Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

Performance Evaluation of Backend Frameworks for REST API: A Comparative Study of Spring Boot, Flask, Express.js, Laravel FrankenPHP, and Gin

Aufa Syaihan Azzahidi[1], Bangun Wijayanto*[2], Agus Darmawan[3]

[1,2,3]Informatics, Universitas Jenderal Soedirman, Indonesia

Email: [2]bangun.wijayanto@unsoed.ac.id

Received : Jun 2, 2025; Revised : Jun 27, 2025; Accepted : Jul 12, 2025; Published : Aug 19, 2025

Abstract

One major impact of this development is the shift in application development, particularly in data integration across different platforms. *Web services* have emerged as a solution for system integration and multi-platform application development. One implementation of *Web services* is Representational State Transfer. The choice of programming language and *framework* is also crucial in web application development, directly affecting performance and efficiency. Research on *framework* performance is necessary to sup*port* the development of an Academic Information System. This study will use parameters such as *response time*, *throughput*, and *resource usage*, employing a *performance testing method* modified by the author. The *method* includes problem identification, data collection, *backend* development, performance *testing*, and conclusion. The test results show that Spring Boot outperforms others in all parameters with stable and efficient performance. Gin is suitable for medium-scale data, Flask excels in scalability but lacks stability, Express.js is efficient CPU *usage*, and Laravel with FrankenPHP is *Memory*-efficient. These results serve as a reference for selecting *framework*s according to REST API development needs. This research supports developers in selecting appropriate backend frameworks for high-performance REST API systems.

Keywords : *API, Backend, Framework, K6, Performance Testing.*

1. PENDAHULUAN

Teknologi informasi terus berkembang pesat seiring berjalannya waktu. Perkembangan ini seiring dengan kemajuan teknologi pada aplikasi dan layanan yang berbasis web [1]. Salah satu dampak besar dari perkembangan ini adalah perubahan dalam pengembangan aplikasi. Arsitektur *microservices* dan *nano services* kini me*main*kan peran penting, memberikan solusi atas tantangan dalam pengembangan aplikasi *mobile* dengan banyak pengguna. Di sisi lain, pengembangan aplikasi *mobile* membutuhkan integrasi data antar platform yang berbeda, seperti *website* dan aplikasi *mobile*, guna mencegah terjadinya duplikasi data [2].

Web service hadir sebagai solusi untuk integrasi sistem dan pengembangan aplikasi berbasis *multiplatform*. Teknologi ini berfungsi sebagai penghubung komunikasi antara sistem yang berbeda, memastikan aplikasi beroperasi dalam jaringan yang sama dengan protokol standar yang ditetapkan oleh *Web service* . Dengan demikian, penggunaan *Web service* mampu mengatasi kendala ketergantungan pada situs web konvensional [3].

Salah satu bentuk implementasi dari *Web service* adalah *Representational State Transfer* (REST). REST merupakan sebuah arsitektur perangkat lunak yang menetapkan aturan mengenai cara kerja API. Arsitektur berbasis REST dapat digunakan untuk mendukung komunikasi yang efisien dan dapat diandalkan sesuai dengan kebutuhan skala. Arsitektur ini mudah untuk diterapkan dan dimodifikasi, serta memberikan visibilitas dan probabilitas lintas platform pada seluruh sistem API [4]. REST

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

berfungsi untuk memfasilitasi interaksi antara berbagai sistem dan aplikasi melalui *Application Programming Interface* (API). API adalah sekumpulan bahasa dan format pesan yang digunakan oleh aplikasi untuk berinteraksi dengan sistem operasi atau program kontrol lainnya, yang memungkinkan pertukaran data antar sistem dengan cara yang ter standardisasi dan efisien [5].

Pemilihan bahasa pemrograman yang tepat menjadi aspek krusial dalam pengembangan aplikasi. Berdasarkan data dari GitHub, antara tahun 2014 hingga 2022, bahasa pemrograman yang paling populer adalah Python, JavaScript, TypeScript, Jawa, C#, C++, PHP, Shell, C, dan Go, adapun Obj-c dan Ruby pada popularitas yang sama degan Go [6]. Pada tahun 2023, JavaScript masih menduduki peringkat pertama, diikuti oleh Python dan Typescript [7].

Java adalah bahasa pemrograman yang terus berkembang dan berorientasi pada objek, memungkinkan penggunaannya di berbagai perangkat, termasuk ponsel [8]. Python merupakan bahasa pemrograman dengan *syntax* yang sederhana dan aplikasi yang luas. Selain itu, Python bersifat *open source*, menjadikannya salah satu pilihan terbaik bagi pemula dalam mempelajari pemrograman [9]. JavaScript adalah bahasa *scripting* yang dijalankan di sisi klien, di mana komputer pengguna memproses script secara mandiri. Bahasa ini sering digunakan untuk membuat animasi dan elemen interaktif lainnya pada halaman web [10]. PHP merupakan bahasa berbasis skrip yang dijalankan di dalam web server. Selain itu, PHP juga dikenal sebagai *Hypertext Preprocessor*. Bahasa ini hanya dapat dieksekusi di sisi server, dengan hasil yang dikirimkan ke klien. Proses eksekusi kode PHP dilakukan oleh interpreter di server, yang dikenal sebagai *server-side scripting*, berbeda dengan *Java Virtual Machine* (JVM) yang mengeksekusi program di sisi klien [11]. Go, atau yang juga dikenal sebagai Golang, merupakan bahasa pemrograman *open source* dengan sintak yang menyerupai C dan C++. Dikembangkan oleh tiga ilmuwan komputer dari Google, Robert Griesemer, Ken Thompson, dan Rob Pike. Go awalnya dibuat untuk mengatasi masalah kebocoran memori yang menjadi keterbatasan dalam C++ [12].

Pemilihan *framework* juga menjadi hal yang sangat penting dalam pengembangan aplikasi web karena dapat mempengaruhi performa dan efisiensi. *Framework* bertindak sebagai kerangka kerja yang menyediakan struktur dan komponen dasar, yang bertujuan untuk mempercepat proses pengembangan [13]. Berdasarkan popularitas penggunaan bahasa pemrograman, penulis memilih Spring Boot (Java), Flask (Python), Express.js (JavaScript), Laravel dengan FrankenPHP (PHP), dan Gin (Golang) sebagai *framework* yang akan diuji dalam penelitian ini.

Spring Boot adalah sebuah *framework* Java yang mempermudah pengembang dalam membangun aplikasi berbasis Spring dengan konfigurasi yang sederhana [14]. Flask merupakan *framework* aplikasi web WSGI yang ringan. *Framework* ini dirancang untuk memungkinkan pengembangan aplikasi dimulai dengan cepat dan mudah, sehingga dapat meminimalkan waktu pemuatan [15]. ExpressJS adalah *framework* web yang sangat populer untuk NodeJS, digunakan dalam berbagai produk, termasuk aplikasi web dan RESTful API [16]. Laravel merupakan *framework* web berbasis PHP yang bersifat *open source* dan gratis. *Framework* ini dikembangkan oleh Taylor Otwell dan dirancang untuk membangun aplikasi web dengan menggunakan pola *Model View Controller* (MVC) [17]. Gin adalah *Framework* web HTTP berperforma tinggi yang ditulis dalam Go, dirancang agar sederhana dan mudah digunakan sambil tetap menyediakan fitur yang kuat dan fleksibel untuk membangun aplikasi terdistribusi [18].

