Jurnal Riset Teknik Informatika (JURETI)
Vol. 1, No. 1, Agustus 2024, pp. 19-26

DOI: .. . ./jureti
p-ISSN: XXXX-XXXX
e-ISSN: XXXX-XXXX

COMPARATIVE ANALYSIS OF REST API PERFORMANCE BETWEEN EXPRESS.JS FRAMEWORK AND HAPI.JS USING APACHE JMETER

Beben Sutara [1], Sandy Shultan Gunawan[2],

[1, 2]Informatics, Faculty of Information Technology, Sebelas April University Sumedang, Indonesia
Email: [1]beben@unsap.ac.id, [2]A2.2000103@mhs.stmik-sumedang.ac.id,

(Article received: date; Revision: date; published: date)

Abstract

By utilizing the Application Programming Interface (API) to integrate data and to connect applications that are running on different platforms, allowing them to connect with each other. REST API is one of the implementations of the API that is currently popular among developers. In the selection of technology in the development of the REST API is very important because it can affect performance on the server. The Express.js and Hapi.js frameworks are one of the technologies used to create a REST API using the javascript language. To find out the performance of a framework, a test is needed using the performance testing method to determine the response time and CPU and memory usage. The test results show that Hapi.js has an average response time of 2426.4 ms faster than the Express.js framework which gets an average response time of 3295.8 ms, while for CPU usage the Hapi.js framework is 18.6% and the Express.js framework is 14.3%. For memory usage, the Hapi.js framework is 65.4% more than the Express.js framework 64.1%. These results show that using performance testing methods is effective in measuring framework performance, providing important insights into the efficiency and performance of technologies used for REST API development.

Keywords: *Express.js, Hapi.js, Performance Testing, REST API*

ANALISIS PERBANDINGAN PERFORMA REST API ANTARA FRAMEWORK EXPRESS.JS DENGAN HAPI.JS MENGGUNAKAN APACHE JMETER

Abstrak

Dengan memanfaatkan Application Programming Interface (API) dalam melakukan integrasi data dan untuk menghubungkan aplikasi yang sedang berjalan di berbagai platform yang berbeda, sehingga memungkinkan mereka untuk saling terhubung. REST API merupakan salah satu implementasi dari API yang sedang populer dikalangan para pengembang. Dalam pemilihan teknologi dalam pembangunan REST API sangat penting karena dapat mempengaruhi kinerja pada server. Framework Express.js dan Hapi.js adalah salah satu teknologi yang digunakan untuk membuat sebuah REST API dengan menggunakan bahasa *javascript*. Untuk mengetahui performa dari suatu *framework*, diperlukan suatu pengujian dengan menggunakan metode *performance testing* untuk mengetahui respon time serta penggunaan CPU dan memori. Hasil pengujian menunjukkan bahwa Hapi.js mempunyai waktu respon rata – rata 2426.4 ms lebih cepat dibandingkan *framework* Express.js yang memperoleh waktu respon rata – rata 3295.8 ms, sedangkan untuk penggunaan CPU *framework* Hapi.js 18.6% dan *framework* Express.js 14.3%. Untuk penggunaan memori *framework* Hapi.js 65.4% lebih banyak dari *framework* Express.js 64.1%. Hasil ini menunjukkan bahwa dengan menggunakan metode *performance testing* efektif dalam mengukur kinerja *framework*, memberikan wawasan penting tentang efisiensi dan performa teknologi yang digunakan untuk pengembangan REST API.

Kata kunci: *Express.js, Hapi.js, Performance Testing, REST API*

1. PENDAHULUAN

Pada era digital saat ini, perkembangan suatu teknologi informasi dan komunikasi menjadi semakin pesat, membawa dampak yang signifikan pada berbagai aspek kehidupan masyarakat. Salah satu bidang yang mengalami transformasi besar adalah pengembangan aplikasi. Arsitektur *microservices* dan *nano services* menjadi faktor yang terjadinya sebuah transformasi ini, perkembangan arsitektur ini menjadi solusi untuk menghadapi tantangan pengembangan aplikasi mobile yang memiliki peminat yang luas. Dalam mengembangkan sebuah aplikasi mobile,

melakukan sebuah integrasi data sangat diperlukan untuk mencegah terjadinya duplikasi data antar platform yang berbeda, seperti website dan mobile [1].

