]

[1]Fakultas Teknologi Informasi dan Komunikasi, Program Studi Teknik Informatika, Universitas Semarang, Kota Semarang, Indonesia
[2]Fakultas Teknologi Informasi dan Komunikasi, Program Studi Sistem Informasi, Universitas Semarang, Kota Semarang, Indonesia
Email: [1,*]djokohndk@gmail.com, [2]gallet@usm.ac.id, [3]rifai@usm.ac.id
Email Penulis Korespondensi: djokohndk@gmail.com

Abstrak–Meningkatnya kebutuhan aplikasi berbasis web yang mampu menangani banyak pengguna secara bersamaan menjadikan performa backend *Application Programming Interface* (API) sebagai aspek penting dalam menjamin responsivitas dan skalabilitas sistem. Perbedaan arsitektur serta model eksekusi pada setiap teknologi backend menyebabkan karakteristik performa yang dihasilkan juga berbeda. Selain itu, penelitian terdahulu umumnya masih menggunakan konfigurasi dan lingkungan pengujian yang bervariasi sehingga hasil perbandingan belum sepenuhnya merepresentasikan kondisi yang setara. Penelitian ini bertujuan untuk menganalisis dan membandingkan performa backend REST API pada Node.js, .NET, dan Laravel Octane dalam lingkungan pengujian yang terstandarisasi. Penelitian menggunakan pendekatan eksperimen kuantitatif dengan teknik *load testing* menggunakan Grafana K6 melalui skenario pengujian hingga 150 *virtual users* menggunakan permintaan HTTP GET. Parameter yang dievaluasi meliputi *throughput* atau *requests per second* (RPS), waktu respons (*latency*), *error rate*, serta penggunaan CPU dan memori. Hasil penelitian menunjukkan bahwa Node.js memperoleh performa terbaik dengan *throughput* sebesar 493,93 *requests per second* dan *latency* rata-rata 77,43 ms, diikuti .NET dengan *throughput* 324,94 *requests per second* dan *latency* 163,06 ms, sedangkan Laravel Octane memperoleh *throughput* 192,29 *requests per second* dengan *latency* 358,62 ms. Seluruh backend menunjukkan *error rate* sebesar 0%. Dari aspek efisiensi sumber daya, Node.js juga memiliki penggunaan CPU dan memori paling rendah dibandingkan .NET maupun Laravel Octane. Hasil penelitian menunjukkan bahwa perbedaan model eksekusi berpengaruh terhadap *throughput*, waktu respons, dan efisiensi penggunaan sumber daya, sehingga dapat menjadi pertimbangan dalam menentukan teknologi backend yang sesuai untuk pengembangan aplikasi dengan kebutuhan performa dan skalabilitas tinggi.

Kata Kunci: REST API; Node.js; .NET; Laravel Octane; Load Testing

Abstract–The increasing demand for web-based applications capable of handling a large number of concurrent users requires the selection of backend technologies that provide optimal performance and resource efficiency. However, previous studies generally employed different configurations and testing environments, making performance comparisons among backend technologies less objective and difficult to generalize. This study aims to analyze and compare the performance of backend REST APIs developed using Node.js, .NET, and Laravel Octane within a standardized testing environment. The research adopted a quantitative experimental approach using load testing with Grafana K6 by simulating workloads of up to 150 virtual users through HTTP GET requests. The evaluated performance metrics included throughput or requests per second (RPS), response time (latency), error rate, as well as CPU and memory utilization. The results showed that Node.js achieved the best performance with a throughput of 493.93 requests per second and an average latency of 77.43 ms, followed by .NET with a throughput of 324.94 requests per second and a latency of 163.06 ms, while Laravel Octane achieved a throughput of 192.29 requests per second with a latency of 358.62 ms. All backend technologies maintained a 0% error rate throughout the testing process. In terms of resource efficiency, Node.js also demonstrated the lowest CPU and memory utilization compared to .NET and Laravel Octane. These findings indicate that differences in execution models significantly influence throughput, response time, and resource utilization efficiency, providing valuable insights for selecting appropriate backend technologies to develop high-performance and highly scalable web applications.

Keywords: REST API; Node.js; .NET; Laravel Octane; Load Testing

1. PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah mendorong transformasi signifikan dalam pengembangan aplikasi berbasis web, khususnya pada lapisan backend yang berperan dalam memproses permintaan pengguna secara efisien dan andal. Dalam arsitektur sistem modern, Application Programming Interface (API) menjadi komponen utama yang memungkinkan integrasi dan komunikasi antar sistem yang berjalan pada berbagai platform. API berfungsi sebagai jembatan yang menghubungkan berbagai layanan dan memungkinkan pertukaran data secara terstruktur antar aplikasi (Maulana et al., 2022). Salah satu pendekatan yang paling banyak digunakan adalah Representational State Transfer (REST), yang menawarkan fleksibilitas, skalabilitas, serta kemudahan implementasi dalam sistem terdistribusi (F. Pratama & Farisi, 2025; Sutara & Gunawan, 2024; Zulkahfi et al., 2025). Popularitas REST API yang tinggi juga didorong oleh kemampuannya dalam mendukung interoperabilitas dan efisiensi komunikasi antar layanan pada arsitektur microservices (Oktafianto et al., 2025). Selain itu, REST juga dikenal sebagai arsitektur yang mampu mendukung komunikasi berbasis HTTP secara ringan dan efisien dalam berbagai sistem berbasis web (I. G. A. E. Pratama et al., 2022).

Seiring meningkatnya kebutuhan akan aplikasi dengan respons cepat dan kemampuan menangani beban tinggi, performa backend API menjadi faktor krusial yang menentukan kualitas layanan sistem. Parameter performa seperti throughput, response time (latency), error rate, serta penggunaan sumber daya seperti CPU dan memori menjadi indikator utama dalam mengevaluasi kinerja suatu sistem (Hanggara et al., 2022; Oktafianto et al., 2025). Selain itu, pemilihan bahasa pemrograman dan framework backend memiliki pengaruh signifikan terhadap performa, karena setiap teknologi memiliki karakteristik arsitektur dan mekanisme eksekusi yang berbeda (Hadinata & Stianingsih, 2024;

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

Siahaan & Wijaya, 2024). Efisiensi penggunaan sumber daya juga menjadi perhatian penting, karena aplikasi modern sangat sensitif terhadap delay yang dapat memengaruhi pengalaman pengguna (Amarulloh et al., 2023). Oleh karena itu, pemilihan teknologi yang tepat menjadi tantangan penting dalam pengembangan aplikasi modern.