Penulis memilih Express.js karena selain pengembangan yang cepat, Express.js memiliki kecepatan *request* 71.87% lebih cepat pada *method* GET dibandingkan dengan NestJS [1]. Spring Boot dipilih karena popularitas di dunia pekerjaan dan sedikitnya penggunaan *dependencies*, sehingga cocok sebagai *framework* dengan performa tinggi [19]. Dibandingkan dengan Django, Flask jauh lebih ringan dan cepat karena Flask dibuat dengan ide menyederhanakan inti *framework*-nya seminimal mungkin, oleh karena itu penulis memilih *framework* ini [20]. Laravel dengan PHP biasa cenderung lebih lambat dibandingkan dengan *framework* lain [13], namun kini sudah tersedia FrankenPHP yang dapat

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

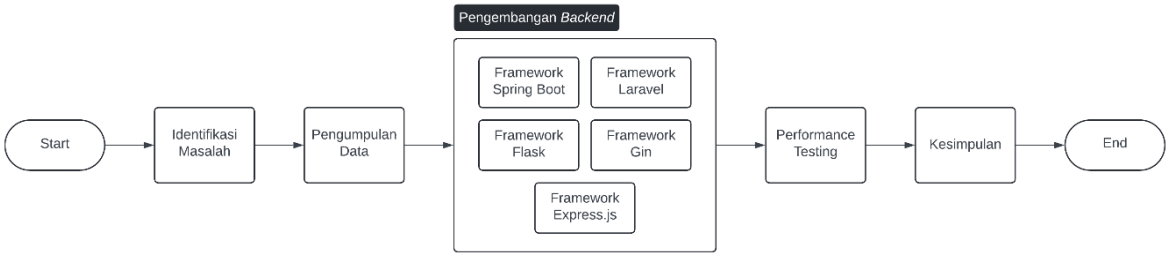
mempercepat kinerja dari Laravel [21]. Gin merupakan *framework* dari Golang untuk pengembangan API di mana *framework* ini memiliki *response time* yang cepat dan penggunaan CPU yang sedikit [22].

Berbeda dari penelitian sebelumnya yang seringkali hanya membandingkan dua atau tiga kerangka kerja (*framework*), atau berfokus pada perbandingan dalam ekosistem bahasa yang sama, penelitian ini menawarkan pendekatan yang lebih komprehensif. Riset ini secara langsung mengevaluasi dan membandingkan performa lima *framework* populer dari lima bahasa pemrograman yang berbeda dalam lingkungan pengujian dan beban kerja yang identik. Inklusi Laravel dengan FrankenPHP menjadi salah satu kebaruan utama, mengingat teknologi ini relatif baru dan berpotensi mengubah tolok ukur performa aplikasi berbasis PHP. Dengan demikian, penelitian ini menyajikan sebuah evaluasi yang lebih luas, berimbang, dan relevan dengan perkembangan teknologi server terkini, memberikan pandangan yang menyeluruh bagi para pengembang dalam memilih arsitektur *backend* yang optimal.

Sistem Informasi Akademik di Universitas Jenderal Soedirman menjadi konteks utama penelitian ini. Sistem ini mengelola berbagai data akademik mahasiswa dan dosen. Sistem ini membutuhkan pengembangan yang efisien dan memiliki performa tinggi untuk mendukung berbagai proses akademik, termasuk pengelolaan nilai, pengisian KRS, serta informasi akademik lainnya. Oleh karena itu, penting untuk mengevaluasi dan membandingkan performa berbagai *framework* dalam mendukung pengembangan sistem ini, guna memastikan efisiensi dan keandalan aplikasi yang dikembangkan.

2. METODE

Penelitian ini akan menggunakan metode *performance testing* yang dimodifikasi oleh penulis. Langkah-langkah dari metodologi penelitian dapat dilihat pada Gambar 1.



Gambar 1. Metode Penelitian

2.1. Identifikasi Masalah

Tahap awal dari penelitian ini adalah menentukan permasalahan utama yang menjadi dasar penelitian, yaitu membandingkan performa Spring Boot, Flask, Express.js, Laravel FrankenPHP, dan Gin sebagai *framework backend* dalam pengembangan REST API. Tahapan ini bertujuan untuk memahami konteks dan urgensi penelitian serta menentukan fokus analisis.

2.2. Pengumpulan Data

Data yang relevan dikumpulkan sebagai dasar untuk pengembangan dan pengujian. Data ini meliputi data tabel yang akan diuji, serta dokumentasi dari kedua *framework* untuk memastikan bahwa kebutuhan pengujian dapat setara, seperti penggunaan *library*.

2.3. Pengembangan *Backend*

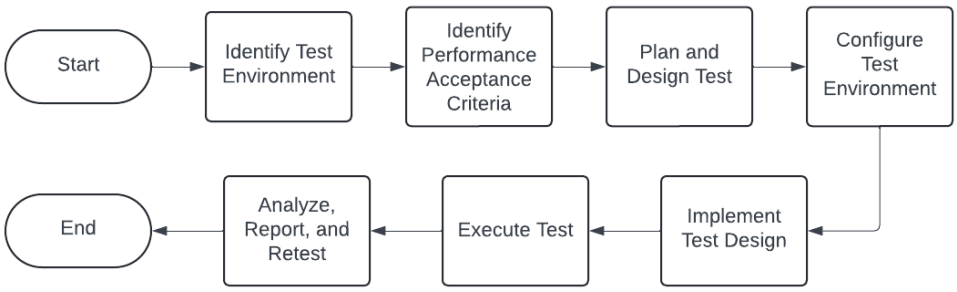
Tahap ini merupakan pengembangan *backend* menggunakan Spring Boot, Flask, Express.js, Laravel FrankenPHP, dan Gin. Hasil dari pengembangan ini adalah fungsi *create*, *read*, *update*, dan *delete* (CRUD), yang diimplementasikan dalam bentuk REST API. Proses pengembangan ini

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

memastikan bahwa setiap *framework* dapat menangani permintaan dengan efisien, serta memberikan dasar yang solid untuk pengujian dan evaluasi lebih lanjut terhadap kinerja masing-masing *framework*.

2.4. *Performance Testing*

Pengujian API yang telah dibuat akan dilaksanakan pada tahap ini.. Jenis performance test yang akan dilakukan adalah *load testing*. API akan menerima sejumlah besar permintaan untuk mengukur performa, dengan fokus parameter *response time*, *resource usage* (CPU *usage* dan *memory usage*), dan *throughput*. Tahapan ini mencakup *Identify Test Environment*, *Identify Performance Acceptance Criteria*, *Plan and Design Test*, *Configure Test Environment*, *Implement Test Design*, *Execute Tests* dan *Analyze, report, and Retest*. Tahapan ini dapat dilihat pada Gambar 2.