Application Programming Interface (API) merupakan sebuah antarmuka yang dapat melakukan integrasi data dan menghubungkan aplikasi yang berjalan di berbagai platform yang berbeda, sehingga memungkinkan mereka untuk saling terhubung. Selain itu, API dapat membantu proses development dengan menyediakan sebuah fungsi-fungsi secara terpisah sehingga developer tidak perlu membuat fungsi yang sama berulang kali [2].

Representational State Transfer (REST) merupakan sebuah gaya arsitektur untuk berkomunikasi antar system dengan memanfaatkan sebuah protokol HTTP dalam proses pertukaran informasi. Pendekatan ini telah menjadi umum dalam industri pengembangan perangkat lunak, di antara berbagai jenis arsitektur API yang ada, REST API mencapai tingkat popularitas yang signifikan. Dalam membangun sebuah REST API banyak bahasa pemrograman dan *framework* yang bisa digunakan. Pemilihan teknologi dalam membangun sebuah REST API ini akan mempengaruhi sebuah kinerja pada server, termasuk waktu response, penggunaan CPU, dan penggunaan memori [3], [4].

Banyak sekali bahasa pemrograman dan *framework* yang dapat digunakan dalam membangun sebuah REST API, namun ada beberapa bahasa pemrograman yang cukup populer dan banyak digunakan. *Javascript* merupakan salah satu bahasa pemrograman yang popular dikalangan para pengembang untuk membangun sebuah REST API. *Javascript* pada awalnya dirancang untuk menghadirkan interaktivitas dan dinamis pada halaman web pada sisi *client*. Namun, dengan adanya kemunculan sebuah Node.js sebagai *runtime* pada *javascript*, memungkinkan *javascript* dapat dijalankan pada sisi *server* [5], [6].

Hapi.js dan Express.js merupakan contoh *framework javascript* yang dapat digunakan dalam pengembangan REST API. Hapi.js merupakan *framework* yang memiliki sebuah pengaruh dan memiliki fleksibilitas dengan tujuan utama untuk menyederhanakan dalam pengembangan sebuah aplikasi web dan API yang *scalable* dan aman. Hapi.js memiliki kepopuleran pada platform Github dengan memiliki 14.4 ribu bintang, kepopuleran ini didukung oleh beberapa website yang di bangun menggunakan Hapi.js contohnya PayDash, colonizers, Niceboard.co, Creamalab, dan masih banyak lagi [7], [8].

Selanjutnya, Express.js adalah sebuah *framework* yang dibuat untuk sebuah *web application*. *Framework* Express.js juga dikenal sebagai *framework* yang ringan karena tidak memerlukan banyak dependensi tambahan dengan menggunakan pola desain yang fleksibel sehingga ideal untuk pengembangan aplikasi web dan API. Express.js memiliki kepopuleran pada platform Github dengan memiliki 62.4k ribu bintang, kepopuleran ini didukung dengan beberapa website yang dibangun menggunakan Express.js contohnya IBM, Paypal, GoDaddy, Storylens, dan masih banyak lagi [9], [10].

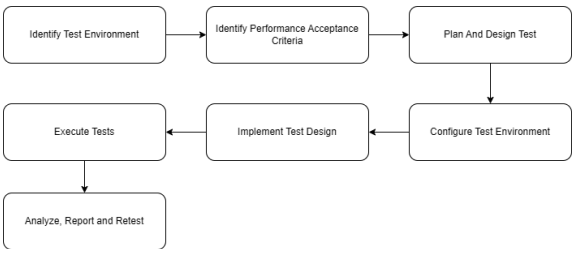
Kedua *framework* tersebut memiliki sebuah perbedaan yang signifikan dalam tingkat popularitas dan komunitas yang aktif. Express.js secara umum lebih populer dan memiliki komunitas yang lebih besar dibandingkan Hapi.js [11]. Dengan adanya sebuah perbedaan ini adanya sebuah kebingungan saat melakukan pengembangan REST API.

