JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 2, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i2.9264

ANALISIS PERBANDINGAN PERFORMA DAN EFEKTIVITAS FRAMEWORK LARAVEL DAN NODE.JS DALAM PENGEMBANGAN WEB

Eldoni Mosul[1*], Mohamad Jajuli[2], Iqbal Maulana[3]

[1,2,3]Informatika, Universitas Singaperbangsa Karawang; Jl. HS.Ronggo Waluyo, Puseurjaya, Telukjambe Timur, Karawang, Jawa Barat 41361

Keywords:
Efektivitas Pengembang; Laravel; Node.js; Performa Web; REST API.

Corespondent Email:
eldonimosul2003@gmail.com

Abstrak. Pemilihan framework backend yang tepat sangat penting dalam pengembangan web modern, khususnya untuk menyeimbangkan performa dan efektivitas pengembangan. Penelitian ini membandingkan Laravel (PHP) dan Node.js (JavaScript) melalui pengembangan REST API aplikasi To-Do List. Metode yang digunakan adalah eksperimen kuantitatif dengan parameter Response Time, Throughput, Resource Usage (CPU & RAM), serta Development Experience (waktu pengembangan dan maintainability). Pengujian beban dilakukan menggunakan k6 dengan 50 virtual users. Hasil menunjukkan perbedaan performa yang jelas. Node.js unggul pada operasi I/O-bound (CRUD) dengan throughput hingga tiga kali lebih tinggi dan penggunaan CPU stabil di sekitar 11%, sehingga cocok untuk skalabilitas tinggi. Sebaliknya, Laravel lebih stabil pada proses CPU-bound seperti autentikasi dan enkripsi, sementara Node.js mengalami blocking dalam kondisi tertentu. Dari sisi efektivitas, Laravel sedikit lebih cepat dengan waktu pengembangan 78 menit dibandingkan 83 menit pada Node.js karena arsitektur opinionated dan ekosistem yang matang. Kesimpulannya, pemilihan framework bergantung pada prioritas proyek: Laravel untuk pengembangan cepat dan stabilitas komputasi, sedangkan Node.js untuk lalu lintas data tinggi.

Abstract. *Choosing an appropriate backend framework is essential for modern web development, particularly in balancing performance and development effectiveness. This study compares Laravel (PHP) and Node.js (JavaScript) through the development of a To-Do List REST API. A quantitative experimental approach was applied using performance indicators including Response Time, Throughput, Resource Usage (CPU & RAM), and Development Experience (development time and maintainability). Load testing was conducted using k6 with 50 virtual users. The results demonstrate a clear performance distinction. Node.js outperforms Laravel in I/O-bound operations (CRUD), achieving up to three times higher throughput with stable CPU utilization around 11%, making it suitable for high-scalability scenarios. Conversely, Laravel shows better stability in CPU-bound processes such as authentication and encryption, while Node.js exhibits blocking behavior under certain conditions. In terms of development effectiveness, Laravel is slightly faster, requiring 78 minutes compared to 83 minutes for Node.js, due to its opinionated architecture and mature ecosystem. The study concludes that framework selection depends on project priorities: Laravel is preferable for rapid development and computational stability, whereas Node.js is ideal for high data-traffic environments.*

1. PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah membawa perubahan signifikan dalam dunia pengembangan perangkat lunak, khususnya pada aplikasi web. Pesatnya pertumbuhan ini dibuktikan oleh data survei Netcraft (2024), yang mencatat bahwa jumlah *website* aktif kini telah melampaui satu miliar. Aplikasi web saat ini tidak hanya dituntut memiliki tampilan antarmuka yang menarik, tetapi juga harus mampu memberikan performa tinggi, efisien dalam penggunaan sumber daya, serta responsif terhadap kebutuhan pengguna secara *real-time* [1]. Untuk memenuhi tuntutan tersebut, penggunaan *framework* menjadi sangat penting karena menyediakan struktur kerja yang terstandarisasi untuk mempercepat proses pengembangan aplikasi [2].

Berdasarkan data *Statistics and Data* (2025), Laravel dan ekosistem Node.js mendominasi peta popularitas *framework backend* di kalangan pengembang global. Laravel, yang dibangun di atas bahasa PHP, mengadopsi arsitektur *Model-View-Controller* (MVC) dan dikenal dengan ekosistemnya yang kuat [3]. Sementara itu, Node.js merupakan platform JavaScript *runtime* yang bersifat *event-driven* dan *non-blocking I/O*, membuatnya unggul dalam menangani banyak permintaan secara bersamaan [4].