Berbagai penelitian telah dilakukan untuk membandingkan performa backend API menggunakan beragam teknologi (Handayani et al., 2026). Studi yang membandingkan beberapa bahasa pemrograman seperti Go, PHP, dan JavaScript menunjukkan bahwa Node.js cenderung memiliki kinerja lebih baik dalam menangani permintaan dan memberikan waktu respons yang lebih rendah dibandingkan PHP (F. Pratama & Farisi, 2025). Hasil serupa juga ditemukan dalam penelitian yang membandingkan Express.js dengan Laravel, di mana Express.js menunjukkan keunggulan dalam waktu respons dan efisiensi penggunaan sumber daya (Hadinata & Stianingsih, 2024). Penelitian lain yang melibatkan perbandingan antara Flask, Laravel, dan Express.js juga menunjukkan bahwa Express.js memiliki performa terbaik dalam hal response time dengan tingkat kesalahan yang rendah (Purwanto, 2023). Selain itu, studi lain menunjukkan bahwa Express.js mampu memberikan response time yang lebih cepat dibandingkan framework berbasis PHP seperti CodeIgniter, meskipun dengan konsumsi resource yang berbeda (Maulana et al., 2022).

Selain perbandingan antar bahasa dan framework, beberapa penelitian juga menyoroti perbedaan performa di dalam ekosistem yang sama. Misalnya, studi yang membandingkan framework Node.js seperti Express.js dan Fastify menunjukkan bahwa Fastify memiliki throughput yang lebih tinggi dan waktu respons yang lebih rendah dibandingkan Express.js pada skenario tertentu (Sutono et al., 2025). Penelitian lain yang membandingkan Hapi.js dan NestJS juga menunjukkan adanya perbedaan signifikan dalam penggunaan CPU, memori, serta waktu respons, yang dipengaruhi oleh desain arsitektur masing-masing framework (Oktafianto et al., 2025). Selain itu, perbandingan antara berbagai arsitektur API seperti REST, GraphQL, dan gRPC menunjukkan bahwa setiap pendekatan memiliki karakteristik performa yang berbeda tergantung pada skenario penggunaan dan kompleksitas data (Chandra & Farisi, 2025). Hal ini menunjukkan bahwa performa tidak hanya ditentukan oleh bahasa pemrograman, tetapi juga oleh desain internal framework dan arsitektur API yang digunakan.

Di sisi lain, perkembangan teknologi backend juga mencakup evolusi model eksekusi yang digunakan dalam memproses permintaan (Amartharizqi et al., 2026). Dalam ekosistem PHP, pendekatan worker model seperti Laravel Octane diperkenalkan untuk mengatasi keterbatasan arsitektur tradisional berbasis request-response yang memiliki overhead tinggi. Pendekatan ini memungkinkan proses aplikasi tetap aktif di memori sehingga dapat meningkatkan throughput dan menurunkan latency (Iskandar et al., 2026). Penelitian terkait menunjukkan bahwa model event-driven non-blocking cenderung lebih efisien dalam menangani permintaan paralel dibandingkan model tradisional, terutama pada sistem dengan beban tinggi (Iskandar et al., 2026; F. Pratama & Farisi, 2025). Selain itu, penggunaan framework modern juga bertujuan untuk meningkatkan efisiensi pengembangan sekaligus menjaga performa sistem melalui struktur kode yang lebih terorganisir (Kansha et al., 2023).

Meskipun berbagai penelitian telah dilakukan, sebagian besar studi masih memiliki keterbatasan dalam hal cakupan dan konsistensi lingkungan pengujian. Banyak penelitian hanya membandingkan dua atau tiga teknologi dalam kondisi yang berbeda, baik dari sisi konfigurasi sistem, metode pengujian, maupun skenario beban kerja (Siahaan & Wijaya, 2024; Supria et al., 2024). Perbedaan tersebut menyebabkan hasil penelitian sulit dibandingkan secara langsung dan komprehensif. Selain itu, masih terbatas penelitian yang secara eksplisit mengkaji pengaruh model eksekusi terhadap performa backend API dalam satu lingkungan pengujian yang terstandarisasi. Beberapa penelitian sebelumnya juga menunjukkan bahwa keterbatasan parameter pengujian dapat memengaruhi validitas hasil analisis performa (Chandra & Farisi, 2025).

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk menganalisis dan membandingkan performa REST API yang dibangun menggunakan tiga teknologi backend, yaitu Node.js, .NET, dan Laravel dengan Laravel Octane. Pengujian dilakukan menggunakan pendekatan *load testing* dengan parameter yang meliputi *throughput*, *response time*, *error rate*, serta konsumsi sumber daya sistem. Seluruh pengujian dilakukan dalam lingkungan yang terstandarisasi dengan konfigurasi sistem yang identik guna memastikan keadilan perbandingan (*fair benchmarking*).

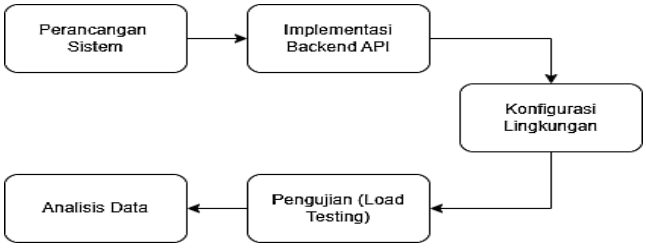
Kontribusi utama dari penelitian ini terletak pada analisis komparatif yang tidak hanya berfokus pada hasil numerik performa, tetapi juga mengkaji keterkaitan antara model eksekusi masing-masing teknologi dengan karakteristik kinerja yang dihasilkan. Dengan demikian, penelitian ini diharapkan dapat memberikan pemahaman yang lebih mendalam bagi pengembang maupun peneliti dalam menentukan teknologi backend yang optimal sesuai dengan kebutuhan performa dan efisiensi sistem.

2. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan eksperimen kuantitatif untuk menganalisis dan membandingkan performa backend REST API pada Node.js, .NET, dan Laravel Octane. Pendekatan eksperimen kuantitatif digunakan karena memungkinkan pengukuran performa sistem secara objektif berdasarkan data numerik yang diperoleh dari hasil pengujian sehingga hubungan antara teknologi backend dan karakteristik performanya dapat dianalisis secara terukur (Prilyansyah & Susanto, 2025; Yulianto et al., 2024). Metode pengujian yang diterapkan adalah load testing, yaitu teknik pengujian yang digunakan untuk mengevaluasi kemampuan sistem dalam menangani sejumlah permintaan secara bersamaan pada kondisi beban tertentu. Pengujian dilakukan menggunakan Grafana K6 dengan parameter pengujian meliputi *throughput* (*requests per second*), *response time* (*latency*), *error rate*, serta penggunaan CPU dan

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

memori sebagai indikator performa dan efisiensi sistem (Hadinata & Stianingsih, 2024; Hanggara et al., 2022). Tahapan penelitian disusun secara sistematis mulai dari perancangan sistem, implementasi backend REST API, konfigurasi lingkungan pengujian, pelaksanaan load testing, hingga analisis data hasil pengujian. Alur tahapan penelitian ditunjukkan pada Gambar 1.