Berdasarkan uraian diatas, memilih sebuah teknologi atau *framework* dalam sebuah pengembangan aplikasi sangatlah penting khususnya dalam mengembangkan sebuah REST API karena dapat mempengaruhi kinerja dari sebuah aplikasi. Oleh karena itu penelitian ini bertujuan untuk membandingkan kinerja atau performa dari *framework* Express.js dan *framework* Hapi.js dengan menggunakan metode *performance testing*.

Objek penelitian akan menggunakan data mahasiswa yang akan dibuatkan dan disimpan pada database MySql, data tersebut akan digunakan untuk sample data yang nantinya dipanggil dengan menggunakan *framework* Express dan Hapi.js. tujuannya adalah untuk menghasilkan sebuah *response* yang dapat ditampilkan kepada pengguna serta mengevaluasi kinerja dari kedua *framework* tersebut.

2. METODE PENELITIAN

Metode penelitian yang akan digunakan pada penelitian ini dengan menggunakan metode *performance testing*. *Performance testing* adalah sebuah proses menjalankan pengujian kepada sebuah aplikasi dengan mensimulasikan sebuah *virtual user.* Proses ini diimplementasikan dengan bantuan perangkat lunak yang memungkinkan meniru kondisi penggunaan oleh pengguna nyata [12]. Metode ini bertujuan agar dapat menguji *scalability, availability* dan kinerja baik dari perangkat keras maupun perangkat lunak. *Performance testing* akan dilakukan dalam beberapa tahapan seperti pada gambar 1 [13].



Gambar 1 Tahapan *Performance Testing*

Berikut detail penjelasan tahapan *performance testing* pada gambar 1:

A. *Identify Test Environment*

Pada tahap ini, akan dilakukan identifikasi *environment* yang diperlukan untuk pengujian. *Environment* ini diperlukan untuk memantau setiap permintaan yang akan diberikan kepada masing-masing *endpoint* di setiap *framework.* Adapun komponen *environment* yang akan digunakan meliputi server, *tools* dan *framework* itu sendiri.

B. *Identify Performance Acceptance Criteria*

Pada tahap ini, akan dilakukan identifikasi kriteria pengujian yang disesuaikan dengan kasus penelitian. Pada penelitian ini akan menggunakan data mahasiswa yang berjumlah 1.000 baris data. Seluruh data ini akan dimanipulasi oleh sistem yang akan digunakan. Berdasarkan kasus tersebut, jelas bahwa sistem harus memiliki kecepatan tinggi dalam memproses data, karena akan diakses oleh banyak pengguna yang ingin melihat data tersebut.

C. *Plan and Design Test*

Pada tahap ini, akan dilakukan perencanaan dan perancangan pengujian untuk kedua *framework* dengan mempertimbangkan setiap kemungkinan penggunaan oleh pengguna. Pengujian ini mencakup kegiatan melihat, menambahkan, merubah serta menghapus data.

D. *Configure Test Environment*

Pada tahap ini, konfigurasi *environment* pengujian akan dilakukan sesuai dengan penjelasan pada tahap pertama, sehingga setiap *environment* siap digunakan untuk proses pengujian. Proses konfigurasi ini meliputi penyiapan *server* dan *framework* pada kedua *environment* pengujian.

E. *Implement Test Design*

Pada tahap ini, *environment* yang sudah disiapkan akan diimplementasikan sesuai dengan rencana pengujian yang sudah ditetapkan pada tahap sebelumnya. JMeter yang sudah terpasang pada komputer *client* peneliti, akan dilakukan pengaturan sesuai dengan rencana pengujian.

F. *Execute Tests*

Pada tahap ini, akan dilakukan eksekusi pengujian serta memonitoring setiap hasil pengujian dari kedua *framework* baik Express.js maupun Hapi.js sehingga diperoleh data yang akan digunakan pada tahap selanjutnya.

G. *Analyze, Report and Retest*

Pada tahap ini, akan dilakukan analisa dan perbandingan antara *framework* Express.js dan Hapi.js yang sudah dilakukan pengujian dengan tujuan untuk mengetahui kinerja dari masing-masing *framework* sehingga diperoleh sebuah hasil yang menyatakan *framework* mana yang optimal secara kinerja.