Dalam praktik pengembangan aplikasi berbasis REST API, kedua *framework* ini memiliki karakteristik yang berbeda. Laravel menawarkan fitur bawaan seperti Eloquent ORM dan autentikasi yang terintegrasi, memudahkan pengembang membangun struktur terorganisir [5]. Sebaliknya, Node.js sering dipadukan dengan Express.js untuk membangun API yang ringan dan fleksibel. Meskipun banyak digunakan, penelitian yang membandingkan performa dan efektivitas keduanya secara komprehensif masih terbatas. Minimnya perbandingan yang mencakup dua sisi—yakni performa teknis (seperti waktu respons, *throughput*, penggunaan sumber daya, dan *error rate*) serta efektivitas pengembangan (waktu pengerjaan, kemudahan penggunaan, dan pemeliharaan kode)—menimbulkan kesenjangan informasi (*research gap*) [6], [7].

Berdasarkan kondisi tersebut, penelitian ini dilakukan untuk menganalisis perbandingan performa teknis dan efektivitas framework Laravel dan Node.js dalam pengembangan REST API. Objek penelitian menggunakan sistem *To-Do List* yang memfasilitasi operasi *Create, Read, Update, Delete* (CRUD) dan autentikasi keamanan. Hasil penelitian ini diharapkan dapat menjadi referensi komprehensif bagi pengembang web maupun institusi akademik dalam menentukan teknologi *backend* yang paling sesuai dengan kebutuhan sistem.

2. TINJAUAN PUSTAKA

2.1. Laravel

Laravel merupakan *framework* berbasis PHP yang mengadopsi arsitektur MVC [8]. Laravel menyediakan *Object-Relational Mapping* (ORM) bernama Eloquent yang memudahkan interaksi dengan basis data menggunakan pendekatan berorientasi objek. Secara fundamental, PHP beroperasi dengan model *shared-nothing architecture* di mana setiap *request* HTTP yang masuk memicu proses PHP baru. Model ini memungkinkan isolasi antar *request* yang baik, namun dalam skenario beban tinggi, pembentukan proses baru dapat menimbulkan *overhead* sumber daya sistem [9].

2.2. Node.js

Node.js adalah lingkungan *runtime* JavaScript yang bersifat *event-driven* dan *non-blocking I/O*. Arsitektur ini memungkinkan Node.js menangani banyak permintaan simultan pada *single-thread* tanpa memblokir proses lainnya [10]. Node.js sering dipadukan dengan Express.js, sebuah *framework* minimalis yang menyederhanakan manajemen *routing* dan *middleware*, menjadikannya sangat andal untuk aplikasi *real-time* berskala besar dengan kebutuhan akses data yang intensif [11]

2.3. RESTful API

REST (*Representational State Transfer*) API merupakan gaya arsitektur untuk membangun layanan web yang memungkinkan komunikasi antar sistem melalui protokol HTTP [12]. REST API memungkinkan klien untuk melakukan operasi baca, buat, perbarui, dan hapus data pada *server* dengan metode HTTP seperti GET, POST, PUT, dan DELETE. Metode ini telah menjadi standar *de facto* dalam mengintegrasikan layanan antar platform dan *microservices*.

2.4. Evaluasi Kinerja (Performance Testing)

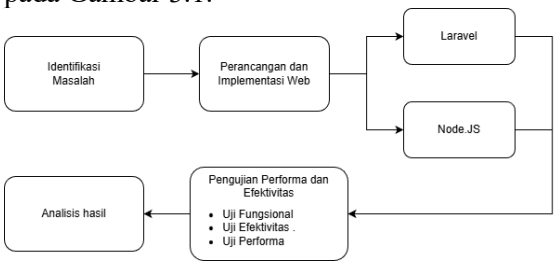
Pengujian performa bertujuan untuk mensimulasikan virtual *user* guna mengetahui kinerja sistem [13]. Kinerja sistem diukur melalui metrik *Response Time* (kecepatan respons server), *Throughput* (kapasitas request per detik/RPS), *Resource Usage* (penggunaan CPU dan Memori), serta *Error Rate*. Penelitian ini menggunakan k6, alat *open-source* berbasis JavaScript yang dirancang khusus untuk *load testing* berskala besar pada API, guna menghasilkan metrik yang komprehensif dan akurat.