Gambar 1. Tahapan Penelitian

Berdasarkan Gambar 1, tahapan penelitian ini disusun ke dalam beberapa tahapan utama yang meliputi perancangan sistem, implementasi backend API, konfigurasi lingkungan pengujian, proses pengujian performa, serta analisis data. Tahapan penelitian ini dirancang secara terstruktur untuk memastikan bahwa seluruh proses berjalan secara sistematis dan hasil yang diperoleh dapat dibandingkan secara objektif.

2.1 Perancangan Sistem

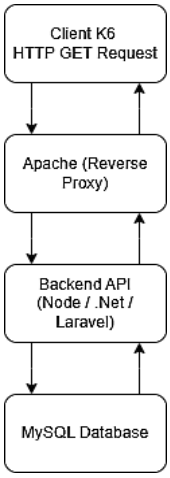
Tahap perancangan sistem dilakukan untuk menentukan arsitektur backend REST API, skenario pengujian, serta parameter evaluasi yang digunakan dalam penelitian. Pada tahap ini dirancang tiga backend REST API menggunakan Node.js, .NET, dan Laravel Octane dengan struktur endpoint, logika bisnis, serta format respons JSON yang identik agar setiap backend memproses jenis permintaan dan beban kerja yang sama selama proses pengujian. Selain itu, ditetapkan parameter pengujian yang meliputi *throughput* (*requests per second*), *response time* (*latency*), *error rate*, penggunaan CPU, dan penggunaan memori sebagai indikator dalam mengevaluasi performa backend REST API (Hadinata & Stianingsih, 2024; Hanggara et al., 2022).

2.2 Implementasi Backend API

Pada tahap implementasi dilakukan dengan membangun backend REST API menggunakan Node.js, .NET, dan Laravel Octane berdasarkan rancangan sistem yang telah ditetapkan. Ketiga backend dikembangkan menggunakan logika aplikasi, struktur endpoint, dan mekanisme akses basis data yang sama sehingga perbedaan performa yang dihasilkan berasal dari karakteristik teknologi dan model eksekusi masing-masing backend. Seluruh backend dihubungkan dengan basis data MySQL yang memiliki struktur tabel dan jumlah data yang identik sebanyak 350 data untuk menjaga konsistensi beban pengujian dan mendukung proses *fair benchmarking* (Maulana et al., 2022; Yulianto et al., 2024)

2.3 Konfigurasi Lingkungan

Tahap konfigurasi lingkungan pengujian dilakukan untuk memastikan seluruh backend REST API diuji pada kondisi perangkat keras dan perangkat lunak yang sama sehingga hasil pengujian dapat dibandingkan secara objektif. Seluruh backend dijalankan pada komputer dengan prosesor Intel Core i5-6200U, RAM 12 GB, dan sistem operasi Windows. Apache digunakan sebagai *reverse proxy* untuk meneruskan permintaan menuju backend REST API, MySQL digunakan sebagai basis data, sedangkan Grafana K6 digunakan sebagai alat *load testing* untuk menghasilkan beban sesuai skenario pengujian. Konfigurasi lingkungan pengujian yang seragam bertujuan meminimalkan pengaruh faktor eksternal terhadap hasil pengukuran (Hadinata & Stianingsih, 2024)



Gambar 2. Alur Sistem

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

Gambar 2 menunjukkan arsitektur lingkungan pengujian yang digunakan pada penelitian. Permintaan HTTP GET dikirimkan oleh Grafana K6 sebagai client menuju Apache yang berperan sebagai reverse proxy. Selanjutnya, Apache meneruskan permintaan ke backend REST API sesuai teknologi yang sedang diuji, yaitu Node.js, .NET, atau Laravel Octane. Backend kemudian mengakses basis data MySQL untuk memproses permintaan dan mengembalikan respons kepada client. Arsitektur ini memastikan seluruh backend diuji menggunakan alur komunikasi yang sama sehingga hasil pengujian dapat dibandingkan secara objektif.

2.4 Pengujian (Load testing)

Tahap pengujian *load testing* dilakukan menggunakan Grafana K6 untuk mengevaluasi performa backend REST API pada Node.js, .NET, dan Laravel Octane. Pengujian dilakukan melalui tiga fase, yaitu *ramp-up*, *steady state*, dan *ramp-down*, dengan jumlah beban meningkat secara bertahap hingga mencapai 150 *virtual users*. Setiap *virtual user* mengirimkan permintaan HTTP GET secara berulang ke endpoint REST API untuk mensimulasikan akses pengguna terhadap layanan backend. Parameter yang diukur meliputi *throughput* (*requests per second*), *response time* (*latency*), *error rate*, penggunaan CPU, dan penggunaan memori sebagai dasar analisis performa masing-masing backend (Hanggara et al., 2022; F. Pratama & Farisi, 2025).

```
import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
    stages: [
        { duration: '10s', target: 50 },   // ramp-up
        { duration: '1m', target: 150 },  // steady load
        { duration: '10s', target: 0 },    // ramp-down
    ],
};

export default function () {
    const BASE_URL = __ENV.BASE_URL || 'http://localhost:3000';
    const res = http.get(`${BASE_URL}/api/products`);

    check(res, {
        'status is 200': (r) => r.status === 200,
    });

    sleep(0.1);
}
```

Gambar 3. Skrip Pengujian Load Testing K6

Gambar 3. Merupakan skrip pengujian load testing K6, Skrip tersebut menunjukkan bahwa setiap *virtual user* mengirimkan permintaan HTTP GET secara berulang ke endpoint API dengan interval waktu tertentu. Validasi dilakukan terhadap status respons untuk memastikan bahwa sistem memberikan respons yang sesuai selama proses pengujian. Pendekatan ini digunakan untuk mensimulasikan pola akses pengguna secara realistis serta mengukur performa sistem dalam kondisi beban yang dinamis. Dengan menggunakan skenario pengujian yang terstruktur dan lingkungan yang terkontrol, hasil pengujian diharapkan dapat memberikan gambaran yang objektif mengenai performa masing-masing teknologi backend dalam menangani permintaan secara bersamaan.

