Prosiding Seminar Nasional Sains dan Teknologi Seri III
Fakultas Sains dan Teknologi, Universitas Terbuka
Vol. 2 No. 1 (2025)
e-ISSN: 3047-6569

EFEKTIVITAS PENGGUNAAN NODE JS DALAM PEMBUATAN REST API UNTUK APLIKASI KATASTROFA

Aditya Putra Kejora*, Wahyu Noviani Purwanti
Program Studi Sistem Informasi, Universitas Terbuka, Kota Tangerang Selatan

**Penulis korespondensi: adity.putra14@gmail.com*

ABSTRAK

Kebutuhan akan aplikasi web efisien terus meningkat, terutama untuk pengelolaan data *real-time* dalam jumlah besar. REST API menjadi pendekatan utama untuk integrasi data yang cepat dan terstruktur, dengan Node.js yang merupakan platform runtime *Javascript* semakin populer berkat arsitektur non-blokir dan asinkron yang mampu menangani permintaan dalam jumlah besar. Pendekatan Agile juga dapat meningkatkan fleksibilitas pengembang dalam merespons perubahan kebutuhan sistem dikarenakan pekerjaan dapat dilaksanakan secara paralel, sehingga proses dapat berjalan lebih cepat. Penelitian ini memiliki tujuan untuk menganalisis efektivitas Node.js dalam pengembangan REST API untuk aplikasi Katastrofa, sebuah aplikasi pelaporan kebencanaan berbasis *crowdsourcing*. Fokusnya adalah mengidentifikasi keunggulan, tantangan, serta solusi untuk mengoptimalkan performa Node.js. Proses pengujian perangkat lunak dilakukan dengan menggunakan metode *performance testing*. Metode ini merupakan metode yang efektif untuk menguji kecepatan, respon, stabilitas dan penggunaan *resource*. Hasilnya pengujian *Javascript* dengan menggunakan Node.Js, mampu menunjukkan proses kerja yang lebih optimal dalam menangani berbagai beban kerja sehingga menjadi efektif.

Kata kunci: Node.JS, API, Restfull, Katastrofa, Agile

1 PENDAHULUAN

Kondisi dunia di era digital saat ini, menyebabkan kebutuhan aplikasi berbasis web yang efektif dan efisien semakin meningkat, terutama pada sektor yang mengandalkan pertukaran data dalam jumlah besar secara *real-time*. Salah satu pendekatan yang banyak diadopsi adalah penggunaan *Representational State Transfer Application Programming Interface* (REST API) dalam pengembangan aplikasi. REST API memungkinkan integrasi data dan komunikasi antar sistem secara cepat dan terstruktur, terutama dalam lingkungan yang menuntut akses cepat dan responsivitas yang tinggi. Node.js, sebuah platform berbasis *Javascript* yang terkenal dengan arsitektur non-blokir dan asinkron, semakin populer dalam pengembangan REST API karena kemampuannya menangani jumlah permintaan yang besar dengan waktu respons yang cepat.

Berbagai penelitian telah dilakukan untuk mengevaluasi efektivitas Node.js dalam pengembangan aplikasi berbasis REST API. Sebagai contoh, penelitian oleh Qaisa et al. (2023) mendemonstrasikan bagaimana Node.js dapat digunakan dalam pengembangan aplikasi manajemen tugas yang efektif bagi karyawan di perusahaan tertentu dengan pendekatan Agile. Hasil penelitian menunjukkan bahwa Node.js mampu meningkatkan efisiensi dalam manajemen tugas melalui REST API yang dirancang untuk integrasi yang mulus antar komponen aplikasi. Selain itu, penelitian oleh Purwanto (2023) mengungkapkan bahwa dibandingkan dengan

framework lainnya, seperti Flask dan Laravel, Node.js melalui framework Express.js menunjukkan performa yang unggul dalam hal waktu respons data fetching yang lebih cepat.