3. HASIL DAN PEMBAHASAN

3.1. *Identify Test Environment*

Pada penelitian ini, server yang akan digunakan dari layanan cloud DigitalOcean dengan dua buah *virtual machine* (VM). Setiap REST API untuk masing-masing *framework* akan ditempatkan pada dua server yang berbeda dengan spesifikasi yang sama. Tujuan dari pengaturan ini adalah untuk memastikan bahwa setiap *framework* dapat menangani semua permintaan secara independen tanpa harus berbagi sumber daya. Hal ini memastikan bahwa performa dari masing-masing *framework* dapat diuji secara adil dan akurat.

Untuk melakukan monitoring dari setiap hasil pengujian, peneliti akan menggunakan perangkat lunak JMeter. JMeter akan menghasilkan data berupa *response time* dan *error rate* dari setiap permintaan ke masing-masing *endpoint*. Selain itu, untuk memonitoring sumber daya, peneliti akan menggunakan sebuah *plugin* data yang ada pada JMeter, yaitu *PerfMon Metrics Collector*, untuk mengetahui penggunaan CPU dan memori saat server menangani permintaan. Selain *server* dan perangkat lunak, untuk penggunaan *framework* akan menggunakan *framework* Express.js versi 4.19 dan *framework* Hapi.js versi 21.3 yang akan dijalankan menggunakan Node.js versi 22.5.0, untuk spesifikasi dari server yang akan digunakan dapat terlihat pada tabel 1.

Tabel 1 Spesifikasi test Environment

Server	
Cloud service	DigitalOcean
CPU	1 Core
SSD	25 GB
Sistem Operasi	Ubuntu Server Version 22.04(LTS) x64
Web Server	Nginx
Database	
DBMS	MySQL
Versi	8.0.37
Tools	
Aplikasi	Apache JMeter
Versi	5.63
Framework	
Framework 1	Express.js v4.19
Framework 2	Hapi.js v21.3

3.2. *Identify Performance Acceptance Criteria*

Adapun kriteria kebutuhan untuk sistem yang sudah ditetapkan pada tabel 2 salah satu *performance objective* pada penelitian ini adalah sebuah *response time*, sehingga untuk melakukan pengujian peneliti menetapkan untuk *response time* tidak melebihi lima detik ketika diakses oleh seribu user untuk kedua *framework*. Penggunaan sumber daya masing-masing *framework* juga perlu dianalisis. Oleh karena itu, tujuan kinerja lainnya adalah memantau penggunaan sumber daya, yaitu CPU dan memori. Dalam penelitian, kriteria penggunaan sumber daya adalah di bawah 75% ketika kedua *framework* menangani banyak permintaan.

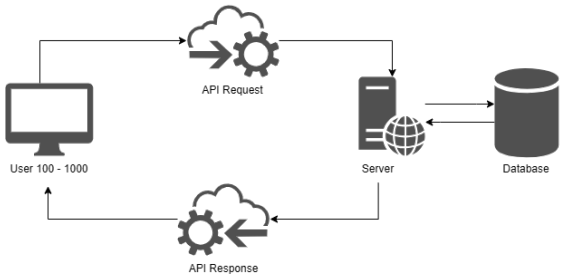
Tabel 2 Kriteria Pengujian

Performance Objective	*Criteria*
Response Time	< 5 detik ketika 1000 pengguna akses secara bersamaan
Penggunaan CPU	< 75%
Penggunaan Memory	< 75%

3.3. Plan and Design Test

Perencanaan pengujian diperlukan untuk kedua *framework* dengan mempertimbangkan setiap kemungkinan penggunaan oleh pengguna, yaitu melihat, menambahkan, mengubah, dan menghapus data. Berdasarkan objek penelitian, perencanaan pengujian ini akan melakukan pengiriman permintaan dari klien dengan menggunakan Jmeter ke setiap endpoint yang telah disediakan oleh *framework* yang telah ditempatkan pada masing-masing server dengan menggunakan metode HTTP (GET, POST, PUT, DELETE).