Gambar 2. *Flowchart* Metode Performance Testing

2.5. Kesimpulan

Tahap ini merupakan tahap terakhir dari penelitian, di mana penulis menarik kesimpulan berdasarkan hasil analisis perbandingan. Kesimpulan ini memberikan gambaran mengenai keunggulan dan kekurangan dari masing-masing *framework*.

3. HASIL

3.1. Identifikasi Masalah

Pemilihan *backend framework* dalam pengembangan REST API menjadi faktor krusial yang dapat mempengaruhi performa, efisiensi, serta skalabilitas aplikasi. Setiap *framework* memiliki pendekatan dan arsitektur yang berbeda dalam menangani *request*, mengelola sumber daya, serta menangani komunikasi antar sistem. Oleh karena itu, penelitian ini berfokus pada perbandingan performa empat *backend framework*, yaitu Spring Boot, Flask, Express.js, Laravel FrankenPHP, dan Gin, yang masing-masing memiliki karakteristik unik. Dengan adanya perbedaan pada tiap *framework*, penelitian ini bertujuan untuk mengidentifikasi *framework* mana yang memiliki performa terbaik dalam konteks pengembangan sistem informasi akademik, terutama dalam menangani berbagai permintaan pengguna secara bersamaan.

3.2. Pengumpulan Data

Untuk memastikan pengujian yang objektif dan setara, penelitian ini mengumpulkan data dari berbagai sumber, termasuk dokumentasi resmi masing-masing *framework*, studi literatur terkait performa *backend*, serta eksperimen langsung melalui implementasi REST API pada setiap *framework*. Data yang dikumpulkan mencakup konfigurasi server, struktur kode, dependensi yang digunakan, serta teknik optimasi yang direkomendasikan oleh masing-masing *framework*.

Data yang digunakan dalam pengujian adalah tabel Kartu Rencana Studi (KRS) pada Sistem Informasi Akademik Unsoed yang berjumlah 5.547.580 data dengan tambahan kolom id_krs sebagai

P-ISSN: 2723-3863 https://jutif.if.unsoed.ac.id
E-ISSN: 2723-3871 DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

primary key, kolom yang akan digunakan dapat dilihat pada Tabel 1. Pengujian dilakukan dengan berbagai skenario beban kerja untuk mengukur waktu respons, konsumsi CPU, penggunaan memori, serta efisiensi dalam menangani *request* secara bersamaan. Data ini nantinya akan dianalisis untuk mengidentifikasi kekuatan dan kelemahan masing-masing *framework* dalam mendukung pengembangan aplikasi berskala akademik.

Tabel 1. Tabel KRS

Nama Kolom	Tipe Data
id_krs	integer
nim	varchar(30)
kode matakuliah	varchar(30)
matakuliah	varchar(150)
semester	integer
tahunakademik	integer

3.3. Pengembangan *Backend*

Pada tahap ini, pengembangan dari *backend* diusahakan untuk menggunakan *library* seminimal mungkin, serta semirip mungkin. *Method* yang akan digunakan pada pengembangan ini antara lain, GET, GET ID, POST, PUT, dan DELETE. *Endpoint* dari *method* yang digunakan akan sama pada tiap *framework*. Setiap *framework* memiliki struktur atau arsitektur yang berbeda seperti MVC, MVVM, MVP, dan sebagainya. Pada pengembangan *backend* ini dibebaskan dalam pemilihan arsitektur. Meskipun perbedaan arsitektur dapat memengaruhi performa, dampaknya tidak dianggap cukup signifikan untuk menjadi faktor penentu [23].

3.3.1. Spring Boot

Spring Boot adalah sebuah *framework* Java yang mempermudah pengembang dalam membangun aplikasi berbasis Spring dengan konfigurasi yang sederhana. Spring Boot adalah salah satu proyek dari Pivotal, sebuah perusahaan yang mengembangkan *framework* Spring. *Framework* ini menggunakan Groovy, yaitu bahasa pemrograman *scripting* yang dikembangkan di atas *Java Virtual Machine* (JVM), yang dapat diinterpretasikan atau dikompilasi [14].

Spring Boot menggunakan bahasa dasar Java dengan versi JDK 21.0.6. Pembuatan *backend* menggunakan Spring Boot dipermudah dengan *tools* spring initializr yang dapat diakses pada https://start.spring.io/. Pada web tersebut, dapat dilakukan pengaturan terkait project yang akan dibuat. Adapun *dependencies* yang akan digunakan dalam pengembangan yaitu, spring-boot-starter-web, spring-boot-starter-data-jpa, dan postgresql. Pada pengembangan ini dibuat *model*, *repository*, *service*, DTO, *controller*, dan *main* agar dapat berjalan dengan lancar.

3.3.2. Flask

Flask merupakan *framework* aplikasi web WSGI yang ringan. *Framework* ini dirancang untuk memungkinkan pengembangan aplikasi dimulai dengan cepat dan mudah, sehingga dapat meminimalkan waktu pemuatan. Dengan kemampuan tersebut, Flask mampu mendukung pengembangan aplikasi yang kompleks. Flask adalah pembungkus sederhana yang mirip dengan Werkzeug dan Jinja, namun telah berkembang menjadi salah satu *framework* aplikasi web Python yang paling populer [15]. Flask menawarkan berbagai keuntungan, seperti sintaksis yang sederhana dan ringan, sehingga mendukung pengembangan yang cepat. Dukungan komunitas yang luas juga mempermudah pengembang dalam mencari solusi atau bantuan saat menghadapi kendala [24].

Pengembangan Flask dilakukan menggunakan bahasa pemrograman Python. Pada penelitian ini, penulis menggunakan Python versi 3.13.2. Selain itu, terdapat beberapa *library* tambahan yang digunakan dalam pengembangan *backend*, terutama Flask sebagai *framework* web, psycopg2 untuk menghubungkan aplikasi dengan *database* PostgreSQL serta dotenv untuk mengelola *environment variables*. Pada pengembangan ini dibuat *config*, *routes*, dan *main* agar dapat berjalan dengan lancar.

3.3.3. Express.js

ExpressJS adalah *framework* web yang sangat populer untuk NodeJS, digunakan dalam berbagai produk, termasuk aplikasi web dan RESTful API. *Framework* ini memiliki dokumentasi yang lengkap dan mudah dipahami [16]. Express.js dibangun di atas Node.js merupakan kerangka kerja web yang sangat fleksibel dan efisien, ideal untuk membangun API, aplikasi web berskala besar, atau micro*service*. Kemampuannya memanfaatkan *model* berbasis peristiwa Node.js menjadikannya berkinerja tinggi dan efisien dalam menangani operasi asinkron dan koneksi yang banyak, sangat cocok untuk aplikasi web modern. Kesederhanaan dan kemudahannya dalam mendefinisikan rute menjadikannya pilihan utama untuk membangun RESTful API, dengan kemampuan memproses data JSON, formulir, dan query string [25].