2.5 Analisis Data

Tahap analisis data dilakukan setelah seluruh proses *load testing* selesai dilaksanakan. Data hasil pengujian dianalisis menggunakan analisis deskriptif dengan membandingkan nilai *throughput*, *response time* (*latency*), *error rate*, penggunaan CPU, dan penggunaan memori dari masing-masing backend REST API. Hasil perbandingan tersebut digunakan untuk mengidentifikasi pengaruh model eksekusi terhadap karakteristik performa Node.js, .NET, dan Laravel Octane sehingga dapat memberikan rekomendasi pemilihan teknologi backend yang sesuai dengan kebutuhan performa dan skalabilitas sistem (Oktafianto et al., 2025; Yulianto et al., 2024)

3. HASIL DAN PEMBAHASAN

3.1 Hasil Pengujian Performa

Berdasarkan tahapan penelitian yang telah dijelaskan pada Bab 2, proses evaluasi performa backend REST API dilakukan menggunakan metode *load testing* dengan memanfaatkan Grafana K6 sebagai alat pengujian. Pengujian diterapkan pada tiga backend yang telah diimplementasikan menggunakan Node.js, .NET, dan Laravel Octane dengan struktur endpoint, logika aplikasi, serta basis data yang identik. Seluruh backend dijalankan pada lingkungan pengujian

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

yang sama sehingga perbedaan hasil yang diperoleh dipengaruhi oleh karakteristik teknologi dan model eksekusi masing-masing backend, bukan oleh perbedaan konfigurasi sistem. Proses *load testing* dilakukan melalui tiga tahapan, yaitu *ramp-up*, *steady state*, dan *ramp-down*, dengan jumlah beban meningkat secara bertahap hingga mencapai 150 *virtual users*. Pada setiap tahapan tersebut, Grafana K6 mengirimkan permintaan HTTP GET secara berulang ke endpoint REST API untuk menghasilkan data performa berupa *throughput*, *response time* (*latency*), *error rate*, penggunaan CPU, dan penggunaan memori. Data yang dihasilkan kemudian digunakan sebagai dasar dalam melakukan analisis komparatif terhadap ketiga backend. Ringkasan hasil pengujian performa yang diperoleh dari proses load testing menggunakan Grafana K6 ditunjukkan pada Tabel 1.

Tabel 1. Hasil Benchmark K6

Backend	Throughput (RPS)	Avg Latency (ms)	P90 (ms)	P95 (ms)	Error Rate
Node.js	493.93	77.43	7.95	10.9	0%
.NET	324.94	163.06	87.21	104.37	0%
Laravel Octane	192.29	358.62	247.45	430.32	0%

Data pada Tabel 1 menunjukkan hasil pengukuran yang diperoleh setelah seluruh skenario *load testing* selesai dijalankan. Parameter *throughput* menunjukkan jumlah permintaan yang berhasil diproses setiap detik, sedangkan *latency* menggambarkan rata-rata waktu yang dibutuhkan backend dalam memberikan respons terhadap permintaan pengguna. Nilai P90 dan P95 digunakan untuk mengevaluasi konsistensi waktu respons pada sebagian besar permintaan, sedangkan *error rate* menunjukkan tingkat keberhasilan backend dalam menangani seluruh permintaan yang dikirimkan selama proses pengujian. Seluruh parameter tersebut dianalisis untuk mengetahui pengaruh perbedaan model eksekusi terhadap performa masing-masing backend REST API.

Hasil pengujian menunjukkan adanya perbedaan performa yang cukup signifikan antara Node.js, .NET, dan Laravel Octane. Node.js memperoleh nilai *throughput* sebesar 493,93 RPS, sedangkan .NET memperoleh 324,94 RPS dan Laravel Octane memperoleh 192,29 RPS. Nilai tersebut menunjukkan bahwa Node.js mampu memproses permintaan dalam jumlah yang lebih besar pada interval waktu yang sama dibandingkan dua backend lainnya. Semakin tinggi nilai *throughput*, semakin besar pula kemampuan backend dalam melayani permintaan pengguna secara bersamaan. Berdasarkan hasil tersebut, Node.js memiliki kapasitas pemrosesan permintaan yang paling baik pada skenario pengujian yang diterapkan.

Selain *throughput*, perbedaan performa juga terlihat pada nilai *response time* (*latency*). Node.js mencatat rata-rata *latency* sebesar 77,43 ms, sedangkan .NET memperoleh 163,06 ms, dan Laravel Octane mencapai 358,62 ms. Nilai ini menunjukkan bahwa Node.js mampu memberikan respons kepada pengguna dalam waktu yang lebih singkat dibandingkan kedua backend lainnya. Sebaliknya, peningkatan *latency* pada .NET dan Laravel Octane menunjukkan bahwa backend membutuhkan waktu yang lebih lama untuk menyelesaikan setiap permintaan ketika jumlah pengguna meningkat. Kondisi tersebut menunjukkan adanya perbedaan efisiensi dalam memproses permintaan selama proses *load testing* berlangsung.

Analisis terhadap nilai persentil P90 dan P95 memberikan gambaran mengenai kestabilan waktu respons selama pengujian. Node.js memperoleh nilai P90 sebesar 7,95 ms dan P95 sebesar 10,9 ms, sedangkan .NET memperoleh nilai P90 sebesar 87,21 ms dan P95 sebesar 104,37 ms. Laravel Octane menunjukkan nilai P90 sebesar 247,45 ms dan P95 sebesar 430,32 ms. Nilai persentil yang lebih rendah menunjukkan bahwa sebagian besar permintaan dapat diproses dengan waktu respons yang relatif konsisten. Sebaliknya, nilai persentil yang semakin tinggi menunjukkan adanya variasi waktu respons yang lebih besar ketika backend menangani peningkatan jumlah permintaan. Berdasarkan hasil tersebut, Node.js menunjukkan tingkat kestabilan performa yang lebih baik dibandingkan .NET maupun Laravel Octane.

Seluruh backend yang diuji menghasilkan *error rate* sebesar 0%, yang menunjukkan bahwa seluruh permintaan HTTP GET berhasil diproses tanpa mengalami kegagalan selama proses pengujian berlangsung. Hasil tersebut mengindikasikan bahwa ketiga backend memiliki tingkat keandalan yang baik dalam menangani beban hingga 150 *virtual users*. Dengan demikian, perbedaan performa yang diperoleh pada penelitian ini tidak dipengaruhi oleh adanya kesalahan layanan, melainkan berasal dari kemampuan masing-masing backend dalam memproses permintaan secara efisien.

Untuk memperjelas hasil pengujian, Gambar 4 hingga Gambar 6 menampilkan keluaran (*output*) Grafana K6 pada masing-masing backend. Tampilan tersebut memberikan informasi mengenai distribusi waktu respons, jumlah permintaan yang berhasil diproses, serta kondisi sistem selama *load testing*. Data tersebut selanjutnya digunakan sebagai dasar dalam menganalisis karakteristik performa masing-masing backend berdasarkan model eksekusi yang digunakan.

Gambar 4 hasil pengujian pada Node.js menunjukkan performa yang stabil selama seluruh tahapan *load testing*. Nilai *throughput* yang tinggi disertai *latency* yang rendah menunjukkan bahwa backend mampu mempertahankan kecepatan pemrosesan permintaan meskipun jumlah pengguna meningkat secara bertahap. Distribusi waktu respons yang relatif konsisten juga menunjukkan bahwa Node.js mampu menjaga kestabilan layanan tanpa mengalami penurunan performa yang berarti selama proses pengujian berlangsung.