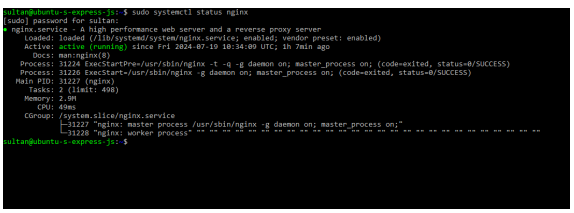
Terlihat pada gambar 2 dijelaskan bahwa alur pengujian yang akan dilakukan secara bersamaan dengan penambahan jumlah pengguna dari 100 hingga 1000 pengguna dengan adanya interval lima menit untuk setiap pengujian.



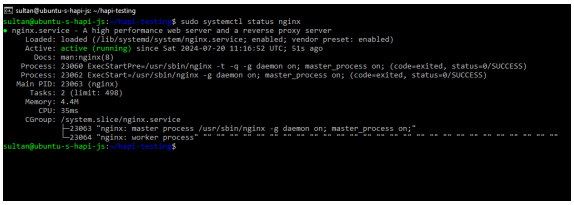
Gambar 2 Alur Pengujian

3.4. Configure Test Environment

Pada tahap ini dilakukan konfigurasi pada *environment* yang sudah diidentifikasi pada tahap pertama sehingga setiap *environment* sudah siap digunakan untuk pengujian. Berikut gambar terkait konfigurasi status Apache di server 1 untuk *framework* Express,js dan server 2 untuk *framework* Hapi.js.



Gambar 3 Apache Server Express.js



Gambar 4 Apache Server Hapi.js

3.5. Implement Test Design

Pada tahap ini, *environment* yang telah disiapkan akan diimplementasikan sesuai dengan rencana pengujian yang telah disiapkan sebelumnya. Jmeter yang telah terpasang pada komputer klien akan diatur sesuai dengan rencana pengujian tersebut. Pembuatan rencana pengujian pada Jmeter akan disesuaikan dengan kedua *framework* dan IP publik dari masing-masing *server.* Pembuatan rencana pengujian pada Jmeter mengacu pada *endpoint* yang akan menangani setiap permintaan, *endpoint* dapat terlihat pada tabel berikut:

Tabel 3 Endpoint API Express.js

HTTP Method	**URL**
GET	/express/users
POST	/express/users
PUT	/express/users/:id
DELETE	/express/users/:id

Tabel 4 Endpoint API Hapi.js

HTTP Method	**URL**
GET	/hapi/users
POST	/hapi/users
PUT	/hapi/users/{id}
DELETE	/hapi/users/{id}

3.6. Execute Tests

Pada tahap ini dilakukan eksekusi pengujian serta memonitoring setiap hasil pengujian dari kedua *framework* baik Express.js maupun Hapi.js sehingga diperoleh data yang akan digunakan pada tahap selanjutnya. Untuk menjalankan rencana pengujian yang sudah disimpan perlu menuliskan *script* pada *command prompt*. *Script* untuk menjalankan rencana pengujian terbagi menjadi tiga perintah yaitu:

- -n –t [test_plant_file] berfungsi untuk mengeksekusi file test pada folder yang sudah ditentukan.
- -l [file_result] berfungsi untuk menghasilkan file csv yang didalmnya terdapat hasil pengujian.
- -e –o [report_folder] berfungsi untuk menyimpan berkas *dashboard* hasil dari

pengujian agar mudah dalam menganalisa hasil pengujian.

3.7. Analyze, Report and Retest

Untuk hasil pengujian *framework* Express.js terlihat pada tabel 5 dengan nilai rata-rata *response time* sebesar 3295.8 ms. Sedangkan untuk rata-rata penggunaan CPU adalah 14.3% dan rata-rata penggunaan memori adalah 64.1%.

Tabel 5 Hasil Pengujian Framework Express.js

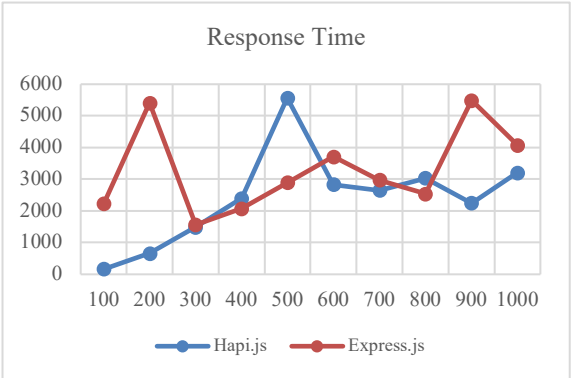
User	Response Time (ms)	CPU Usage	Memory Usage
100	2224	7.3%	60.4%
200	5410	6.0%	61.8%
300	1563	15.0%	62.5%
400	2072	18.8%	63.1%
500	2889	17%	64.8%
600	3709	13.7%	67.7%
700	2979	17.5%	64.0%
800	2540	19.5%	63.1%
900	5494	12.8%	66.7%
1000	4078	15.6%	67.4%
Average	**3295.8**	**14.3%**	**64.1%**