Namun, penggunaan Node.js dalam pengembangan REST API juga memiliki tantangan. Berdasarkan tinjauan yang dilakukan oleh Shah dan Soomro (2017), Node.js menghadapi tantangan signifikan terutama dalam hal pengelolaan beban kerja besar dan kompleksitas sistem yang meningkat. Kendati demikian, penelitian oleh Sutara dan Gunawan (2024) menunjukkan bahwa meskipun Express.js populer dalam pengembangan REST API, framework lain seperti Hapi.js dapat menawarkan alternatif dengan waktu respons yang lebih baik meski dengan konsumsi resource pada CPU dan memori yang sedikit lebih tinggi. Selain itu, pendekatan metode Agile seperti yang diuraikan oleh Inayah (2024) dalam penelitian manajemen proyek sistem informasi, mampu meningkatkan fleksibilitas dan responsivitas tim pengembang terhadap perubahan kebutuhan, yang sangat relevan dalam lingkungan pengembangan berbasis Node.js. Hal itu yang menyebabkan pada saat pembuatan Katastrofa digunakan metode agile, karena aplikasi yang dibuat dapat dilakukan testing dan perbaikan secara paralel, sehingga proses *update* menjadi lebih cepat dan efisien.

Dengan latar belakang ini, penelitian ini bertujuan untuk menganalisis efektivitas Node.js dalam pengembangan REST API pada aplikasi Katastrofa.

Gambar 1. Tampilan Aplikasi Katastrofa

Katastrofa adalah sebuah aplikasi pelaporan kebencanaan berbasis *crowdsourcing* dengan mengumpulkan data menggunakan aktifitas pengguna dalam melaporkan kebencanaan. Aplikasi ini mirip dengan PetaBencana.id yang juga merupaka aplikasi pelaporan bencana yang disampaikan oleh Sulaeman (2021). Hal yang menjadi membedakan adalah pada aplikasi

katastrofa ini juga menunjukan cuaca yang *realtime* pada saat mengakses aplikasi tersebut serta dapat memberikan komentar untuk mendukung atau mengevaluasi laporan yang dikirimkan. Kajian ini diharapkan dapat memberikan kontribusi yang signifikan dalam mengidentifikasi kelebihan, tantangan, serta solusi untuk pengoptimalan performa Node.js dalam mengelola arsitektur REST API, sekaligus menegaskan peran penting metode Agile dalam meningkatkan keberhasilan proyek pengembangan aplikasi berbasis REST API. Sehingga aplikasi yang dibuat dapat berjalan secara optimal, terutama bagian *backend*.

2 METODE

Metode yang digunakan dalam proses penelitian ini adalah *Performance Testing*, yang merupakan salah satu metode pengujian untuk mengukur efektivitas performa penggunaan bahasa pemrograman. Dalam konteks penelitian ini, *Performance Testing* diterapkan untuk membandingkan kinerja antara bahasa pemrograman *Javascript* dan PHP, khususnya dalam hal kecepatan respons serta efisiensi penggunaan sumber daya.

Performance Testing adalah proses evaluasi aplikasi yang dilakukan dengan mensimulasikan aktivitas pengguna melalui penggunaan virtual user. Metode ini memungkinkan peneliti untuk memantau performa sistem dalam kondisi yang menyerupai penggunaan dunia nyata, sehingga hasilnya mencerminkan efisiensi aplikasi yang diuji. Dalam penelitian ini, fokus utama adalah pengujian terhadap Application Programming Interface (API) yang dibangun menggunakan kedua bahasa pemrograman tersebut.

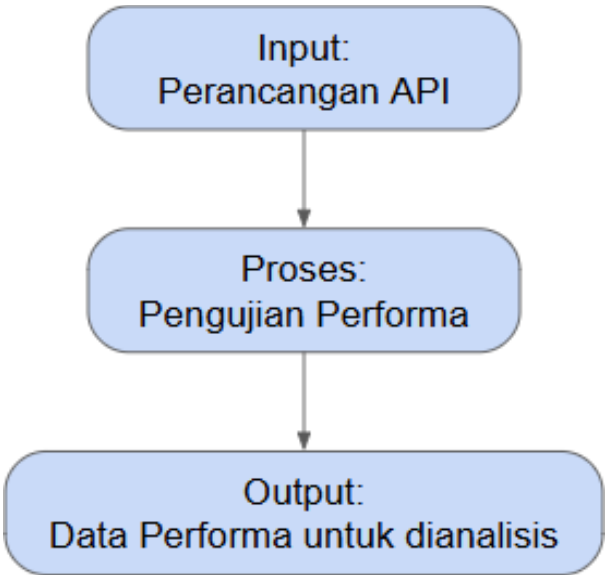
Tujuan dari pengujian ini adalah untuk memberikan gambaran yang objektif mengenai perbandingan kinerja API berbasis *Javascript* dan PHP, terutama dalam hal:

- Waktu respons, yaitu seberapa cepat API dapat memproses permintaan dari klien dan memberikan hasil.
- Konsumsi sumber daya, seperti penggunaan memori dan prosesor, yang dapat mempengaruhi efisiensi operasional sistem.

Melalui eksperimen ini, kinerja API yang dikembangkan dalam *Javascript* dan PHP akan diukur dan dianalisis secara mendalam. Data yang dihasilkan akan menjadi dasar untuk menentukan bahasa pemrograman mana yang lebih unggul dalam konteks spesifik yang diuji. Dengan demikian, penelitian ini dapat memberikan wawasan yang bermanfaat bagi pengembang perangkat lunak dalam memilih bahasa pemrograman yang sesuai dengan kebutuhan dan prioritas kinerja.
Untuk mendukung validitas dan keakuratan hasil penelitian, proses eksperimen ini dirancang dengan pendekatan metodologi yang terstruktur, meliputi:

- **Perancangan API**: Membuat API sederhana yang memiliki fungsi serupa di kedua bahasa pemrograman.
- **Implementasi Pengujian**: Melakukan simulasi akses API dengan *virtual user* dalam jumlah tertentu untuk mengukur performa di bawah beban.
- **Pengumpulan Data**: Mencatat metrik kinerja seperti waktu respon, *throughput*, dan konsumsi sumber daya pada masing-masing API.
- **Analisis Data**: Membandingkan hasil pengujian untuk menentukan efisiensi relatif antara *Javascript* dan PHP.

  Dengan pendekatan ini, penelitian diharapkan dapat memberikan hasil yang komprehensif dan dapat diandalkan dalam mengevaluasi kinerja API berbasis *Javascript* dan PHP.



Gambar 2. Kerangka Berpikir

Secara lebih rinci, penelitian yang dilakukan meliputi beberapa langkah berikut:

1. Perancangan REST API
   Pada tahapan ini dibuat rancangan REST API yang dibuat menggunakan *Javascript* dan PHP. API tersebut dijalankan dengan alamat masing-masing seperti berikut:
   a. API PHP, dengan alamat http://localhost:8080/api
   b. API *Javascript* , dengan alamat http://localhost:3000/api
2. Desain Eksperimen
   Dipersiapkan dua buah API dengan fungsi yang identik dibuat menggunakan bahasa pemrograman *Javascript* dan PHP yang akan berjalan pada server yang sama.
   a. API PHP berjalan pada PHP Versi 8.2
   b. API *Javascript* berjalan pada node.js versi 22.11.00
3. Lingkungan Pengujian
   Lingkungan yang digunakan adalah dengan menggunakan PC dengan spesifikasi seperti dibawah ini:
   - ***Hardware*** yang digunakan:
     a. Processor : Intel Core i7-4790 @3.60 Ghz
     b. RAM : 16GB DDR3
     c. Disk : SSD 512 GB
     d. OS : Windows 10 Professional
   - ***Software*** yang digunakan dalam proses ini adalah:
     a. Visual Studio Code untuk membuat program
     b. Node JS, untuk menjalankan API *Javascript*
     c. Postman, sebagai alat pengujian
4. Skenario Pengujian
   Skenario yang dibuat adalah dengan melakukan uji performa dengan menggunakan metode GET untuk mengambil data *string* dan *timestamp.* Setiap skenario pengujian

bertujuan untuk mengevaluasi waktu respons API terhadap beban yang berbeda. Parameter yang diamati meliputi waktu respons rata-rata, jumlah permintaan yang berhasil dengan 3 skenario yang masing-masing telah diberi nama, yaitu:

- Skenario *Fixed* 1
  - Deskripsi:
    Skenario ini menggunakan 50 pengguna virtual yang mengakses API secara bersamaan selama 10 menit. Beban ini merepresentasikan situasi dengan jumlah pengguna aktif yang relatif rendah, seperti aplikasi kecil atau situs dengan lalu lintas ringan.
  - Tujuan:
    Mengukur waktu respons rata-rata dan jumlah permintaan berhasil dalam situasi dengan tingkat beban rendah.
  - Harapan:
    API diharapkan memberikan waktu respon yang cepat dan stabil karena beban pengguna relatif kecil.
- Skenario *Fixed* 2
  - Deskripsi:
    Pada skenario ini, 100 pengguna virtual secara bersamaan mengakses API dalam durasi 10 menit. Kondisi ini menggambarkan aplikasi dengan jumlah pengguna sedang, seperti aplikasi internal organisasi atau situs dengan jumlah pengunjung moderat.
  - Tujuan:
    Menilai kinerja API saat beban meningkat menjadi dua kali lipat dibandingkan *Fixed* 1.
  - Harapan:
    API seharusnya tetap mampu menangani permintaan dengan performa stabil, meskipun mungkin terjadi sedikit peningkatan waktu respon dibandingkan *Fixed* 1.
- Skenario *Ramp Up*
  - Deskripsi:
    Skenario ini dirancang untuk meniru kondisi lonjakan jumlah pengguna secara bertahap. Dimulai dengan 25 pengguna virtual di menit pertama, jumlah pengguna akan terus meningkat hingga mencapai 100 pengguna pada akhir sesi pengujian. Durasi pengujian tetap sama, yaitu 10 menit.
  - Tujuan:
    Mengukur kemampuan API dalam menangani peningkatan beban secara dinamis dan mengevaluasi apakah terdapat degradasi performa saat jumlah pengguna bertambah.
  - Harapan:
    API diharapkan mampu mempertahankan kinerja yang stabil, meskipun beban meningkat secara bertahap. Lonjakan pengguna tidak seharusnya menyebabkan kegagalan besar dalam pemrosesan permintaan.

5. Teknik Analisis Data

Data dari hasil pengujian akan dianalisa menggunakan grafik dan uji statistik untuk mengetahui apakah terdapat perbedaan yang signifikan dalam kinerja dari masing-masing API.

3 HASIL DAN PEMBAHASAN

Pada bagian ini, akan dibahas hasil dari setiap tahapan yang telah dilakukan sesuai dengan metode penelitian yang dirancang. Pembahasan disusun secara sistematis berdasarkan tahapan penelitian, dimulai dari analisis kebutuhan hingga pengujian API. Setiap tahapan akan dijelaskan, dilengkapi dengan hasil pengujian, tabel, grafik, serta penjelasan mengenai kondisi pengujian yang dilakukan.

1. Tahap Perancangan REST API

Pada tahap ini dilakukan perancangan REST API dari masing-masing API baik *Javascript* maupun PHP yang mampu menangani request GET dengan menampilkan data JSON seperti pada Tabel 1 dibawah ini.

Tabel 1. Hasil Request GET endpoint /api

Javascript	PHP

{
"message": "Hello from Node.js!",
"timestamp": "2024-12-10T03:14:46.312Z"
}
	{
	"message": "Hello from PHP!",
	"timestamp": "2024-12-10T03:14:46.312Z"
	}

Serta dengan *response code 200* sebagai tanda bahwa request berhasil.

Hasil:

- Perancangan telah sesuai dengan tujuan
- *Endpoint* / API mendukung untuk *request* GET dan menghasilkan *output* yang diharapkan

2. Desain Eksperimen

Pada tahap ini, API *Javascript* disiapkan menggunakan Node.JS versi 22.11.00 dan API PHP dipersiapkan menggunakan PHP versi 8.2.