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

Gambar 4. Hasil Benchmark K6 Node.JS

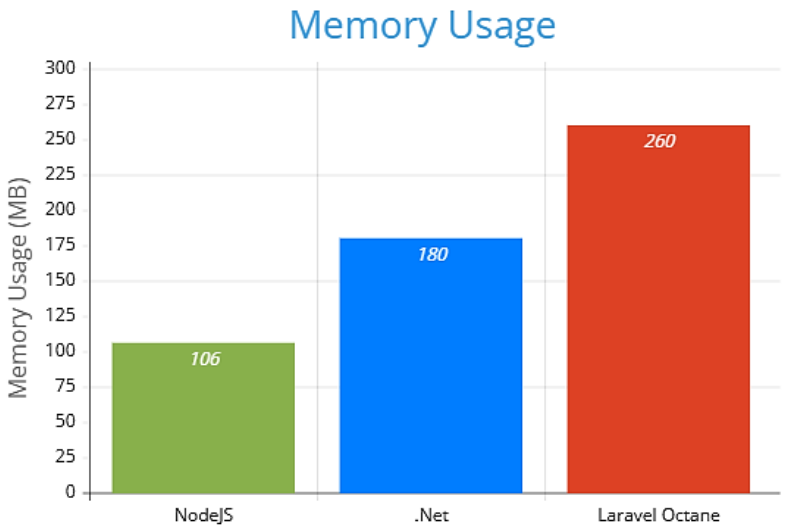
Gambar 5. Hasil Benchmark K6 .Net

Gambar 5 pengujian pada .NET menunjukkan bahwa backend tetap mampu memproses seluruh permintaan tanpa menghasilkan kesalahan. Namun demikian, waktu respons yang diperoleh lebih tinggi dibandingkan Node.js sehingga jumlah permintaan yang dapat diproses setiap detik menjadi lebih rendah. Hasil tersebut menunjukkan bahwa backend masih mampu mempertahankan stabilitas layanan, tetapi memerlukan waktu pemrosesan yang lebih panjang ketika menerima beban yang sama.

Gambar 6 hasil pengujian pada Laravel Octane menunjukkan bahwa seluruh permintaan tetap berhasil diproses tanpa menghasilkan *error rate*. Akan tetapi, nilai *throughput* yang lebih rendah serta *latency* yang lebih tinggi menunjukkan bahwa backend memerlukan waktu yang lebih lama dalam menyelesaikan setiap permintaan. Variasi waktu respons yang lebih besar juga menunjukkan bahwa performa backend cenderung menurun ketika jumlah permintaan meningkat. Hasil tersebut memperlihatkan bahwa Laravel Octane masih mampu memberikan layanan secara stabil, namun efisiensi pemrosesan permintaan belum sebaik Node.js maupun .NET. Selanjutnya, untuk memperoleh gambaran yang lebih lengkap mengenai efisiensi masing-masing backend, penelitian ini juga mengevaluasi penggunaan sumber daya sistem yang meliputi penggunaan memori dan CPU selama proses *load testing*.

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

Gambar 6. Hasil Benchmark K6 Laravel Octane

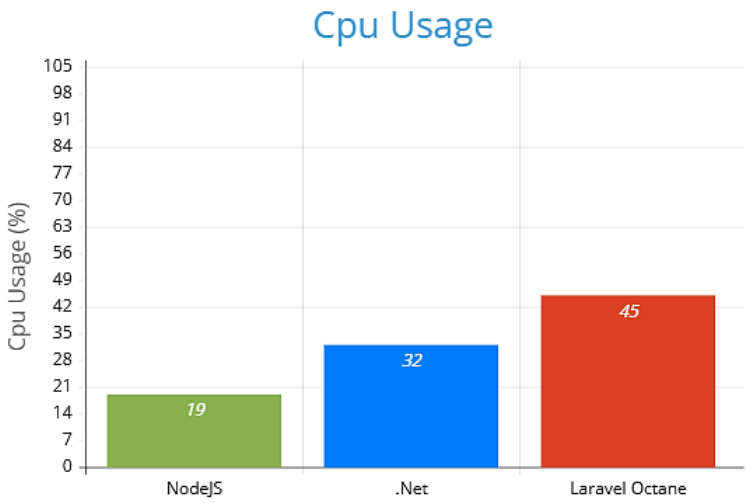


Gambar 7. Memory Usage

Berdasarkan Gambar 7, terdapat perbedaan penggunaan memori yang cukup jelas antar backend. Node.js menggunakan memori paling rendah sebesar 106 MB, sedangkan .NET menggunakan 180 MB, dan Laravel Octane menggunakan 260 MB. Jika dibandingkan, penggunaan memori .NET sekitar 69,8% lebih tinggi daripada Node.js, sementara Laravel Octane menggunakan memori sekitar 145,3% lebih tinggi daripada Node.js. Perbedaan ini menunjukkan bahwa Node.js memiliki jejak memori (*memory footprint*) yang lebih kecil dalam menangani beban pengujian yang sama. Dengan kata lain, Node.js mampu menghasilkan *throughput* tertinggi tanpa membutuhkan alokasi memori yang besar.

Penggunaan memori yang lebih tinggi pada .NET dapat dikaitkan dengan karakteristik *managed runtime* dan mekanisme internal seperti pengelolaan objek, *garbage collection*, serta dukungan eksekusi berbasis thread. Meskipun .NET tetap menunjukkan performa yang cukup baik, konsumsi memori yang lebih besar menunjukkan bahwa sistem membutuhkan ruang eksekusi yang lebih luas untuk mempertahankan stabilitas pemrosesan. Sementara itu, Laravel Octane menunjukkan penggunaan memori tertinggi. Hal ini mengindikasikan adanya *overhead* tambahan dari proses worker yang dipertahankan di memori serta kompleksitas internal framework Laravel. Kondisi tersebut menunjukkan bahwa pendekatan *persistent worker* memang dapat mengurangi proses inisialisasi berulang, tetapi tetap membutuhkan konsumsi memori yang lebih besar.

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838



Gambar 8. CPU Usage

Pada Gambar 8, pola yang serupa juga terlihat pada penggunaan CPU. Node.js menggunakan CPU paling rendah sebesar 19%, sedangkan .NET menggunakan 32%, dan Laravel Octane mencapai 45%. Dibandingkan dengan Node.js, penggunaan CPU .NET sekitar 68,4% lebih tinggi, sedangkan Laravel Octane sekitar 136,8% lebih tinggi. Perbedaan ini menunjukkan bahwa Node.js memiliki efisiensi komputasi yang lebih baik karena mampu menangani jumlah permintaan yang lebih tinggi dengan penggunaan CPU yang lebih rendah.