Pengembangan *backend* ini menggunakan bahasa dasar JavaScript pada Node.js versi 23.8.0. Pengembangan ini membutuhkan *dependencies* tambahan terutama Express, pg, dan dotenv. Adapun fungsi dari masing-masing *dependencies* yaitu, express untuk pengembangan API, pg untuk koneksi dengan postgreSQL, dan dotenv untuk penggunaan *environment variables*. Express tidak memiliki aturan arsitektur dalam pembuatan API yang membuat developer menjadi fleksibel dalam memilih arsitektur *backend*. Pada pengembangan ini dibuat *config*, dan *controller* agar dapat berjalan dengan lancar.

3.3.4. Laravel

Laravel merupakan *framework* web berbasis PHP yang bersifat *open source* dan gratis. *Framework* ini dikembangkan oleh Taylor Otwell dan dirancang untuk membangun aplikasi web dengan menggunakan pola *Model*-View-*Controller* (MVC) [17]. Laravel memiliki berbagai package yang dapat membantu pengembangan sebuah aplikasi, salah satunya adalah Laravel Octane. Laravel Octane meningkatkan performa aplikasi dengan menjalankannya menggunakan server aplikasi berkinerja tinggi, seperti FrankenPHP, Open Swoole, Swoole, dan RoadRunner. Octane hanya melakukan proses inisialisasi aplikasi sekali, menyimpannya di memori, lalu menangani permintaan dengan kecepatan luar biasa.

Pengembangan aplikasi ataupun API menggunakan Laravel dikenal cukup mudah dan cepat. Bahasa dasar yang digunakan pada Laravel 12 adalah PHP dengan versi 8.3.6. *Library* yang dibutuhkan dalam pengembangan ini cukup sedikit karena adanya fitur built-in pada Laravel yang membantu developer dalam mengembangkan sebuah aplikasi. Laravel Octane merupakan *library* tambahan yang digunakan dalam pengembangan *backend* ini, *library* ini memungkinkan untuk menjalankan aplikasi pada server FrankenPHP yang dikenal mempercepat performa dari PHP. Pada pengembangan ini dibuat *migration*, *model*, *controller*, providers, dan *routes* agar dapat berjalan dengan lancar.

3.3.5. Gin

Gin adalah *framework* web HTTP berperforma tinggi yang ditulis dalam Go, dirancang agar sederhana dan mudah digunakan sambil tetap menyediakan fitur yang kuat dan fleksibel untuk membangun aplikasi terdistribusi. Salah satu fitur utamanya adalah siklus permintaan-respons yang cepat, menjadikannya sangat cocok untuk membangun aplikasi web dengan lalu lintas tinggi [18]. Gin juga menyertakan beberapa *middleware* bawaan untuk tugas-tugas seperti pencatatan (logging), penanganan kesalahan, validasi permintaan, serta dukungan untuk *middleware* kustom. Selain itu,

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

sistem *routing* yang kuat memungkinkan pengembang dengan mudah mendefinisikan dan menangani berbagai metode HTTP dan rute dalam aplikasi mereka [26].

Versi Golang yang digunakan pada pengembangan ini adalah 1.24.0. Adapun *library* tambahan yang diperlukan dalam pengembangan meliputi beberapa paket penting untuk menangani berbagai kebutuhan aplikasi yaitu Gin, godotenv, postgres, dan gorm. Pada pengembangan ini dibuat *config*, *model*, *controller*, *routes*, dan *main* agar dapat berjalan dengan lancar.

3.4. *Performance Testing*

3.4.1. *Identify Test Environment*

Penelitian ini menggunakan *virtual machine* berbasis Linux melalui WSL 2 untuk menjalankan lima *framework*, yaitu Express.js, Spring Boot, Flask, Laravel, dan Gin secara terpisah, dengan pengujian performa menggunakan k6 serta pemantauan *resource* oleh Prometheus dan node_exporter, yang divisualisasikan melalui k6 Web Dashboard dan Grafana.

Tabel 2. Spesifikasi *Test Environment*

Server	
Service	Windows Subsystem for Linux
Processor Count	20
RAM	8 GB
Sistem Operasi	Ubuntu 24.04.2 LTS
Versi	2
Database	
DBMS	PostgreSQL
Versi	16.8
Tools	
Performance	K6 versi 0.57.0
Resource	Prometheus versi 2.53.3
	Grafana versi 11.5.2
	Node_exporter versi 1.9.0
Frameworks	
Framework 1	Express.js versi 4.21.2
Framework 2	Spring Boot versi 3.4.2
Framework 3	Flask versi 3.1.0
Framework 4	Laravel versi 12.0
Framework 5	Gin versi 1.10.0

3.4.2. *Identify Performance Test Criteria*

Tabel 3. Kriteria Pengujian

Performance Objective	Criteria
Response time	< 5 detik untuk *load* data ringan dan < 60 detik untuk *load* data menengah dan berat
Error rate	< 100%
Penggunaan CPU	< 90% saat menangani permintaan besar
Penggunaan Memori	< 90% saat menangani permintaan besar

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

Penelitian ini menetapkan response time dan error rate sebagai performance objective, dengan pengujian GET pada jumlah data bertingkat hingga 1.000.000 untuk mengukur performa tiap framework, sambil memastikan waktu respons tidak melebihi lima detik dan konsumsi resource tidak mencapai 100% serta framework yang melebihi batas error rate tidak dilanjutkan ke tahap pengujian berikutnya.

3.4.3. *Plan and Design Test*

Rencana pengujian akan dilakukan dengan mengirimkan permintaan dari *client* menggunakan k6 ke setiap *Endpoint* API yang telah disediakan oleh masing-masing *framework*. Pengujian ini dijalankan pada server yang telah dikonfigurasi, dengan metode HTTP seperti GET, POST, PUT, dan DELETE untuk mengevaluasi performa setiap *framework*. Pada pengujian GET akan dilakukan pengujian pada pengambilan 100, 1000, 10000, 100000, 500000, dan 1000000 data yang kemudian akan dilakukan *ranking framework* berdasarkan data yang diambil. Selain itu, *user* yang akan membuat *request* berjumlah 20 secara konstan selama 10 menit.

3.4.4. *Configure Test Environment*

Pengaturan lingkungan yang optimal sangat penting agar pengujian berjalan adil dan hasilnya akurat. Setiap *framework* dikonfigurasi untuk berjalan dalam mode *production*, seperti penggunaan PM2 pada Express.js, Waitress pada Flask, serta *build* dan systemd untuk Spring Boot dan Gin. Laravel menggunakan Laravel Octane untuk meningkatkan performa dengan *event-loop*. PostgreSQL juga telah dipastikan aktif dan dapat diakses oleh semua *framework*. Dengan lingkungan yang terkontrol ini, setiap *framework* diuji dalam kondisi terbaik untuk memastikan hasil perbandingan performa yang valid.

3.4.5. *Implement Test Design*

Pembuatan rencana pengujian pada k6 dilakukan dengan skenario yang sama untuk seluruh *framework*, dengan perbedaan hanya pada *port* masing-masing *framework*. Perancangan ini mengacu pada *Endpoint* API yang bertugas menangani setiap permintaan. Daftar lengkap *Endpoint* API yang digunakan dapat dilihat pada Tabel 4.