2.5. Efektivitas Pengembangan dan Kualitas Kode

Produktivitas dalam pengembangan web dinilai melalui waktu pengembangan, pengalaman pengembang (*developer experience*), dan pemeliharaan kode [14], [15]. Pemeliharaan kode diukur menggunakan *Maintainability Index* (MI), yang mengkalkulasikan skor (0-100) berdasarkan kompleksitas siklomatik (*Cyclomatic Complexity*), volume Halstead, dan jumlah baris kode (LOC) [16]. Alat seperti PhpMetrics (untuk PHP) dan Plato (untuk Node.js) digunakan untuk mendapatkan metrik statis ini secara objektif

3. METODE PENELITIAN

Penelitian ini menggunakan Metode Eksperimen Kuantitatif dengan melakukan pengujian langsung terhadap Laravel 12.43.1 dan Node.js v22.19.0 dalam membangun layanan REST API. Pendekatan ini bertujuan untuk menguji hubungan sebab-akibat secara terukur dengan memanipulasi variabel *framework* pada lingkungan pengujian yang identik. Tahapan penelitian divisualisasikan pada Gambar 3.1.



Gambar 3.1 Metodologi Penelitian

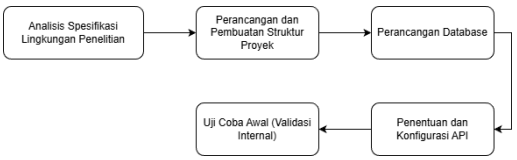
Berdasarkan Gambar 3.1, penelitian dibagi menjadi empat tahapan utama:

3.1.1. Identifikasi Masalah

Menentukan parameter pengujian yang mencakup *response time, throughput, resource usage, error rate*, serta aspek efektivitas seperti waktu pengembangan dan *Maintainability Index*.

3.1.2. Perancangan Implementasi Aplikasi Web

Membangun dua aplikasi *To-Do List* dengan fungsionalitas CRUD dan autentikasi yang identik secara paralel. Alur pengembangan sistem ini dirancang menggunakan prinsip MVC seperti yang diilustrasikan pada Gambar 3.2.



Gambar 3.2 Workflow Pengembangan Aplikasi Web

3.1.3. Pengujian Performa dan Efektivitas

Pengujian efektivitas dilakukan dengan mencatat waktu pengembangan (*time tracking*) dan menganalisis kode statis. Pengujian fungsional dilakukan dengan Postman , sementara *benchmarking* performa dilakukan menggunakan k6 dengan simulasi 50 *Virtual Users* selama 30 detik untuk setiap iterasi (sebanyak 30 iterasi).

3.1.4. Analisis Hasil

Data efektivitas dianalisis secara deskriptif komparatif, sedangkan data performa diuji menggunakan statistik inferensial (Uji Normalitas Shapiro-Wilk dan Uji Hipotesis *Mann-Whitney/Independent T-Test*) melalui *software* JASP.

4. HASIL DAN PEMBAHASAN

4.1. Lingkungan dan Perancangan Sistem

Pengembangan dan pengujian sistem dalam penelitian ini dilakukan pada lingkungan perangkat keras dan perangkat lunak yang terkontrol guna memastikan keadilan dan validitas komparasi antara kedua *framework*. Rincian spesifikasi lingkungan pengujian yang digunakan dijabarkan pada Tabel 4.1.

Tabel 4.1 Spesifikasi Lingkungan Pengujian

Kategori	Komponen	Spesifikasi
Perangkat Keras	Prosesor	AMD Ryzen 5 3500U
	Memori (RAM)	8 GB
Perangkat Lunak	Sistem Operasi	Windows 11
	Basis Data	MariaDB versi 10.4.32
	Framework Backend	Laravel 12.43.1 (PHP), Node.js v22.19.0 (JavaScript)
	Alat Pengujian	Postman (Fungsional), k6 (Load Testing)

Basis data dirancang secara terpusat menggunakan MariaDB untuk menjamin konsistensi *indexing* dan beban *I/O (Input/Output)* pada kedua sistem. Logika bisnis inti diimplementasikan sedemikian rupa agar urutan eksekusi programnya simetris dan seimbang. Sebagai contoh, Gambar 4.3 dan 4.4 menunjukkan perbandingan *snippet* kode *controller* pada saat pembuatan tugas (*task*) baru.

```php
public function createTask(Request $request) {
    try {
        $task = Task::create([
            'user_id' => $request->user()->id,
            'title' => $request->title,
            'description' => $request->description,
            'is_completed' => false
        ]);
        return response()->json(['status' => 'success', 'data' => $task], 201);
    } catch (\Exception $e) {
        return response()->json(['status' => 'error', 'message' => $e->getMessage()], 500);
    }
}
```