Konsumsi CPU yang lebih tinggi pada .NET menunjukkan adanya beban tambahan dalam pengelolaan proses paralel, terutama karena pendekatan *multithreaded asynchronous* memerlukan mekanisme pengaturan thread dan eksekusi internal yang lebih kompleks. Pada Laravel Octane, penggunaan CPU tertinggi menunjukkan bahwa sistem membutuhkan lebih banyak siklus pemrosesan untuk menangani beban yang sama. Hal ini dapat menjadi indikasi adanya *overhead* pada pengelolaan worker, framework, serta proses komunikasi dengan aplikasi. Jika dikaitkan dengan hasil *throughput* dan *latency*, Laravel Octane memiliki konsumsi CPU paling tinggi tetapi menghasilkan *throughput* paling rendah dan *latency* paling besar, sehingga efisiensi komputasinya lebih rendah dibandingkan Node.js dan .NET.

Secara keseluruhan, hasil pada Gambar 4 dan Gambar 5 menunjukkan bahwa efisiensi sumber daya tidak selalu sejalan dengan besarnya konsumsi CPU dan memori. Node.js justru menghasilkan performa terbaik dengan penggunaan sumber daya paling rendah, sedangkan Laravel Octane menggunakan sumber daya paling besar tetapi menghasilkan performa paling rendah. Temuan ini memperkuat bahwa evaluasi backend API tidak cukup hanya melihat *throughput* dan *latency*, tetapi juga perlu mempertimbangkan efisiensi penggunaan CPU dan memori sebagai indikator penting dalam menentukan teknologi yang optimal untuk sistem berskala tinggi.

3.2 Analisis Perbandingan Performa

Berdasarkan hasil pengujian pada Tabel 1, perbedaan performa antara Node.js, .NET, dan Laravel Octane menunjukkan pola yang konsisten pada seluruh metrik utama. Node.js menunjukkan keunggulan yang signifikan dengan nilai *throughput* sekitar 493.93 RPS, yang lebih tinggi dibandingkan .NET sebesar 324.94 RPS dan Laravel Octane sebesar 192.29 RPS. Perbedaan ini mengindikasikan bahwa Node.js memiliki kapasitas pemrosesan permintaan yang lebih besar dalam kondisi beban yang sama, sehingga lebih mampu menangani jumlah pengguna yang tinggi secara simultan.

Dari sisi waktu respons, Node.js juga menunjukkan performa yang lebih baik dengan rata-rata *latency* sebesar 77.43 ms, yang jauh lebih rendah dibandingkan .NET sebesar 163.06 ms dan Laravel Octane sebesar 358.62 ms. Perbedaan ini menunjukkan bahwa Node.js tidak hanya mampu memproses lebih banyak permintaan, tetapi juga memberikan respons yang lebih cepat kepada pengguna. Sebaliknya, peningkatan *latency* yang signifikan pada .NET dan Laravel Octane menunjukkan adanya keterbatasan dalam menangani beban tinggi secara efisien.

Analisis terhadap nilai persentil P95 memberikan gambaran yang lebih mendalam terkait stabilitas performa. Node.js memiliki nilai P95 sebesar 10.9 ms, yang menunjukkan bahwa sebagian besar permintaan dapat diproses dengan waktu respons yang konsisten tanpa adanya lonjakan signifikan. Sebaliknya, nilai P95 pada .NET mencapai 104.37 ms dan Laravel Octane sebesar 430.32 ms, yang mengindikasikan adanya variasi waktu respons yang lebih besar. Variasi ini berpotensi menyebabkan ketidakstabilan performa, terutama pada kondisi beban tinggi.

Jika dikaitkan dengan penggunaan sumber daya, terlihat bahwa performa tinggi pada Node.js tidak diikuti dengan peningkatan konsumsi CPU dan memori yang signifikan. Hal ini menunjukkan bahwa Node.js tidak hanya unggul dari sisi performa mentah (*raw performance*), tetapi juga dari sisi efisiensi sistem. Sebaliknya, Laravel Octane menunjukkan konsumsi sumber daya yang lebih tinggi dengan performa yang relatif lebih rendah, yang mengindikasikan adanya *overhead* dalam proses eksekusi. .NET berada di antara keduanya dengan performa yang cukup stabil namun masih memerlukan penggunaan resource yang lebih besar dibandingkan Node.js.

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

Dengan demikian, perbedaan performa yang dihasilkan tidak hanya dipengaruhi oleh kemampuan memproses permintaan dalam jumlah besar, tetapi juga oleh efisiensi dalam pemanfaatan sumber daya sistem. Teknologi yang mampu menjaga keseimbangan antara *throughput*, *latency*, dan penggunaan resource cenderung memiliki performa yang lebih optimal dalam kondisi beban tinggi.

Selain peningkatan *throughput*, penurunan *latency* yang signifikan pada Node.js menunjukkan bahwa sistem tidak mengalami degradasi performa meskipun menangani jumlah permintaan yang lebih besar. Hal ini menunjukkan bahwa efisiensi sistem tidak hanya ditentukan oleh kemampuan memproses permintaan dalam jumlah besar, tetapi juga oleh kemampuan menjaga stabilitas waktu respons. Dengan demikian, Node.js menunjukkan karakteristik performa yang tidak hanya tinggi, tetapi juga konsisten pada berbagai kondisi beban.

3.3 Pembahasan Model Eksekusi

Perbedaan performa yang signifikan antara Node.js, .NET, dan Laravel Octane dalam penelitian ini dapat dijelaskan melalui model eksekusi yang digunakan oleh masing-masing teknologi backend. Model eksekusi merupakan faktor fundamental yang menentukan bagaimana suatu sistem menangani permintaan secara paralel (*concurrency*), serta bagaimana sumber daya sistem dimanfaatkan selama proses eksekusi berlangsung.

Node.js menggunakan model *event-driven non-blocking*, di mana satu thread utama mampu menangani banyak permintaan secara simultan tanpa harus menunggu proses I/O selesai. Pendekatan ini memungkinkan sistem untuk meminimalkan waktu tunggu (*idle time*) dan meningkatkan efisiensi dalam pemrosesan permintaan. Hal ini tercermin dari nilai *throughput* yang tinggi serta *latency* yang rendah pada hasil pengujian.

Sebaliknya, .NET menggunakan pendekatan *multithreaded asynchronous*, yang memanfaatkan banyak thread untuk menangani permintaan secara paralel. Meskipun pendekatan ini cukup efektif dalam meningkatkan concurrency, pengelolaan thread memerlukan overhead tambahan seperti *context switching* dan alokasi resource untuk setiap thread. Hal ini dapat menyebabkan peningkatan waktu respons serta konsumsi CPU yang lebih tinggi dibandingkan pendekatan non-blocking.

Laravel Octane menggunakan pendekatan *worker-based process*, di mana proses aplikasi dipertahankan dalam memori untuk mengurangi overhead inisialisasi pada setiap permintaan. Meskipun pendekatan ini memberikan peningkatan performa dibandingkan PHP tradisional, hasil pengujian menunjukkan bahwa pendekatan ini masih memiliki keterbatasan dalam efisiensi, terutama dalam penggunaan CPU dan memori. Hal ini kemungkinan disebabkan oleh kompleksitas pengelolaan worker serta overhead framework yang lebih besar.