Tabel 4. Daftar *Endpoints*

Endpoint	HTTP *Method*
/krs	GET
/krs/:id	GET (id)
/krs	POST
/krs/:id	PUT

3.4.6. *Execute Test*

Tabel 5. Perintah Menjalankan *Tools*

Tools	Command
Node_exporter	node_exporter-1.9.0.linux-amd64/node_exporter
Prometheus	prometheus.exe
K6	K6_WEB_DASHBOARD=true k6 run src/script.js

Pada tahap ini, dilakukan eksekusi pengujian serta pemantauan dan dokumentasi hasil pengujian dari seluruh framework agar data yang diperoleh dapat digunakan pada tahap selanjutnya. Untuk menjalankan pengujian menggunakan k6 dan Prometheus, diperlukan perintah pada PowerShell dan WSL. Hasil pengujian dari k6 akan ditampilkan secara real-time melalui k6 Web Dashboard, sedangkan data resource usage dikumpulkan oleh node_exporter, kemudian diambil oleh Prometheus dan

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

divisualisasikan menggunakan Grafana. Perintah yang digunakan untuk menjalankan masing-masing alat dapat dilihat pada Tabel 5.

3.4.7. *Analyze, report, and Retest*

Hasil dari pengujian akan dibagi menjadi 5 bagian *method* yaitu, GET, GET ID, POST, PUT, dan DELETE. Pada *method* GET, akan dibagi menjadi 6 bagian, 100, 1.000, 10.000, 100.000, 500.000, dan 1.000.000 pengambilan data yang akan dilihat dan ditentukan *framework* yang optimal berdasarkan *load* data yang diambil. Seluruh pengujian dilakukan dengan pengaturan yang sama yaitu, 20 *virtual user*s dan dijalankan selama 10 menit.

Pada Tabel 6 menunjukkan pengujian pengambilan 100 data, Spring Boot unggul dengan *throughput* terbanyak dengan total *request* 4.6 juta serta *response time* tercepat yaitu 2 ms, namun *framework* ini membutuhkan CPU dan *memory* yang cukup tinggi.

Tabel 6. Hasil Pengujian *Method* GET 100 Data

No	*Framework*	*Response time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Spring Boot	2 ms	0%	7,690 *req*/s	54.53%	644.25 MB
2	Express.js	4 ms	0%	4,680 *req*/s	6.96%	62.91 MB
3	Laravel FrankenPHP	6 ms	0%	3,200 *req*/s	70.41%	2.10 MB
4	Gin	6 ms	0%	3,020 *req*/s	56.65%	60.82 MB
5	Flask	57 ms	0%	347.62 *req*/s	20.19%	23.07 MB

Hasil pengujian terhadap 1.000 data dapat dilihat pada Tabel 7. Hasil ini menunjukkan perbedaan performa yang signifikan. Gin menempati posisi teratas dengan *throughput* tertinggi dengan jumlah 813.600 *request*, serta waktu respons tercepat, yaitu 14 ms. Meskipun konsumsi CPU dan *memory* lebih tinggi dibandingkan Express.js, Gin tetap menunjukkan performa yang stabil.

Tabel 7. Hasil Pengujian *Method* GET 1.000 Data

No	*Framework*	*Response time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Gin	14 ms	0%	1,350 *req*/s	54.45%	100.66 MB
2	Spring Boot	16 ms	0%	1,230 *req*/s	74.31%	1332.70 MB
3	Express.js	17 ms	0%	1,160 *req*/s	6.53%	23.07 MB
4	Laravel FrankenPHP	27 ms	0%	733.88 *req*/s	83.75%	48.23 MB
5	Flask	61 ms	0%	322.69 *req*/s	18.35%	32.51 MB

Tabel 8. Hasil Pengujian *Method* GET 10.000 Data

No	*Framework*	*Response time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Gin	84 ms	0 %	236.99 *req*/s	68.21%	193.99 MB
2	Express.js	174 ms	0 %	144.08 *req*/s	8.37%	300.94 MB
3	Laravel FrankenPHP	239 ms	0 %	83.52 *req*/s	88.14%	356.27 MB
4	Flask	336 ms	0 %	59.33 *req*/s	10.31%	23.07 MB
5	Spring Boot	341 ms	0 %	58.51 *req*/s	73.31%	2082.52 MB

Tabel 8 merupakan hasil pengujian terhadap pengambilan 10.000 data yang menunjukkan bahwa Gin menempati posisi teratas dengan total 142.200 request, response time tercepat sebesar 84 ms, dan

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

throughput tertinggi yaitu 236.99 request per detik. Meskipun konsumsi CPU dan memory lebih tinggi dibandingkan Express.js, Gin tetap memberikan performa yang konsisten dan stabil.

Hasil pengujian pada Tabel 9 menunjukkan bahwa Gin tetap menempati peringkat pertama dengan jumlah *request* tertinggi yaitu 16.200, *response time* tercepat sebesar 742 ms, dan *throughput* paling tinggi mencapai 26.93 *request* per detik. Meskipun konsumsi CPU dan *memory* cukup tinggi, yaitu 78.75% dan 1127.43 MB, Gin tetap menunjukkan performa yang paling stabil dan mampu menangani beban besar secara konsisten.

Tabel 9. Hasil Pengujian *Method* GET 100.000 Data

No	*Framework*	*Response time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Gin	742 ms	0%	26.93 *req*/s	78.75%	1127.43 MB
2	Express.js	1 s	0%	11.35 *req*/s	9.85%	938.60 MB
3	Flask	2 s	0%	7.82 *req*/s	7.28%	273.68 MB
4	Laravel FrankenPHP	2 s	100%	9.07 *req*/s	89.42%	2876.93 MB
5	Spring Boot	48 s	99%	0.40 *req*/s	55.96%	2273.91 MB

Hasil pengujian terhadap 500.000 data kembali menempatkan Gin di peringkat pertama dengan performa paling seimbang. Gin berhasil memproses 3.300 *request* dengan *response time* rata-rata 3 detik dan *error rate* yang sangat kecil, yaitu hanya 0.03%.

Tabel 10. Hasil Pengujian *Method* GET 500.000 Data

No	*Framework*	*Response time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Gin	3 s	0.03%	5.45 *req*/s	81.75%	4666.83 MB
2	Flask	13 s	0%	1.47 *req*/s	7.44%	956.93 MB
3	Express.js	8 s	80%	2.39 *req*/s	8.30%	2555.47 MB
4	Spring Boot	59 s	100%	0.32 *req*/s	8.33%	2182.26 MB

Hasil pengujian terhadap pengambilan 1.000.000 data menunjukkan performa *framework* yang sangat menurun drastis dibandingkan pengambilan data yang lebih sedikit. Pada pengujian ini, Flask menempati posisi pertama walaupun *response time* sangat tinggi, yaitu 29 detik. Hal ini disebabkan karena Flask menjadi satu-satunya *framework* yang berhasil menyelesaikan *request* tanpa *error*, atau dengan *error rate* 0 %, yang sangat krusial dalam skenario berskala besar.