Selanjutnya untuk hasil dari pengujian *framework* Hapi.js dapat terlihat pada tabel 6 dengan nilai rata-rata *response time* sebesar 2426.4 ms. Sedangkan untuk rata-rata penggunaan CPU adalah 18.6% dan rata-rata penggunaan memori adalah 65.4%.

Tabel 6 Hasil Pengujian Framework Hapi.js

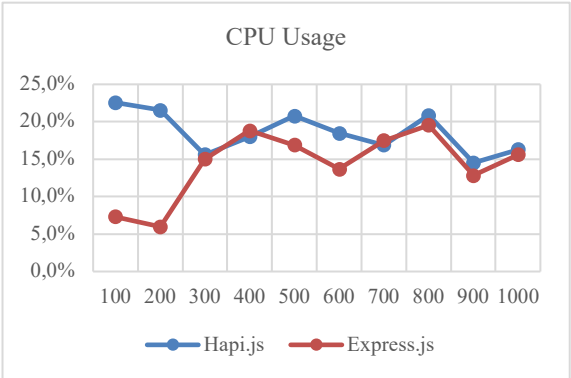
User	Response Time (ms)	CPU Usage	Memory Usage
100	162	22.6%	61.9%
200	662	21.5%	62.7%
300	1493	15.6%	63.3%
400	2398	18.0%	65.9%
500	5568	20.8%	69.6%
600	2838	18.5%	67.0%
700	2654	16.9%	67.8%
800	3032	20.9%	66.8%
900	2246	14.5%	62.5%
1000	3211	16.3%	66.7%
Average	**2426.4**	**18.6%**	**65.4%**

Berdasarkan pengujian yang telah dilakukan maka dapat diperoleh hasil rata-rata *response time*, penggunaan CPU dan penggunaan memori dari masing-masing *framework* baik Express.js maupun Hapi.js. Untuk menganalisanya peneliti akan menguraikan hasil pengujian tersebut dalam beberapa grafik.



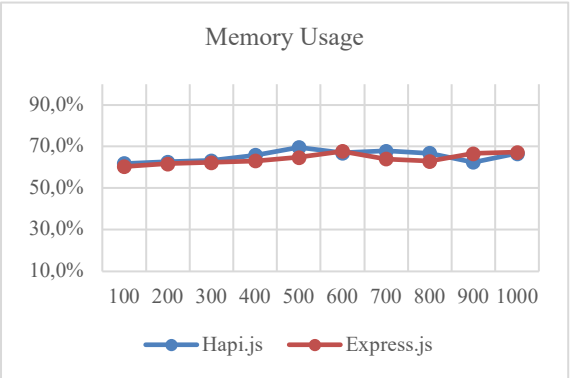
Gambar 5 Grafik Response Time

Berdasarkan pengujian dapat dilihat pada gambar 5 bahwa *framework* Hapi.js secara konsisten memiliki waktu respon yang lebih cepat dibandingkan dengan Express.js. waktu respon rata-rata Hapi.js adalah 2426.4 ms, sedangkan Express.js memiliki waktu respon rata-rata 3595.8 ms. Hal ini menunjukan bahwa Hapi.js lebih responsif dalam menangani permintaan pengguna dibandingkan dengan Express.js.