Temuan ini menunjukkan bahwa model eksekusi memiliki pengaruh yang sangat signifikan terhadap performa backend API. Pendekatan non-blocking yang digunakan oleh Node.js terbukti lebih efisien dalam menangani beban tinggi dibandingkan pendekatan berbasis thread maupun proses. Dengan demikian, pemilihan model eksekusi menjadi faktor penting dalam menentukan performa sistem, terutama pada aplikasi dengan kebutuhan concurrency yang tinggi.

3.4 Analisis Skalabilitas Sistem

Selain performa pada kondisi beban tertentu, penting untuk menganalisis bagaimana masing-masing teknologi backend merespons peningkatan jumlah pengguna secara bertahap. Skalabilitas sistem menjadi faktor krusial dalam menentukan kemampuan suatu backend API untuk mempertahankan performa ketika terjadi lonjakan trafik.

Berdasarkan pola hasil pengujian, Node.js menunjukkan karakteristik skalabilitas yang lebih baik dibandingkan .NET dan Laravel Octane. Hal ini terlihat dari kemampuannya dalam mempertahankan nilai latency yang relatif stabil meskipun jumlah pengguna meningkat hingga mencapai beban puncak. Peningkatan jumlah request tidak menyebabkan lonjakan waktu respons yang signifikan, yang menunjukkan bahwa sistem mampu mengelola concurrency secara efisien.

Sebaliknya, .NET menunjukkan peningkatan latency yang cukup signifikan seiring dengan bertambahnya jumlah pengguna. Hal ini mengindikasikan adanya batas dalam efisiensi pengelolaan thread, di mana semakin banyak permintaan yang diproses secara paralel akan meningkatkan overhead sistem. Kondisi ini dapat menyebabkan penurunan performa ketika sistem dihadapkan pada beban yang lebih besar.

Laravel Octane menunjukkan kecenderungan penurunan performa yang lebih jelas pada kondisi beban tinggi. Meskipun menggunakan pendekatan worker-based untuk meningkatkan efisiensi, hasil pengujian menunjukkan bahwa sistem masih mengalami keterbatasan dalam menangani concurrency secara optimal. Hal ini dapat disebabkan oleh overhead dalam pengelolaan worker serta konsumsi resource yang tinggi.

Dengan demikian, dapat disimpulkan bahwa kemampuan skalabilitas sangat dipengaruhi oleh model eksekusi yang digunakan. Teknologi dengan pendekatan non-blocking seperti Node.js cenderung memiliki skalabilitas yang lebih baik dibandingkan pendekatan berbasis thread maupun proses.

3.5 Implikasi Hasil Penelitian

Hasil penelitian ini memberikan implikasi penting dalam pemilihan teknologi backend untuk pengembangan aplikasi modern. Dalam konteks sistem dengan kebutuhan performa tinggi dan jumlah pengguna yang besar, Node.js dapat menjadi pilihan yang optimal karena mampu memberikan *throughput* tinggi, waktu respons rendah, serta efisiensi penggunaan sumber daya yang baik.


TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

.NET dapat menjadi alternatif yang tepat untuk sistem yang membutuhkan stabilitas dan integrasi dengan ekosistem tertentu, meskipun terdapat kompromi pada aspek performa dibandingkan Node.js. Pendekatan *multithreaded* yang digunakan memungkinkan sistem tetap berjalan stabil, namun dengan biaya resource yang lebih tinggi.

Sementara itu, Laravel Octane lebih sesuai digunakan pada sistem yang mengutamakan kemudahan pengembangan dan fleksibilitas framework. Meskipun performanya lebih rendah dibandingkan Node.js dan .NET, teknologi ini tetap relevan dalam konteks pengembangan aplikasi yang tidak memiliki kebutuhan concurrency yang tinggi.

Selain itu, hasil penelitian ini juga menunjukkan bahwa performa backend tidak hanya ditentukan oleh bahasa pemrograman atau framework yang digunakan, tetapi juga oleh model eksekusi yang mendasarinya. Oleh karena itu, pengembang perlu mempertimbangkan karakteristik model eksekusi dalam memilih teknologi backend, terutama pada sistem yang membutuhkan skalabilitas dan efisiensi tinggi.

Dalam konteks yang lebih luas, penelitian ini juga memberikan kontribusi terhadap pemahaman mengenai hubungan antara arsitektur sistem dan performa, khususnya dalam implementasi REST API. Dengan menggunakan lingkungan pengujian yang terstandarisasi, hasil yang diperoleh dapat menjadi referensi bagi penelitian selanjutnya dalam melakukan evaluasi performa backend secara lebih komprehensif.

Selain itu, hasil penelitian ini juga memberikan implikasi praktis dalam konteks pengembangan sistem skala besar, seperti aplikasi berbasis microservices dan sistem real-time. Pada sistem tersebut, kemampuan dalam menangani permintaan secara simultan dengan efisien menjadi faktor utama dalam menjaga performa layanan. Oleh karena itu, pemilihan teknologi backend yang tepat dapat memberikan dampak signifikan terhadap biaya operasional sistem, terutama dalam penggunaan resource seperti CPU dan memori pada lingkungan produksi.

4. KESIMPULAN

Penelitian ini berhasil menganalisis dan membandingkan performa backend REST API pada Node.js, .NET, dan Laravel Octane menggunakan metode *load testing* dalam lingkungan pengujian yang terstandarisasi. Berdasarkan hasil pengujian yang telah dilakukan, diketahui bahwa masing-masing backend memiliki karakteristik performa yang berbeda dalam menangani permintaan pengguna secara bersamaan. Node.js menunjukkan performa paling baik dengan kemampuan memproses permintaan secara lebih efisien dan konsisten dibandingkan .NET maupun Laravel Octane. Sementara itu, .NET menunjukkan performa yang cukup stabil dengan kemampuan pemrosesan yang berada pada tingkat menengah, sedangkan Laravel Octane masih memerlukan sumber daya yang lebih besar untuk menghasilkan performa yang setara. Hasil penelitian juga menunjukkan bahwa penerapan metode *load testing* mampu memberikan gambaran yang objektif mengenai karakteristik performa setiap backend karena seluruh pengujian dilakukan menggunakan skenario, parameter, dan lingkungan yang sama. Dengan pendekatan tersebut, perbedaan performa yang diperoleh dapat dikaitkan dengan karakteristik teknologi dan model eksekusi masing-masing backend sehingga proses perbandingan dapat dilakukan secara lebih adil (*fair benchmarking*). Kontribusi penelitian ini terletak pada penyediaan hasil analisis komparatif yang dapat menjadi referensi dalam pemilihan teknologi backend REST API sesuai kebutuhan performa dan skalabilitas aplikasi. Selain memberikan informasi mengenai kemampuan pemrosesan permintaan, penelitian ini juga memperlihatkan hubungan antara performa backend dengan efisiensi penggunaan sumber daya sistem sehingga dapat menjadi bahan pertimbangan dalam proses pengembangan aplikasi berbasis web. Penelitian ini sendiri masih memiliki keterbatasan karena pengujian hanya dilakukan menggunakan metode HTTP GET dengan skenario *read-heavy workload*, jumlah beban maksimum 150 *virtual users*, serta lingkungan pengujian lokal. Oleh karena itu, penelitian selanjutnya disarankan untuk mengembangkan skenario pengujian dengan menambahkan operasi HTTP lainnya seperti POST, PUT, dan DELETE, menggunakan jumlah pengguna virtual yang lebih besar, serta menerapkan pengujian pada lingkungan produksi atau sistem terdistribusi sehingga hasil yang diperoleh dapat menggambarkan kondisi implementasi yang lebih nyata.