Hasil:

- API *Javascript* dengan Node.JS versi 22.11.00 dapat berjalan dengan lancar di alamat http://localhost:3000 dengan endpoint /api
- API PHP dengan PHP versi 8.2 dapat berjalan dengan lancar di alamat http://localhost:8080 dengan endpoint /api

3. Lingkungan Pengujian

Lingkungan pengujian yang disiapkan menggunakan Windows 10 sebagai *Operating System* (OS), 16GB RAM DDR3, Disk SSD 256GB serta menggunakan aplikasi Visual Studio Code sebagai code editor dan Postman sebagai aplikasi pengujiannya.

Hasilnya adalah dengan lingkungan pengujian diatas, proses pengujian dapat dilakukan dengan lancer.

4. Skenario Pengujian

Pengujian yang dilakukan menggunakan 3 skenario yang masing-masing memiliki spesifikasi sebagai berikut:

- Skenario *Fixed* 1

  Skenario ini menggunakan 50 pengguna virtual yang mengakses API secara bersamaan selama 10 menit. Beban ini merepresentasikan situasi dengan jumlah pengguna aktif yang

relatif rendah, seperti aplikasi kecil atau situs dengan lalu lintas ringan. Tujuannya untuk mengukur waktu respons rata-rata dan jumlah permintaan berhasil dalam situasi dengan tingkat beban rendah.

- Skenario Fixed 2
  Skenario ini digunakan 100 pengguna virtual secara bersamaan mengakses API dalam durasi 10 menit. Kondisi ini menggambarkan aplikasi dengan jumlah pengguna sedang, seperti aplikasi internal organisasi atau situs dengan jumlah pengunjung moderat. Tujuannya untuk menilai kinerja API saat beban meningkat menjadi dua kali lipat dibandingkan *Fixed* 1.
- Skenario *Ramp Up*.
  Skenario ini dirancang untuk meniru kondisi lonjakan jumlah pengguna secara bertahap. Dimulai dengan 25 pengguna virtual di menit pertama, jumlah pengguna akan terus meningkat hingga mencapai 100 pengguna pada akhir sesi pengujian. Durasi pengujian tetap sama, yaitu 10 menit. Dengan tujuan mengukur kemampuan API dalam menangani peningkatan beban secara dinamis dan mengevaluasi apakah terdapat degradasi performa saat jumlah pengguna bertambah.

Dari proses pengujian yang dilaksanakan, didapatkan hasil pengujian API PHP dan *Javascript* yang telah dirancang dalam bentuk hasil dari Postman yang selanjutnya diolah dalam bentuk tabel dan grafik seperti berikut:

1. *Fixed* 1
   PHP

Gambar 3. *Report PHP Testing Fixed* 1

Gambar 3 adalah *report* dari pengujian pada tahap Fixed 1 pada API PHP

Gambar 4. *Detail Report PHP Testing Fixed* 1

Gambar 4 merupakan detail dari pengujian pada tahap *Fixed* 1 pada API PHP

Javascript

Gambar 5. *Report Javascript Testing Fixed* 1

Gambar 5 adalah *report* dari pengujian pada tahap Fixed 1 untuk API *Javascript*

Gambar 6. *Detail Report Javascript Testing Fixed* 1

Gambar 6 merupakan detail dari pengujian pada tahap *Fixed* 1 pada API *Javascript*

2. *Fixed* 2

PHP

Gambar 7. *Report PHP Testing Fixed* 2

Gambar 7 adalah *report* dari pengujian pada tahap *Fixed* 2 pada API PHP

Gambar 8. *Detail Report PHP Testing Fixed* 2

Gambar 8 merupakan detail dari pengujian pada tahap *Fixed* 2 pada API PHP

Javascript

Gambar 9. *Report Javascript Testing Fixed* 2

Gambar 9 adalah *report* dari pengujian pada tahap *Fixed* 2 untuk API *Javascript*

Gambar 10. *Detail Report Javascript Testing Fixed* 2

Gambar 10 merupakan detail dari pengujian pada tahap *Fixed* 2 pada API *Javascript*

3. *Ramp Up*

PHP

Gambar 11. *Report PHP Testing Ramp Up Test*

Gambar 11 adalah *report* dari pengujian pada tahap *Ramp Up* pada API PHP

2. Metrics for each request

The requests are shown in the order they were sent by virtual users.

Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
GET PHP TEST http://localhost:8080/api	38,767	63.80	3	6	8	143	0

Gambar 12. *Detail Report PHP Testing Ramp Up Test*

Gambar 12 merupakan detail dari pengujian pada tahap *Ramp Up* pada API PHP
Javascript

JS Ramp Test - Dec 2, 2024 (#15)

Open in Postman

Postman collection: API Performance Testing
Report exported on: Dec 2, 2024, 9:48:15 (GMT+7)

Test setup

Virtual users	Start time	Load profile
100 VU	Dec 2, 9:27:49 (GMT+7)	Ramp up (4 minutes 40 seconds)
Duration	**End time**	**Environment**
10 minutes	Dec 2, 9:37:57 (GMT+7)	-

1. Summary

Total requests sent	Throughput	Average response time	Error rate
37,503	61.72 requests/second	2 ms	0.00 %

Gambar 13. *Report Javascript Testing Ramp Up Test*

Gambar 13 adalah *report* dari pengujian pada tahap Fixed 1 untuk API *Javascript*

2. Metrics for each request

The requests are shown in the order they were sent by virtual users.

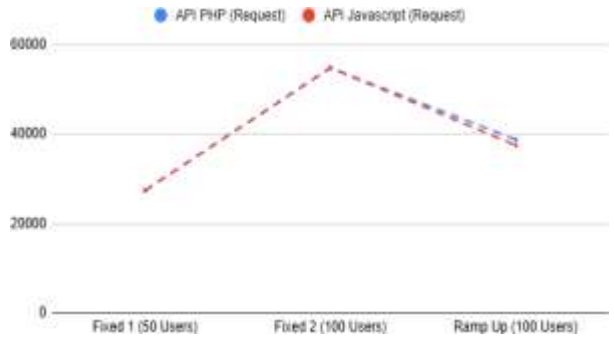
Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
GET JS TEST http://localhost:3000/api	37,503	61.72	1	2	3	218	0

Gambar 14. *Detail Report Javascript Testing Ramp Up Test*

Gambar 14 merupakan detail dari pengujian pada tahap *Testing Ramp Up Test* pada API *Javascript*
Dari hasil pengujian diatas, kemudian dirangkum dalam tabel dan grafik dibawah ini:

Tabel 2. Total *Request* Per Pengujian

Skenario	API PHP (Request)	API *Javascript* (Request)
Fixed 1 (50 Users)	27383	27391
Fixed 2 (100 Users)	54733	54854
Ramp Up (100 Users)	38767	37503

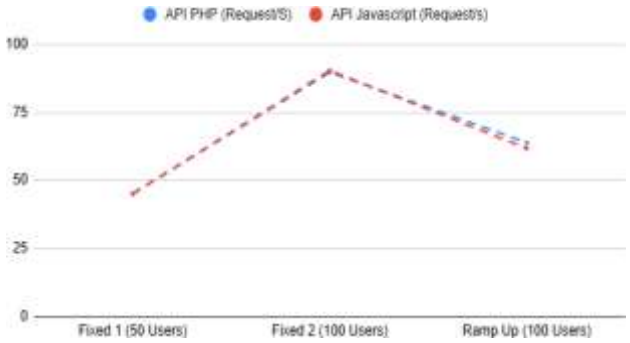


Gambar 15. Grafik Total *Request*

Pada Tabel 2 dan Gambar 15 diatas, terdapat perbedaan yang tipis terkait total *request* yang diterima oleh masing-masing API, dimana *Javascript* unggul dalam menangani *request.*

Tabel 3. *Request Per second*

Skenario	API PHP (Request/s)	API *Javascript* (Request/s)
Fixed 1 (50 Users)	45,08	45,05
Fixed 2 (100 Users)	89,72	90,33
RampUp (100 Users)	63,8	61,72



Gambar 16. Grafik *Request Per Second*

Pada Tabel 3 dan Gambar 16 diatas, terdapat perbedaan yang tipis terkait total *request/second* yang dapat ditangani oleh masing-masing API. Namun *Javascript* masih unggul beberapa poin dalam penanganan *request.*