Tabel 11. Hasil Pengujian *Method* GET 1.000.000 Data

No	*Framework*	*Response time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Flask	29 s	0%	0.66 *req*/s	7.70%	1955.48 MB
2	Gin	1 s	100%	1.70 *req*/s	52.65%	4827.41 MB
3	Express.js	2 s	100%	7.83 *req*/s	1.71%	2467.52 MB

Pada pengambilan data ini hanya Flask yang dapat melanjutkan ke jumlah data berikutnya, sehingga penulis melakukan pengujian hingga Flask menyentuh *error rate* 100%. Hasil dari pengujian dapat dilihat pada Tabel 12. Pada pengujian 2.000.000 data, Flask hanya menangani 204 *request* dengan 93% *error rate* dan *response time* 29 detik, menunjukkan kesulitan dalam menangani beban besar. Konsumsi CPU tetap rendah, yaitu 6.83%, dan *memory* meningkat hingga 3.881 MB.

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

Tabel 12. Hasil Pengujian Flask *Method* GET 2.000.000 Dan 3.000.000 Data

Jumlah Data	*Response time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
2 juta	29 s	93%	0.32 *req*/s	6.83%	3881.42 MB
3 juta	5 ms	100%	2,060 *req*/s	2.76%	2837.71 MB

Pengujian GET ID dilakukan pengambilan data dengan id_krs 1 pada seluruh *framework*. Hasil pengujian pada Tabel 13 menunjukkan bahwa Spring Boot unggul secara keseluruhan dengan *throughput* tertinggi di 16,576.24 *requests*/s dengan total *request* hingga 9,945,805 dan *response time* tercepat yaitu 1.05 ms. Namun Spring Boot memerlukan penggunaan CPU dan *memory* yang cukup tinggi dibandingkan dengan *framework* lainnya.

Tabel 13. Hasil Pengujian *Method* GET ID

No	*Framework*	*Response Time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Spring Boot	1.05 ms	0%	16,576.24 *req*/s	30.06%	312.69 MB
2	Express.js	2.00 ms	0%	7,160.00 *req*/s	7.45%	126.88 MB
3	Laravel FrankenPHP	3.81 ms	0%	5,074.76 *req*/s	69.22%	6.29 MB
4	Gin	4.16 ms	0%	4,691.82 *req*/s	63.90%	56.62 MB
5	Flask	55.83 ms	0%	357.59 *req*/s	21.59%	28.31 MB

Pengujian pada *method* POST, seluruh data yang dikirimkan sama dengan data yang dikirim adalah nim, kode matakuliah, matakuliah, semester, dan tahunakademik. Pada Tabel 14, Spring Boot menjadi yang terbaik dengan *throughput* tertinggi dengan total *request* 3,195,346 dan *response time* tercepat. *Framework* ini membutuhkan *memory* yang lebih banyak dibandingkan dengan *framework* lainnya demi meningkatkan performanya.

Tabel 14. Hasil Pengujian *Method* POST

No	*Framework*	*Response Time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Spring Boot	3.69 ms	0%	5,325.53 *req*/s	15.46%	251.66 MB
2	Express.js	6.20 ms	0%	3,186.73 *req*/s	5.00%	95.42 MB
3	Laravel FrankenPHP	6.59 ms	0%	2,986.53 *req*/s	41.75%	4.19 MB
4	Gin	8.07 ms	0%	2,441.03 *req*/s	50.80%	41.94 MB
5	Flask	58.77 ms	0%	339.38 *req*/s	18.44%	19.92 MB

Data yang diubah pada pengujian *method* PUT adalah nim, kode matakuliah, matakuliah, semester, dan tahunakademik. Seluruh data yang diubah sama pada setiap *framework* dengan tiap *virtual user* mengakses id_krs yang berbeda pada *range* tertentu agar menghindari terjadinya tabrakan. Hasil pengujian *method* PUT pada Tabel 15 menunjukkan bahwa Spring Boot menempati posisi teratas dengan performa terbaik secara keseluruhan. *Framework* ini menghasilkan *throughput* tertinggi yaitu

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

3.903,75 *request* per detik dengan total *request* 2,342,270 dan *response time* sangat rendah sebesar 5.05 ms. Walaupun konsumsi *memory* cukup tinggi, Spring Boot menunjukkan efisiensi tinggi dalam hal kecepatan dan volume pemrosesan.

Tabel 15. Hasil Pengujian *Method* PUT

No	*Framework*	*Response Time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Spring Boot	5.05 ms	0%	3,903.75 *req*/s	16.58%	366.37 MB
2	Express.js	7.79 ms	0%	2,542.72 *req*/s	4.49%	84.93 MB
3	Laravel FrankenPHP	8.45 ms	0%	2,331.57 *req*/s	46.13%	6.29 MB
4	Gin	10.63 ms	0%	1,854.19 *req*/s	52.95%	73.40 MB
5	Flask	61.35 ms	0%	324.96 *req*/s	17.53%	14.68 MB

Tabel 16. Hasil Pengujian *Method* DELETE

No	*Framework*	*Response Time* (avg.)	*Error rate*	*Throughput*	CPU *Usage* (avg.)	*Memory Usage* (avg.)
1	Spring Boot	5.05 ms	0%	3,903.75 *req*/s	16.58%	366.37 MB
2	Express.js	7.79 ms	0%	2,542.72 *req*/s	4.49%	84.93 MB
3	Laravel FrankenPHP	8.45 ms	0%	2,331.57 *req*/s	46.13%	6.29 MB
4	Gin	10.63 ms	0%	1,854.19 *req*/s	52.95%	73.40 MB
5	Flask	61.35 ms	0%	324.96 *req*/s	17.53%	14.68 MB

Pengujian method ini mengambil id_krs secara acak pada range tertentu yang diakses oleh virtual user, hal ini dilakukan agar menghindari terjadinya tabrakan antar virtual user. Berdasarkan hasil pengujian method DELETE pada Tabel 16, Spring Boot menunjukkan performa terbaik dengan throughput tertinggi dan response time tercepat. Meskipun penggunaan memory lebih besar dibandingkan framework lain, efisiensi eksekusinya tetap unggul.

4. DISKUSI

Hasil pengujian performa kelima *framework backend* menunjukkan adanya pertukaran yang jelas antara kecepatan pemrosesan untuk operasi sederhana dan kemampuan menangani beban data dalam jumlah besar. Tidak ada satu *framework* pun yang unggul secara absolut di semua skenario pengujian, yang menggarisbawahi pentingnya pemilihan teknologi berdasarkan kasus penggunaan spesifik.

Spring Boot secara konsisten mendominasi pengujian untuk operasi transaksional tunggal, seperti GET (ID), POST, PUT, dan DELETE. Dengan throughput mencapai 16.576 req/s pada metode GET ID dan *response time* rata-rata di kisaran 1-5 ms untuk operasi CRUD, Spring Boot membuktikan efisiensi *Java Virtual Machine* (JVM) dan *Just-In-Time* (JIT) *compiler* dalam menangani permintaan yang terdefinisi dengan baik dan berulang. Namun, keunggulan ini dibayar dengan konsumsi memori yang sangat tinggi, yang terbukti menjadi kelemahan fatalnya. Pada pengujian pengambilan 100.000 data,

P-ISSN: 2723-3863 https://jutif.if.unsoed.ac.id
E-ISSN: 2723-3871 DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

Spring Boot mengalami *error rate* 99% dan server mengalami *crash*, menunjukkan ketidakmampuannya menangani agregasi data besar tanpa optimasi lebih lanjut.