Gambar 6 Grafik Penggunaan CPU

Berdasarkan gambar 6, terlihat perbedaan rata-rata penggunaan CPU antara kedua framework. Hapi.js consistently menunjukkan penggunaan CPU yang lebih tinggi dibandingkan dengan Express.js, dengan rata-rata 18.6% untuk Hapi.js dan 14.3% untuk Express.js. Penggunaan CPU yang lebih tinggi pada Hapi.js mengindikasikan bahwa framework ini memanfaatkan lebih banyak sumber daya CPU untuk mencapai waktu respon yang lebih cepat. Namun, hal ini juga bisa menunjukkan bahwa Hapi.js mungkin kurang efisien dalam penggunaan CPU dibandingkan dengan Express.js.



Gambar 7 Grafik Penggunaan Memori

Berdasarkan pada gambar 7, memperlihatkan bahwa rata – rata penggunaan memori pada kedua *framework*. Kedua *framework* menunjukkan penggunaan memori yang cukup sebanding. Express.js memiliki rata – rata penggunaan memori sebesar 64.1% sedangkan Hapi.js memiliki rata – rata penggunaan memori sebesar 65.4%. Perbedaan ini tidak signifikan, menunjukan bahwa kedua *framework* memiliki efisiensi yang serupa dalam hal penggunaan memori.

4. DISKUSI

Penelitian ini bertujuan untuk membandingkan kinerja REST API yang dibangun dengan menggunakan *framework* Express.js dan Hapi.js dengan menggunakan metode *performance testing*. Berikut adalah tabel hasil rata – rata dari pengujian yang telah dilakukan:

Tabel 7 Hasil rata-rata Pengujian Kedua Framework

Framework	Response Time (ms)	CPU Usage	Memory Usage
Express.js	3295.8	14.3%	64.1%
Hapi.js	2426.4	18.6%	65.4%

Berdasarkan hasil pengujian yang telah dilakukan, dapat ditarik beberapa kesimpulan mengenai kinerja dari kedua *framework* tersebut:

- Response Time

Hasil dari pengujian menunjukkan bahwa Hapi.js memiliki waktu respon yang lebih cepat dibandingkan dengan Express.js memiliki rata – rata waktu respon 2426.4ms, sementara Express.js memiliki rata – rata 3295.8 ms. Hal ini menunjukan bahwa Hapi.js lebih efisien dalam menangani permintaan dan memberikan respon yang lebih cepat dibandingkan dengan Express.js

- Penggunaan CPU

Penggunaan CPU pada Hapi.js lebih tinggi dibandingkan dengan Express.js. Hapi.js memiliki rata – rata penggunaan 18.6% sedangkan pada Express.js menggunakan rata – rata 14.3% CPU. Ini menunjukkan bahwa Hapi.js memerlukan lebih banyak sumber daya CPU untuk menangani permintaan dibandingkan dengan Express.js.

- Penggunaan Memori

Kedua *framework* menunjukkan penggunaan memori yang relatif tinggi, dengan Hapi.js menggunakan rata – rata 65.4% dan Express.js menggunakan rata – rata 64.1%. Perbedaan ini dalam menggunakan memori antara kedua *framework* ini tidak terlalu signifikan, namun Hapi.js masih memerlukan sedikit lebih banyak memori dibandingkan dengan Express.js.

Dari hasil analisis ini, dapat disimpulkan bahwa Hapi.js memiliki performa yang lebih banyak dalam hal waktu respon dibandingkan dengan Express.js meskipun dengan penggunaan CPU yang lebih tinggi. Sementara itu, penggunaan memori pada kedua *framework* hampir serupa. Oleh karena itu, pemilihan antara Hapi.js dan Express.js dapat disesuaikan dengan kebutuhan spesifik dari aplikasi yang akan dibangun, terutama dalam konteks penggunaan sumber daya CPU dan kebutuhan responsivitas.

5. KESIMPULAN

Penelitian ini berhasil mengevaluasi dan membandingkan performa REST API yang dibangun menggunakan *framework* Express.js dan Hapi.js melalui metode *performance testing*. Hasil penelitian menunjukkan bahwa Hapi.js mampu memberikan waktu respon yang lebih cepat dibandingkan dengan Express.js, meskipun dengan penggunaan CPU yang lebih tinggi. Sementara itu, penggunaan memori antara kedua *framework* tidak menunjukkan perbedaan yang signifikan. Temuan ini memberikan wawasan bahwa pemilihan *framework* harus mempertimbangkan kebutuhan spesifik dari aplikasi, terutama dalam hal responsivitas dan efisiensi penggunaan sumber daya.