Gambar 4.3 Implementasi Fungsi createTask Laravel

```javascript
export const createTask = async (req, res) => {
    try {
        const { title, description } = req.body;
        const task = await Task.create({
            user_id: req.user.id,
            title,
            description,
            is_completed: false
        });
        res.status(201).json({ status: 'success', data: task });
    } catch (error) {
        res.status(500).json({ status: 'error', message: error.message });
    }
};
```

Gambar 4.4 Implementasi Fungsi createTask Node.js

Meskipun sintaksnya berbeda, kedua *framework* secara otomatis mengekstrak user_id dari token, memetakan deskripsi, dan mengembalikan status HTTP 201 *Created* jika berhasil. Keduanya juga mengadopsi mekanisme keamanan berbasis *Bearer Token* (menggunakan Sanctum untuk Laravel dan JWT untuk Node.js).

4.2. Hasil Pengujian Efektivitas

Pengujian efektivitas bertujuan menilai produktivitas pengembang melalui waktu pengembangan, kemudahan penggunaan, dan pemeliharaan kode.

Tabel 4.2 Perbandingan Waktu Pengembangan (Menit).

Tahapan Pengembangan	Laravel (Menit)	Node.js (Menit)
Instalasi dan Konfigurasi Awal (.env, DB)	16	14
Implementasi CRUD Entitas *User*	21	27
Implementasi CRUD Entitas *Task*	29	23
Implementasi Sistem Autentikasi (Sanctum/JWT)	12	19
Total Waktu Pengerjaan	**78**	**83**

Berdasarkan tabel 4.2, pengembangan menggunakan *framework* Laravel membutuhkan waktu total 78 menit, lebih efisien dibandingkan Node.js yang membutuhkan waktu 83 menit. Keunggulan Laravel terlihat dari fitur bawaannya yang memangkas waktu penulisan kode berulang [10].

Selain dari segi waktu, efektivitas juga diukur dari kendala teknis yang dihadapi selama proses pengembangan, yang dirangkum dalam matriks kemudahan penggunaan.

Tabel 4.3 Ringkasan *Developer experience*

Indikator Kemudahan	Laravel (Frekuensi)	Node.js (Frekuensi)
Frekuensi Membuka Dokumentasi Resmi	11	7
Frekuensi Mencari Solusi *Error* (StackOverflow)	5	3

Merujuk pada tabel 4.3, proses pengembangan dengan Laravel dinilai lebih sistematis berkat ekosistem *batteries-included* dan dokumentasi resmi yang sangat terstruktur. Pengembang tidak perlu mencari pustaka (*library*) pihak ketiga untuk fitur standar seperti autentikasi atau ORM. Sebaliknya, Node.js menuntut pengembang untuk secara manual memilih, merangkai, dan mengonfigurasi pustaka eksternal (seperti Express, Mongoose, atau Sequelize) untuk membentuk arsitektur MVC. Meskipun memberikan fleksibilitas tinggi, hal ini meningkatkan kompleksitas awal dan potensi *error* kompatibilitas antar modul [11].

Tabel 4.4 *Maintainability Index Score*

Komponen Kode	Skor MI Laravel	Skor MI Node.js
UserController	39.2	58.52
TaskController	57.02	57.85
Rata-rata Skor MI	48.11	58,18

Sebagai evaluasi jangka panjang, kualitas pemeliharaan kode dianalisis menggunakan alat analisis statis (PhpMetrics untuk Laravel dan Plato untuk Node.js) seperti yang ditunjukkan pada tabel 4.4. Pada parameter ini, Node.js mencatat skor *Maintainability Index* (MI) rata-rata sebesar 58,18, lebih tinggi dari Laravel yang berada di angka 48,11. Keunggulan Node.js di sini disebabkan oleh karakteristik sintaks JavaScript yang lebih ringkas. Sementara itu, struktur *Object-Oriented Programming* (OOP) Laravel yang ketat dan kompleks (penggunaan *namespace, traits,* dan *dependency injection*) cenderung menambah volume baris kode dan kompleksitas siklomatik, yang berdampak pada penurunan skor MI secara matematis [6].

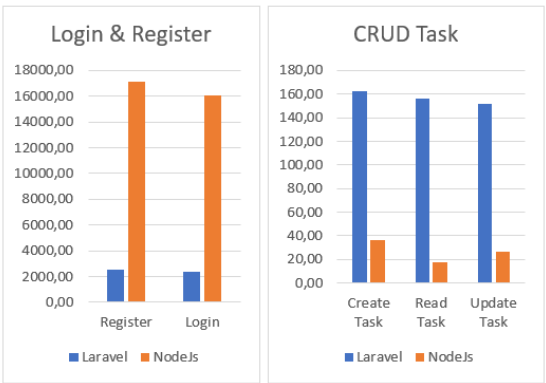
4.3. Hasil Pengujian Performa

Pengujian performa dilakukan untuk membandingkan stabilitas dan efisiensi teknis kedua *framework*.