5. Teknik Analisis data
   Hasil dari proses pengujian kemudian dirangkum dalam tabel 4 yang lebih lengkap terkait pengujian pada API *Javascript* dan PHP sebagai berikut:

Tabel 4. Rekap Hasil Pengujian

Skenario	*Request/Second*		Jumlah *Request*		Average (ms)	
	PHP	Js	PHP	Js	PHP	Js
Fixed 1 (50 Users)	45,08	45,05	27383	27391	5	2
Fixed 2 (100 Users)	89,72	90,33	54733	54854	6	2
Ramp Up (100 Users)	63,8	61,72	38767	37503	6	2

Dari hasil pengujian di atas, *Javascript* terbukti unggul dalam hal:

- **Waktu Respons Stabil:** Konsisten memberikan waktu respons rata-rata 2 ms di semua skenario.
- **Kemampuan Menangani Beban:** Mampu menangani jumlah *request* yang lebih banyak dibandingkan PHP, terutama pada skenario Fixed 2.
- **Efisiensi pada Sistem Real-Time:** *Javascript* cocok digunakan pada aplikasi yang membutuhkan respons cepat dan beban kerja intensif.

Namun, perlu diketahui juga bahwa pada skenario *Ramp Up*, kemampuan PHP sedikit lebih unggul dalam hal *request per second*. Hal ini dapat menjadi pertimbangan bagi pengembang yang membutuhkan stabilitas performa di situasi peningkatan beban bertahap. Berdasarkan hasil pengujian ini, *Javascript* layak dipertimbangkan sebagai bahasa pemrograman utama untuk proyek yang membutuhkan performa tinggi, terutama untuk aplikasi *real-time* dan sistem dengan beban kerja besar. PHP tetap relevan untuk kasus di mana efisiensi *resource* dan stabilitas pada skenario bertahap lebih diutamakan.

4 SIMPULAN DAN SARAN

Berdasarkan hasil pengujian performa API menggunakan bahasa pemrograman *Javascript* dan PHP, disimpulkan bahwa *Javascript* menunjukkan performa yang lebih unggul dalam hal kecepatan respons dan konsistensi waktu respons di semua skenario pengujian. Rata-rata waktu respons *Javascript* tercatat hanya **2 ms**, jauh lebih cepat dibandingkan PHP yang membutuhkan waktu **5-6 ms**. Inilah yang membuat bahasa pemrograman *Javascript* menjadi salah satu pilihan utama untuk aplikasi yang memerlukan waktu respon cepat, seperti aplikasi real-time atau layanan dengan banyak pengguna.

Selain itu, *Javascript* juga dapat menangani jumlah permintaan yang lebih banyak dibandingkan PHP, terutama pada hasil dari skenario Fixed 2, dimana *Javascript* dapat memproses **54854 *request*** dibandingkan dengan PHP yang hanya mencapai **54733 request**. Keunggulan ini menunjukkan bahwa *Javascript* lebih efisien dalam menangani beban kerja tinggi.

Namun, PHP memiliki keunggulan dalam stabilitas performa pada skenario peningkatan beban bertahap, seperti yang terlihat pada skenario *Ramp Up*, dimana PHP mencatat *request per second* lebih tinggi dibandingkan *Javascript* . Hal ini menjadikan PHP tetap relevan untuk digunakan dalam aplikasi dengan kebutuhan stabilitas yang tinggi dan beban kerja bertahap.

Secara keseluruhan, *Javascript* adalah pilihan yang sangat baik untuk aplikasi modern dengan kebutuhan performa tinggi dan respons real-time. Sementara itu, PHP masih merupakan alternatif

yang solid untuk aplikasi yang lebih sederhana atau membutuhkan efisiensi sumber daya yang lebih tinggi. Beberapa saran yang dapat dijadikan panduan dalam pengembangan dan penerapan aplikasi menggunakan bahasa pemrograman *Javascript* maupun PHP.