Untuk penelitian selanjutnya, disarankan untuk mengeksplorasi *framework* yang lain, untuk mendapatkan perspektif yang lebih luas tentang kinerja API dalam berbagai konteks. Selain itu, untuk menambahkan parameter pengujian, seperti analisis latensi, throughput, dan konsumsi energi, juga akan memberikan wawasan yang lebih mendalam dan memungkinkan evaluasi yang lebih komprehensif terhadap kinerja dan efisiensi setiap *framework*.

DAFTAR PUSTAKA

[1] W. Hadinata and L. Stianingsih, “ANALISIS PERBANDINGAN PERFORMA RESTFUL API ANTARA EXPRESS.JS DENGAN LARAVEL FRAMEWORK DENGAN JMETER,” *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 1, Jan. 2024, doi: 10.23960/jitet.v12i1.3845.

[2] Mangapul Siahaan and R. W. Wijaya, “Performance Comparison Between Laravel and ExpressJs Framework Using Apache JMeter,” *JOURNAL OF INFORMATICS AND TELECOMMUNICATION*

ENGINEERING, vol. 7, no. 2, pp. 545–554, Jan. 2024, doi: 10.31289/jite.v7i2.10571.

[3] M. N. Saiholau, "RANCANG BANGUN BACKEND WEBSITE PEMUNGUTAN SUARA DENGAN MENGGUNAKAN FRAMEWORK EXPRESS.JS," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, Apr. 2024, doi: 10.23960/jitet.v12i2.4261.

[4] L. Mulana, K. Prihandani, A. Rizal, U. Singaperbanga, and K. Abstract, "Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api," *Jurnal Ilmiah Wahana Pendidikan*, vol. 8, no. 16, pp. 316–326, 2022, doi: 10.5281/zenodo.7067707.

[5] I. P. A. E. Pratama, "Pengujian Performansi Lima Back-End JavaScript Framework Menggunakan Metode GET dan POST," *Jurnal Resti (Rekayasa Sistem dan Teknologi Informasi)*, vol. 4, no. 6, pp. 1216–1225, 2020.

[6] E. Nurhayati and A. Agussalim, "Rancang Bangun Back-end API pada Aplikasi Mobile AyamHub Menggunakan Framework Node JS Express," *Jurnal Sistem dan Teknologi Informasi (JUSTIN)*, vol. 11, no. 3, pp. 524–531, Jul. 2023, doi: 10.26418/justin.v11i3.66823.

[7] "express vs hapi," stackshare. Accessed: Jun. 19, 2024. [Online]. Available: https://stackshare.io/stackups/npm-express-vs-npm-hapi

[8] Vaishnavi, "Express vs. Hapi: Which is the Best Node.js Framework for Web Development?," atatus. Accessed: Jun. 19, 2024. [Online]. Available: https://www.atatus.com/blog/express-vs-hapi/

[9] D. Hadi Bachtiar, P. Paniran, and I. M. B. Suksmadana, "Perancangan Back-end Api pada Aplikasi Mobile Fruityfit Menggunakan Framework Express JS," *Mars: Jurnal Teknik Mesin, Industri, Elektro Dan Ilmu Komputer*, vol. 2, no. 3, pp. 107–117, 2024, doi: 10.61132/mars.v2i3.138.

[10] "express," stackshare. Accessed: Aug. 10, 2024. [Online]. Available: https://stackshare.io/npm-express

[11] V. Duong, "The Best Node.js Framework: Koa VS Express VS Hapi [Detailed Comparison]," savvycomsoftware. Accessed: Aug. 10, 2024. [Online]. Available: https://savvycomsoftware.com/blog/express-koa-or-hapi-better-performance-with-the-right-nodejs-framework/

[12] H. Sarojadevi, "Performance Testing: Methodologies and Tools," *Journal of Information Engineering and Applications*, vol. 1, no. 5, pp. 5–13, 2011, [Online]. Available: www.iiste.org

[13] simplilearn, "What Is Performance Testing: Definition, Types, Methodology And More," simplilearn. Accessed: Jun. 20, 2024. [Online]. Available: https://www.simplilearn.com/what-is-performance-testing-article