Tabel 4.5 Statistik Deskriptif Response Time (ms)

Endpoint API	Laravel		Node.js	
	Mean	Std. Dev	Mean	Std. Dev
Register	2.521,89	187,9	17.092,41	600,2
Login	2.380,34	44,04	16.018,88	526,7
Create Task	162,23	6,210	35,96	7,82
Read Task	156,44	6,927	17,86	5,19
Update Task	151,66	16,15	26,79	9,32

Data tabel 4.5 menunjukkan Node.js berkinerja superior pada fitur CRUD (*I/O-bound*) dengan waktu respons di bawah 40 ms. Akan tetapi, pada fitur *CPU-bound* (Register dan Login yang melibatkan komputasi Bcrypt), Laravel jauh lebih cepat dan stabil di kisaran 2.300 - 2.500 ms, sedangkan Node.js mengalami lonjakan di atas 16.000 ms.



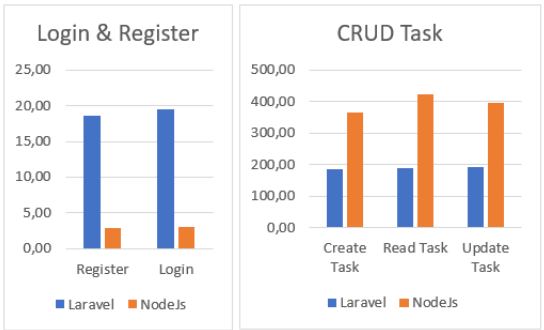
Gambar 4.4 Grafik Perbandingan Rata-rata Response Time

Hasil pengujian beban menggunakan alat k6 dengan 50 *virtual users* menunjukkan adanya perbedaan performa yang signifikan pada metrik *response time*. Gambar 1 mengilustrasikan bahwa Node.js mencatatkan waktu respons yang jauh lebih cepat pada operasi *I/O-bound* seperti proses CRUD (Create, Read, Update, Delete) dibandingkan Laravel. Kecepatan ini sangat dipengaruhi oleh arsitektur *non-blocking I/O* pada Node.js yang memungkinkannya memproses banyak permintaan secara asinkron tanpa harus menunggu proses sebelumnya selesai. Di sisi lain, Laravel menunjukkan performa yang konsisten namun mencatatkan *response time* yang sedikit lebih tinggi, yang merupakan karakteristik wajar dari arsitektur *synchronous* berbasis PHP saat menangani permintaan secara bersamaan.

Tabel 4.6 Statistik Deskriptif Throughput (RPS)

Endpoint API	Laravel		Node.js	
	Mean	Std. Dev	Mean	Std. Dev
Register	18,56	0,478	2,907	0,102
Login	19,55	0,381	2,964	0,095
Create Task	184,1	4,527	366,2	20,42
Read Task	188,8	5,178	422,6	17,16
Update Task	192,7	12,41	394,2	26,70

Pada pengujian kapasitas maksimal di tabel 4.6, Node.js mencatat *throughput* sangat masif pada operasi *Read Task* hingga 422,6 RPS, sekitar 2,2 kali lipat dari Laravel (188,8 RPS) [2].



Gambar 4.5 Grafik Perbandingan Rata-rata *Throughput*

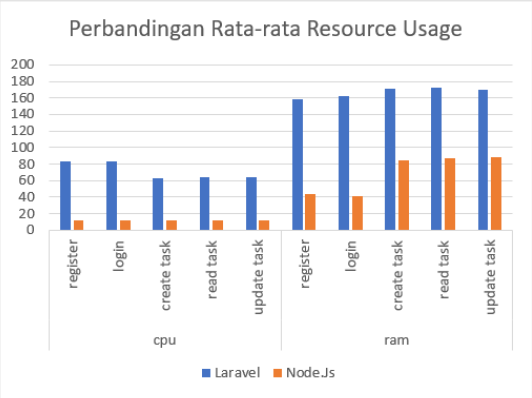
Pada pengujian metrik *throughput*, Node.js menunjukkan keunggulan yang sangat mencolok. Seperti yang divisualisasikan pada Gambar 2, ekosistem JavaScript ini mampu menangani jumlah permintaan per detik (req/s) hingga tiga kali lipat lebih tinggi. Data ini membuktikan tingginya efisiensi *single-threaded event loop* pada Node.js dalam mengelola ribuan koneksi ringan secara simultan tanpa membebani antrean peladen. Meskipun *throughput* Laravel terpaut lebih rendah, *framework* ini tetap mempertahankan stabilitas dan rasio keberhasilan (*success rate*) yang sangat baik, terutama pada operasi *CPU-bound* seperti proses autentikasi (kriptografi *password*), sehingga tetap andal untuk tugas-tugas komputasi yang berat