Sebaliknya, Gin menunjukkan superioritasnya dalam skenario pengambilan data skala menengah. Sebagai bahasa *compiled*, Go memungkinkan Gin mencapai *throughput* dan *response time* yang sangat baik. Walaupun unggul, performa Gin mulai menurun dan menunjukkan instabilitas saat beban data mendekati 500.000 , dan akhirnya gagal total pada 1.000.000 data karena menyebabkan server kehabisan sumber daya dan *restart* berulang kali.

Fenomena menarik ditunjukkan oleh Flask. Meskipun secara konsisten menjadi yang paling lambat pada hampir semua pengujian CRUD , Flask menjadi satu-satunya *framework* yang berhasil menyelesaikan pengujian 1.000.000 data tanpa *error rate* 100%. Sifatnya yang sangat ringan membuatnya memiliki *overhead* minimal, yang secara paradoks memungkinkannya bertahan lebih lama di bawah tekanan memori ekstrem yang menyebabkan framework lain gagal. Namun, dengan *response time* mencapai 29 detik dan stabilitas yang buruk, Flask lebih cocok untuk skenario yang tidak menuntut performa tinggi atau sebagai "pilihan terakhir" untuk skalabilitas data mentah.

Express.js dan Laravel FrankenPHP tampil sebagai juara efisiensi sumber daya. Express.js secara konsisten menunjukkan penggunaan CPU paling rendah di hampir semua skenario, menjadikannya pilihan solid untuk aplikasi yang berjalan di lingkungan dengan daya komputasi terbatas. Sementara itu Laravel FrankenPHP sangat unggul dalam efisiensi memori, terutama pada operasi POST dan PUT, membuktikan bahwa inovasi seperti FrankenPHP berhasil mengatasi kelemahan historis PHP dalam hal manajemen memori dan kecepatan. Keduanya merupakan pilihan yang sangat layak untuk aplikasi skala kecil hingga menengah di mana efisiensi sumber daya menjadi prioritas utama.

5. KESIMPULAN

Berdasarkan analisis hasil pengujian yang telah dilakukan terhadap Spring Boot, Flask, Express.js, Laravel FrankenPHP, dan Gin dalam menangani lima metode HTTP utama, dapat disimpulkan bahwa:

1. Pengujian dilakukan pada lima *framework backend*, yaitu Express.js, Flask, Spring Boot, Gin, dan Laravel FrankenPHP, menggunakan metode HTTP GET, GET ID, POST, PUT, dan DELETE. Setiap metode diuji dengan pendekatan beban 20 *virtual user* selama 10 menit, serta parameter performa yang diamati meliputi *response time*, *throughput*, dan *resource usage* (CPU dan *Memory*).
2. Spring Boot menunjukkan performa paling konsisten dan unggul pada hampir seluruh metode pengujian, yaitu GET dengan 100 data, GET ID, POST, PUT, dan DELETE. Spring Boot memiliki *response time* tercepat, *throughput* tertinggi, dan tidak menghasilkan *error* sama sekali. Meskipun penggunaan *memory* cukup besar, performanya sangat stabil dan cocok untuk pengembangan sistem skala besar dan enterprise.
3. Gin menjadi *framework* yang unggul pada pengujian GET dengan jumlah data sedang pada 1.000–500.000 data. *Response time* dan *throughput* pada Gin relatif sangat baik di skala tersebut, menjadikannya pilihan tepat untuk sistem yang memproses data dalam jumlah menengah. Namun, konsumsi CPU cukup tinggi dan kurang efisien untuk data berskala besar.
4. Flask memiliki keunggulan dalam skalabilitas data karena berhasil menangani GET dengan jumlah data hingga 2 juta dan 3 juta, yang tidak mampu dijalankan oleh *framework* lain. Namun, *response time* mencapai 29 detik dan *error rate* mencapai 93–100%. Meskipun skalabel, Flask tidak stabil untuk kebutuhan performa tinggi. Selain itu, stabilitas dari Flask kurang baik karena terdapat banyak fluktuasi dan lonjakan selama pengujian. Hal ini menunjukkan Flask lebih cocok untuk prototipe atau pengujian internal, bukan sistem produksi dengan *traffic* tinggi.

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

5. Express.js dan Laravel FrankenPHP memperlihatkan performa yang solid dalam penggunaan *resource*. Express.js unggul dalam efisiensi CPU, sementara Laravel FrankenPHP memiliki penggunaan *memory* paling rendah di beberapa skenario. Meskipun tidak selalu menempati posisi teratas dalam hal performa mentah, keduanya layak digunakan untuk aplikasi ringan hingga menengah yang mengutamakan efisiensi *resource*.

Secara keseluruhan, Spring Boot adalah *framework* terbaik dari segi performa umum, dengan kestabilan, kecepatan, dan efisiensi yang sangat seimbang. Gin menjadi pilihan terbaik untuk data menengah, sementara Flask unggul dalam skalabilitas mentah. Express.js dan FrankenPHP cocok digunakan pada sistem dengan kebutuhan *resource* minimal, namun tetap memiliki performa kompetitif. Hasil penelitian ini memberikan acuan bagi pengembang dalam memilih *framework* yang sesuai dengan kebutuhan dan tujuan pengembangan REST API. Untuk penelitian selanjutnya sebaiknya menambahkan pengujian keamanan, pengujian performa secara *real-time*, dan integrasi dengan *framework frontend*.

DAFTAR PUSTAKA

[1] I. P. A. E. Pratama, "Pengujian Performansi Lima Back-End JavaScript *Framework* Menggunakan Metode GET dan POST," *Jurnal RESTI (Rekayas a Sistem dan T eknol ogi Informasi)*, vol. 4, no. 6, pp. 1216–1225, 2020, Accessed: Dec. 04, 2024. [Online]. Available: http://jurnal.iaii.or.id

[2] 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, pp. 531–540, Jan. 2024, doi: 10.23960/jitet.v12i1.3845.

[3] S. A. Achsan and Y. A. Susetyo, "Penerapan RESTful *Web service* Dengan *Framework* Spring Pada Sistem Pengelolaan Aset Ruang," *Jurnal Teknik Informatika (JUTIF)*, vol. 3, no. 2, pp. 395–303, Apr. 2022, doi: 10.20884/1.jutif.2022.3.2.213.

[4] T. Purwanto, "Analisa Perbandingan Kinerja Rest Api Dengan *Framework* Flask, Laravel, Dan Express Js," *Scientica Sacra:Jurnal Sains, Teknologi dan Masyarakat*, vol. 3, no. 4, pp. 49–55, Dec. 2023, Accessed: Dec. 04, 2024. [Online]. Available: http://pijarpemikiran.com/index.php/Scientia

[5] M. K. Naufal, F. Affrianto, and A. B. Cahyono, "Implementasi REST API Untuk Fitur Rencana Strategis Program Pada SIMPEDA," *Automata*, vol. 3, no. 2, Oct. 2022.