REFERENCES

Amartharizqi, M. R., Akbar, F. A., & Aditiawan, F. P. (2026). Perbandingan performa rest api laravel pada web server nginx dan frankenphp dengan load testing (studi kasus: aplikasi pembelajaran xyz). *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *11*(2), 1259–1271. https://doi.org/https://doi.org/10.29100/jipi.v11i2.8156

Amarulloh, A., Kurniasih, & Muchlis. (2023). Analisis Perbandingan Performa Web Service REST Menggunakan Framework Laravel, Django, dan Node.JS pada Aplikasi Berbasis Website. *Jurnal Teknik Informatika Stmik Antar Bangsa*, *09*(01), 12–17.

Chandra, S., & Farisi, A. (2025). Comparative Analysis of RESTfull , GraphQL , and gRPC APIs : Perfomance Insight from Load and Stress Testing. *Jurnal SISFOKOM (Sistem Informasi Dan Komputer)*, *14*(01), 81–85. https://doi.org/https://doi.org/10.32736/sisfokom.v14i1.2315

Hadinata, W., & Stianingsih, L. (2024). Analisis Perbandingan Performa RESTfull API antara Express.JS dengan Laravel Framework dengan JMeter. *JITET (Jurnal Informatika Dan Teknik Elektro Terapan)*, *12*(1), 531–540. https://doi.org/http://dx.doi.org/10.23960/jitet.v12i1.3845

Handayani, D., Ibrahim, S. R. M., Nanang, Valentina, P. E., & Putra, F. M. K. (2026). Komparasi Performa REST API Laravel 11 dan CodeIgniter 4 Menggunakan Metode Eksperimental. *Jurnal Fasilkom*, *16*(1), 167–174.

TIN: Terapan Informatika Nusantara
Vol 7, No 1, June 2026, page 610-620
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin
DOI 10.47065/tin.v7i1.9838

Hanggara, B. T., Nasrullah, M. H., & Pramono, D. (2022). Analisis Perbandingan Performa Framework NestJS dan Lumen pada Studi Kasus Aplikasi Berbasis REST API. *J-INTECH (Journal of Information and Technology)*, *204*, 181–189.

Iskandar, A., Sarif, M. I., Hafiz, M. F., Harahap, D. R., & Purba, S. S. T. (2026). Analisis Komparatif Kinerja Laravel Octane dan Webman pada Layanan API Berbasis Worker Model PHP. *JUKTISI (Jurnal Komputer Teknologi Informasi Sistem Komputer)*, *4*(3), 1583–1590. https://doi.org/https://doi.org/10.62712/juktisi.v4i3.717

Kansha, W. M., Saherih, & Muchlis. (2023). Analisis Perbandingan Struktur dan Performa Framework Codeigniter dan Laravel dalam Pengembangan Web Application. *Jurnal Teknik Informatika STMIK Antar Bangsa*, *09*(01), 25–31.

Maulana, L., Prihandani, K., & Rizal, A. (2022). Analisis Perbandingan Kinerja Framework Codeigniter dengan Express.Js pada Server RESTfull API. *Jurnal Ilmiah Wahana Pendidikan*, *8*(September), 316–326. https://doi.org/https://doi.org/10.5281/zenodo.7067707

Oktafianto, M. A., Hanggara, B. T., & Akbar, M. A. (2025). Analisis Perbandingan Performa Framework Web Server Nest JS dan Hapi JS Berbasis REST API. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *9*(2), 1–10.

Pratama, F., & Farisi, A. (2025). Analisis Perbandingan Kinerja Backend API Menggunakan PHP, Golang, dan JavaScript. *Techno.COM*, *24*(1), 153–165.

Pratama, I. G. A. E., Satwika, I. P., & Wijaya, I. N. Y. A. (2022). Analisis Perbandingan Performa API Metode REST dan GraphQL dengan PHP dan GO. *Jurnal Teknologi Informasi Dan Komputer*, *08*(04), 344–353.

Prilyansyah, Y., & Susanto. (2025). Optimasi Performa Api Aplikasi E-Commerce. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, *10*(2).

Purwanto, T. (2023). Analisa Perbandingan Kinerja REST API dengan Framework Flask, Laravel, dan Express Js. *Scientia Sacra: Jurnal Sains, Teknologi Dan Masyarakat*, *3*(4), 49–55.

Siahaan, M., & Wijaya, R. (2024). Performance Comparison between Laravel and ExpressJs Framework Using Apache JMeter. *JITE (Journal of Informatics and Telecommunication Engineering)*, *7*(January), 545–554. https://doi.org/https://doi.org/10.31289/jite.v7i2.10571

Supria, Faizi, M. N., Yulia, I., Afridon, M., Arizka, A., Sarudin, & Sulistyo, J. N. (2024). Perbandingan Performa Framework Laravel, Flask API Python, dan PHP Native untuk Aplikasi API pada Data AIS Polbeng. *Seminar Nasional Industri Dan Teknologi (SNIT), Politeknik Negeri Bengkalis Perbandingan*, *November*, 17–23.

Sutara, B., & Gunawan, S. S. (2024). Comparative Analysis of Rest API Performance between Express.JS Frameowrk and Hapi.JS Using Apache JMeter. *Jurnal Riset Teknik Informatika (JURETI)*, *1*(1), 19–25.

Sutono, Arrahman, A., Musrifah, A., & Jailani, L. (2025). Analisis Perbandingan Performa Framework Node.JS antara Express.JS dan Fastify Berbasis REST API. *Infotech Journal*, *11*(2), 364–370. https://doi.org/https://doi.org/10.31949/infotech.v11i2.15906

Yulianto, R., Mardiana, P, R. A., & Nama, G. F. (2024). Performance Comparison Aanalysis of Springboot and Laravel Frameowrks Using API Web Service. *JITET (Jurnal Informatika Dan Teknik Elektro Terapan)*, *12*(2), 1145–1153. https://doi.org/http://dx.doi.org/10.23960/jitet.v12i2.4141

Zulkahfi, M. R., Afirianto, T., & Akbar, M. A. (2025). Analisis Performa Load Time Halaman Statis antara Pendekatan Inertia dan REST API pada Laravel. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *9*(6), 1–10.