Tabel 4.7 Statistik Deskriptif Penggunaan CPU dan RAM

EndpointAPI	Metrik	Laravel		Node.js	
		Mean	Std. Dev	Mean	Std. Dev
Register	CPU (%)	82,66	3,79	11,65	0,18
	RAM (MB)	158,44	4,64	44,20	4,49
Login	CPU (%)	83,02	3,56	11,57	0,14
	RAM (MB)	161,84	3,47	41,34	2,29
Create Task	CPU (%)	62,41	2,78	11,71	0,11
	RAM (MB)	171,78	7,25	84,78	4,95
Read Task	CPU (%)	64,17	2,55	11,61	0,15
	RAM (MB)	172,20	6,68	86,63	3,08
Update Task	CPU (%)	63,96	2,68	11,69	0,15
	RAM (MB)	169,30	1,62	88,11	3,28

Tingkat efisiensi perangkat keras diringkas pada tabel 4.7. Node.js mempertahankan penggunaan CPU yang sangat rendah dan datar di kisaran 11% [1]. Laravel, karena menggunakan model *multi-process*, secara konsisten mengalokasikan RAM besar (158-

172 MB) dan memicu penggunaan CPU hingga 83,02% pada komputasi autentikasi. Seluruh metrik performa ini telah divalidasi signifikansinya secara statistik (p-value < 0,001). Adapun *Error Rate* untuk kedua sistem berada di angka sempurna 0% [17].



Gambar 4.6 rafik Perbandingan Rata-rata *CPU Usage*

Selain ketiga metrik performa di atas, pengujian k6 juga memantau tingkat kegagalan permintaan atau *Error Rate*. Selama simulasi 50 *Virtual Users* dengan total ribuan iterasi *request* pada seluruh *endpoint*, baik Laravel maupun Node.js mencatat **Error Rate sebesar 0%**. Hal ini menunjukkan bahwa kedua *framework* memiliki tingkat keandalan (*reliability*) yang sangat baik dan mampu menangani beban maksimal yang diujikan tanpa ada satu pun *request* yang gagal diproses (seluruh respons berstatus HTTP 200 OK) [17].

Tabel 4.8 Rekapitulasi Error rate pengujian API

Endpoint API	Framework	Jumlah Iterasi	HTTP Success	HTTP Error	Error Rate (%)
Register	Laravel	30	100%	0	0%
	Node.js	30	100%	0	0%
Login	Laravel	30	100%	0	0%
	Node.js	30	100%	0	0%
Create Task	Laravel	30	100%	0	0%
	Node.js	30	100%	0	0%
Read Task	Laravel	30	100%	0	0%
	Node.js	30	100%	0	0%
Update Task	Laravel	30	100%	0	0%
	Node.js	30	100%	0	0%

4.4. Hasil Analisis Statistik

Untuk membuktikan bahwa perbedaan performa antara Laravel dan Node.js bukan terjadi secara kebetulan, dilakukan pengujian statistik inferensial menggunakan perangkat lunak JASP terhadap tiga variabel utama pengujian performa, yaitu: **(1) *Response Time*, (2) *Throughput*, dan (3) *Resource Usage***. Tahap pertama adalah pengujian asumsi normalitas data.

Tabel 4.9 Hasil Uji Normalitas dan Signifikansi Response Time

API	Shapiro-Wilk (p)	Jenis Uji	Nilai Uji (U/ T)	Sig. (p-value)	Kesimpulan
Register	< 0,001	Mann-Whitney	U = 0,00	< 0,001	Signifikan
Login	< 0,001	Mann-Whitney	U = 0,00	< 0,001	Signifikan
Create Task	0,953	Idependent T-test	T = 68,71	< 0,001	Signifikan
Read Task	< 0,001	Mann-Whitney	U = 900,00	< 0,001	Signifikan
Update Task	< 0,001	Mann-Whitney	U = 900,00	< 0,001	Signifikan