[6] Github Staff, "Github News Insight." Accessed: Mar. 04, 2025. [Online]. Available: https://github.blog/news-insights/octoverse/octoverse-2024/

[7] T. Indriyani *et al.*, *Bahasa Pemrograman Populer*. PT. Sonpedia Publishing Indonesia, 2024. Accessed: Dec. 04, 2024. [Online]. Available: https://books.google.co.id/books?id=SlvwEAAAQBAJ&lpg=PP1&pg=PA3#v=onepage&q&f=false

[8] N. R. Sari, A. O. Sari, and E. Zuraidah, "Sistem Informasi Pengolahan Nilai Siswa di SD Al-Hidayah Tangerang," *Jurnal PROSISKO*, vol. 8, no. 1, pp. 68–74, Mar. 2021.

[9] A. Ashril Rizal, L. Puji Indra Kharisma, and Fahrurrozi, "Peningkatan Efektivitas Programming dengan Pelatihan Python for Data Science bagi Komunitas Programming Pondok Pesantren Nahdlatul Wathan Anjani," *Jurnal WIDYA LAKSMI*, vol. 1, no. 1, pp. 13–19, Jan. 2021, doi: 0000000000.

[10] F. Sinlae, I. Maulana, F. Setiyansyah, and M. Ihsan, "Pengenalan Pemrograman Web: Pembuatan Aplikasi Web Sederhana Dengan PHP dan MYSQL," *Jurnal Siber Multi Disiplin (JSMD)*, vol. 2, no. 2, pp. 68–82, Jul. 2024, doi: https://doi.org/10.38035/jsmd.v2i2.

[11] T. Maulana, Firdaus, and Guslendra, "Perancangan Sistem Informasi Pembokingan Dan Keuangan Berbasis Web Pada Pict Story Wedding Fotografer Dengan Menggunakan Bahasa Pemrograman PHP Dan *Database* MySQL," *Jurnal Sains Informatika Terapan (JSIT)*, vol. 3, no. 1, pp. 20–25, Feb. 2024, [Online]. Available: https://rcf-indonesia.org/home/

[12] R. Annisa, R. A. Ananda, and W. E. Sulistiono, "Implementasi Golang Clean Architecture Pada Perancangan *Backend* Point Of Sales *Website*," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, pp. 1518–1523, Apr. 2024, doi: 10.23960/jitet.v12i2.4668.

Vol. 6, No. 4, Agustus 2025, Page. 2405-2419
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.4.4811

[13] Z. Fahrus, K. Umam, L. Hakim, J. Adi Prasetyo, and R. Ema Febrita, "Perbandingan Performa *Framework* Laravel dengan ExpressJS Pada Pengembangan Aplikasi Homestay Kosasih," *JIKOM: Jurnal Informatika dan Komputer*, vol. 15, no. 1, p. 1, Nov. 2024.

[14] A. Fauzi, E. Harli, and T. H. Kusmanto, "Pembelajaran Rest *Web service* Dengan *Framework* Springboot," *JAM-TEKNO (Jurnal Pengabdian Kepada Masyarakat TEKNO)*, vol. 2, no. 1, pp. 13–19, Jun. 2021, Accessed: Dec. 05, 2024. [Online]. Available: http://jurnal.iaii.or.id/index.php/JAMTEKNO

[15] B. B. Santoso and P. O. N. Saian, "Implementasi Flask *Framework* pada Development Modul Re*port*ing Aplikasi Sistem Informasi Helpdesk di PT.XYZ," *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, vol. 7, no. 2, pp. 217–226, Apr. 2023, doi: 10.35870/jtik.v7i2.718.

[16] U. N. Aprilyah, "Implementasi Deteksi Similaritas Kode pada Sistem Praktikum Pemrograman Web Berbasis Unit *Testing* JavaScript," Thesis, Universitas Hasanuddin, Makassar, 2020.

[17] D. Purnama Sari, R. Wijanarko, and J. X. Menoreh Tengah, "Implementasi *Framework* Laravel pada Sistem Informasi Penyewaan Kamera (Studi Kasus Di Rumah Kamera Semarang)," *INFORMATIKA DAN RPL*, vol. 2, no. 1, pp. 32–36, 2020.

[18] S. A. Aklani and J. A. Yang, "Performance Analysis Between Interpreted Language-Based (Laravel) And Compiled Language-Based (Gin) Web *Framework*s," *Computer Based Information System Journal*, vol. 11, no. 01, pp. 12–16, 2023, [Online]. Available: http://ejournal.upbatam.ac.id/index.php/cbishttp://ejournal.upbatam.ac.id/index.php/cbis

[19] R. Yulianto, Mardiana, R. A. Pradipta, and G. F. Nama, "Performance Comparison Analysis Of Spring Boot And Laravel *Framework*s Using API *Web service* ," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, pp. 1145–1153, Apr. 2024, doi: 10.23960/jitet.v12i2.4141.

[20] M. G. L. Putra and M. I. A. Putera, "Analisis Perbandingan Metode SOAP Dan REST Yang Digunakan Pada *Framework* Flask Untuk Membangun *Web service* ," *Jurnal Teknologi Informasi dan Komunikasi*, vol. 14, no. 2, Jun. 2019.

[21] Kévin Dunglas, "FrankenPHP: Modern App Server for PHP." Accessed: Mar. 04, 2025. [Online]. Available: https://frankenphp.dev/docs/

[22] Suwarno and A. P. Yulandi, "Analisis Performa *Backend Framework*: Studi Komparasi *Framework* Golang dan Node.js," *Jurnal Riset Sistem Informasi dan Teknik Informatika*, vol. 8, no. 1, pp. 155–168, Feb. 2023, doi: http://dx.doi.org/10.30645/jurasik.v8i1.551.g529.

[23] F. F. Anhar, M. H. P. Swari, and F. P. Aditiawan, "Analisis Perbandingan Implementasi Clean Architecture Menggunakan MVP, MVI, Dan MVVM Pada Pengembangan Aplikasi Android Native," *Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro dan Informatika*, vol. 2, no. 2, pp. 181–191, Jan. 2024, doi: 10.61132/jupiter.v2i2.155.

[24] U. Syach and W. Martyas Edi, "Perancangan Aplikasi Web Manajemen Data Produk Bisnis Perhiasan Berbasis Flask dan MongoDB," *Jurnal Penerapan Teknologi Informasi dan Komunikasi*, vol. 3, no. 2, pp. 162–176, 2024.

[25] J. Vesanto, "Developing a Web-Based Record Store Using React and Express.js," Thesis, Haaga-Helia University, 2024.

[26] M. Avatara and R. Tan, "Implementasi *Framework* Gin dan gRPC pada Pengembangan Back-end Web," *Jurnal Strategi*, vol. 6, no. 1, pp. 52–57, May 2024.