1. Pemilihan bahasa pemrograman harus disesuaikan dengan kebutuhan spesifik proyek. *Javascript* , dengan kecepatan respon yang tinggi dan waktu eksekusi yang stabil, sangat cocok untuk aplikasi yang memerlukan performa *real-time* seperti sistem monitoring, aplikasi *chatting*, atau layanan *streaming*. Di sisi lain, PHP tetap menjadi pilihan yang tepat untuk aplikasi dengan beban kerja moderat dan kebutuhan stabilitas performa, seperti sistem manajemen konten (CMS) atau aplikasi yang tidak memerlukan banyak koneksi simultan.
2. Optimalisasi penggunaan *resource* sangat penting, terutama jika menggunakan *Javascript* yang memiliki konsumsi *resource* lebih tinggi dibandingkan PHP. Hal ini dapat diatasi dengan memastikan ketersediaan infrastruktur server yang memadai serta mengimplementasikan teknik optimasi seperti caching dan pengelolaan memori yang efisien.
3. Disarankan untuk melakukan pengujian lanjutan pada lingkungan produksi guna memahami pengaruh faktor eksternal, seperti latensi jaringan atau konfigurasi server. Selain itu, pengujian dengan skenario yang lebih kompleks, seperti pengolahan data besar atau integrasi dengan basis data, dapat memberikan wawasan yang lebih menyeluruh terkait performa masing-masing bahasa pemrograman.
4. Kombinasi penggunaan PHP dan *Javascript* dapat menjadi solusi yang optimal untuk proyek tertentu. Misalnya, PHP dapat digunakan sebagai *backend* tradisional, sedangkan *Javascript* (Node.js) dapat diimplementasikan untuk layanan real-time yang membutuhkan respons cepat.

Dengan menerapkan saran-saran ini, diharapkan pengembang dapat memilih dan mengimplementasikan teknologi yang sesuai untuk mencapai performa terbaik sesuai kebutuhan proyek.

DAFTAR PUSTAKA

Inayah, A. D. (2024). Analisis tinjauan implementasi metode Agile dalam manajemen proyek sistem informasi. *Jurnal Teknologi Informasi dan Komunikasi Modern*, 1(2). Diakses dari https://journal.smartpublisher.id/index.php/jurtikom/article/view/118/105

Purwanto, T. (2023). Analisa perbandingan kinerja REST API dengan framework Flask, Laravel, dan Express JS. *Pijar Pemikiran Scientia*, 3(4). Diakses dari https://pijarpemikiran.com/index.php/Scientia/article/view/651/612

Qaisa, R. S., Putri, A., & Maghfirah, H. (2023). Perancangan aplikasi Todo menggunakan Node.js dan REST API. *International Journal of Computer Science*, 12(6). Diakses dari http://ijcs.net/ijcs/index.php/ijcs/article/view/3574/361

Sulaeman, T.. (2021). Peran Surveillans Dalam Pemberdayaan Masyarakat: Studi Kasus Petabencana.id. *Jurnal Intelektiva*, 2(07). Diakses dari https://jurnalintelektiva.com/index.php/jurnal/article/view/423/297

Shah, H., & Soomro, T. R. (2017). Node.js challenges in implementation. *ResearchGate*, 17. Diakses dari https://www.researchgate.net/publication/318310544_Nodejs_Challenges_in_Implementation

Sutara, B., & Gunawan, S. S. (2024). Comparative analysis of REST API performance between Express.js framework and Hapi.js using Apache JMeter. *Jurnal Elektronik Riset dan Teknologi Informasi*, 1(1). Diakses dari https://ejournal.jurnalist.org/index.php/jureti/article/view/3/4

Caldwell, G. (2021). *Agile project management*. Alakai Publishing LLC.
Nandaa, A. (2018). *Beginning API development with Node.js*. Packt Publishing.
Pereira, C. R. (2016). *Building APIs with Node.js*. New York: Apress.
Supardi, I. Y. (2021). *Semua bisa menjadi programmer Javascript & Node.js*. Jakarta: Elex Media Komputindo.
Sutherland, J., & Coplien, J. O. (2019). *A Scrum book*. Pragmatic Bookshelf.