Tabel 4.9 merupakan perwakilan dari serangkaian uji statistik untuk seluruh variabel pengujian (*Response Time, Throughput*, dan *Resource Usage*). Karena sebagian besar distribusi data menghasilkan nilai *Shapiro-Wilk* (p) < 0,05 (tidak normal), pengujian dominan menggunakan *Mann-Whitney U Test*. Seluruh parameter pada kedua *framework* membuahkan probabilitas signifikansi (p-value) < 0,001. Hal ini membuktikan secara empiris bahwa dikotomi performa yang terjadi bukan kebetulan, melainkan nyata secara statistik

4.5. Pembahasan

Penelitian ini menemukan dikotomi performa yang kontras. Temuan ini berbeda dengan simpulan penelitian Amarulloh et al. [3] dan Purwanto [4] yang menyatakan Node.js unggul di segala aspek, karena penelitian mereka tidak banyak memicu kelemahan *Single Thread Event Loop* pada Node.js melalui operasi *blocking* seperti enkripsi *password*.

Sebaliknya, pada operasi manipulasi data (*I/O-bound*), hasil ini selaras dengan temuan Azzahidi et al. [2] dan Hadinata & Stianingsih [1] yang mengonfirmasi bahwa arsitektur *non-blocking* Node.js mendominasi secara absolut dalam skalabilitas. Di sisi lain, dari efektivitas pengembangan, ekosistem Laravel terbukti meminimalisir hambatan teknis yang sejalan dengan teori *Developer Experience* [11].

Tabel 4.10 Matriks Kesimpulan Trade-off Laravel vs Node.js

Kriteria Evaluasi	Pemenang	Alasan
Kecepatan Komputasi (CPU)	Laravel	Arsitektur *Multi-process* stabil menangani enkripsi berat, mengatasi limitasi *Single Thread* Node.js pada tugas *CPU-bound* [9].
Kecepatan I/O (Database)	Node.js	Mekanisme *Non-blocking I/O* menangani ribuan *request* konkuren tanpa waktu tunggu (*idle*)[8].
Efisiensi Server (Resource)	Node.js	*Runtime* V8 yang sangat ringan mampu menjaga penggunaan CPU rata-rata tetap stabil di angka ~11%[1].
Produktivitas (Kecepatan Dev)	Laravel	Fitur *scaffolding* dan ekosistem terintegrasi (*batteries-included*) mempercepat fase *prototyping* hingga produksi [10].
Kualitas Kode (Maintainability)	Node.js	Struktur kode yang lebih ringkas (*concise*) menghasilkan skor *Maintainability Index* lebih tinggi, yang sejalan dengan standar metrik evaluasi kualitas perangkat lunak modern [6].

Matriks kompromi teknologi pada tabel 4.10 menegaskan bahwa Laravel cocok untuk produktivitas waktu dan fungsionalitas komputasi stabil, sementara Node.js optimal untuk efisiensi *server* dan trafik layanan yang padat.

5. KESIMPULAN

Berdasarkan analisis perbandingan yang telah dilakukan, disimpulkan bahwa tidak ada *framework* yang mendominasi di semua aspek, melainkan sebuah keputusan *trade-off* yang bergantung pada kebutuhan sistem [8]. Berikut adalah rincian kesimpulannya:

a. **Performa Manipulasi Data (*I/O-bound*):** Node.js sangat superior dalam menangani lalu lintas data dengan *throughput* mencapai 422,6 RPS (2,2 kali lipat dari Laravel). Node.js juga sangat efisien dalam menghemat sumber daya *server*, dengan penggunaan CPU yang stabil di kisaran 11% [1]
b. **Stabilitas Komputasi (*CPU-bound*):** Laravel memiliki keunggulan mutlak dalam menangani beban komputasi berat, seperti proses enkripsi *password*, dengan tingkat stabilitas waktu respons yang jauh lebih baik dan aman dari risiko *blocking* dibandingkan Node.js [13].
c. **Efektivitas Pengembangan:** Ekosistem *batteries-included* bawaan Laravel menjadikannya pilihan yang lebih efektif untuk pengembangan yang cepat. Laravel mampu memangkas waktu pengerjaan menjadi 78 menit, lebih efisien dibandingkan Node.js yang membutuhkan waktu 83 menit [10].

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada orang tua penulis yang selalu mendukung penulis dalam setiap langkahnya, dan kepada Bapak Mohamad Jajuli, M.Si., Bapak Iqbal Maulana, S.Si., M.Sc., serta seluruh Dosen dan Staff Fakultas Ilmu Komputer Universitas Karawang yang telah memberikan dukungan selama proses penelitian ini..

DAFTAR PUSTAKA

[1] W. Hadinata and L. Stianingsih, "Analisis Perbandingan Performa Restfull Api Antara Express.Js Dengan Laravel Framework," *Jurnal Informatika dan Teknik Elektro Terapan (JITET)*, vol. 12, no. 1, 2024.

[2] A. S. Azzahidi, B. Wijayanto, and A. Darmawan, "Performance evaluation of backend frameworks for REST API: A comparative study of Spring Boot, Flask, Express.js, Laravel FrankenPHP, and Gin," *Jurnal Teknik Informatika (JUTIF)*, vol. 6, no. 4, pp. 2405-2419, 2025.

[3] A. Amarulloh, K. Kurniasih, and M. Muchlis, "Analisis Perbandingan Performa Web Service Rest Menggunakan Framework Laravel, Django, Dan Node Js Untuk Akses Data Dengan Aplikasi Website," *Jurnal Teknik Informatika*, vol. 9, no. 1, pp. 14-19, 2023.

[4] T. Purwanto, "Analisa Perbandingan Kinerja Rest Api Dengan Framework Flask, Laravel, Dan Express Js," *Scientia Sacra: Jurnal Sains, Teknologi dan Masyarakat*, vol. 3, no. 4, pp. 49-55, 2023.

[5] A. N. J. A. M. Al, W. H. N. Putra, and D. W. Brata, "Perbandingan Pengujian Performance Framework Laravel Dan Lumen Pada Arsitektur Berbasis Service Sobat-Ps Menggunakan Jmeter Dan Postman (Studi Kasus: Kups Kalimantan Utara)," *Jurnal Sistem Informasi, Teknologi Informasi, dan Edukasi Sistem Informasi*, vol. 5, no. 2, pp. 120-138, 2024.

[6] F. Arcelli Fontana, V. Ferme, and I. Pigazzini, "Benchmarking maintainability metrics: a modern approach to software quality," *arXiv preprint arXiv:2402.01917*, 2024.

[7] A. C. Rompis and R. F. Aji, "Perbandingan performa kinerja Node.js, PHP, dan Python dalam aplikasi REST," *CogITo Smart Journal*, vol. 9, no. 1, pp. 88-97, 2023.

[8] R. Kurniawan and E. Syahputra, "Pengembangan REST API dengan pendekatan minimum viable product (MVP) pada sistem informasi," *Jurnal Rekayasa Sistem dan Teknologi Informasi*, vol. 7, no. 1, pp. 45-58, 2023.

[9] A. Putra, B. Santoso, and C. Pratama, "Penerapan arsitektur REST API menggunakan prinsip separation of concerns pada pengembangan perangkat lunak," *Jurnal Informatika dan Teknologi Perangkat Lunak*, vol. 4, no. 2, pp. 115-128, 2022.

[10] H. Wijaya, "Evaluasi performa dan efektivitas framework web backend dalam pengembangan aplikasi skala produksi," *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 9, no. 3, pp. 210-222, 2022.

[11] C. Treude, M. A. Storey, and D. Ford, "An actionable framework for understanding and improving developer experience," *arXiv preprint arXiv:2205.06352*, 2022.

[12] R. G. Guntara and V. Azkarin, "Implementasi dan Pengujian REST API Sistem Reservasi Ruang Rapat dengan Metode Black Box Testing," *Jurnal Minfo Polgan*, vol. 12, no. 1, pp. 1229-1238, 2023.

[13] Y. Ariyanto, M. F. F. Rachmad, and D. Puspitasari, "Laravel framework and native PHP: Comparison in the creation of rest API," *Matrix: Jurnal Manajemen Teknologi Dan Informatika*, vol. 14, no. 2, pp. 66-73, 2024.

[14] F. Sinlae, R. Irwanda, M. R. Maulana, and A. Syahputra, "Penggunaan framework Laravel dalam membangun aplikasi website berbasis PHP," *Jurnal Sistem dan Manajemen Database (JSMD)*, vol. 2, no. 2, pp. 119-132, 2024.

[15] L. Rahmawati and S. Sumarsono, "Desain Pengembangan Website dengan Arsitektur Model View Controller pada Framework Laravel," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 6, no. 4, pp. 785-790, 2024.

[16] A. Ramadhani, N. Iriadi, and R. Hidayat, "Implementasi Teknologi Rest Api Dengan Node Js Untuk Aplikasi Rekomendasi Destinasi Wisata," *Indonesian Journal Computer Science*, vol. 4, no. 1, pp. 22-29, 2025.

[17] D. A. Syahrony, "Analisis Perbandingan Performa dan Resource Management Backend Framework Golang, Node.js, Python dalam API Web Service," Doctoral dissertation, Politeknik Negeri Jember, 2025.