**TUGAS AKHIR**

**ANALISIS PERFORMA ENAM BACKEND JAVASCRIPT FRAMEWORK MENGGUNAKAN RUNTIME JAVASCRIPT BUN DENGAN METODE GET, POST, PUT DAN DELETE**

****

**MUHAMAD FAIZ**

**203510066**

**PROGRAM STUDI TEKNIK INFORMATIKA**

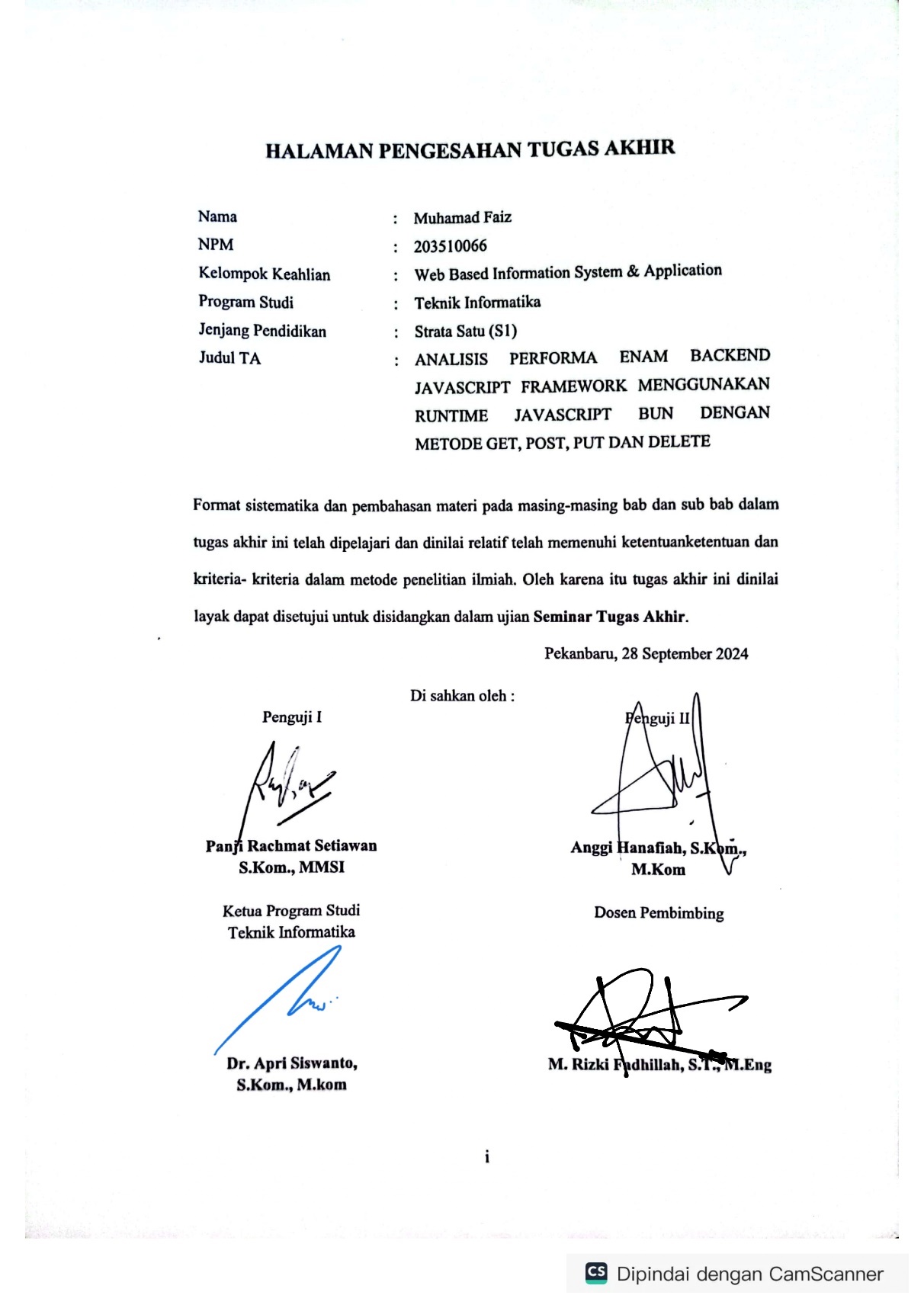
**FAKULTAS TEKNIK**

**UNIVERSITAS ISLAM RIAU**

**PEKANBARU**

**2024**

# HALAMAN PENGESAHAN TUGAS AKHIR



# HALAMAN PENGESAHAN DEWAN PENGUJI TUGAS AKHIR

|  |  |  |
| --- | --- | --- |
| Nama | : | Muhamad Faiz |
| NPM | : | 203510066 |
| Kelompok Keahlian | : | Web Based Information System & Application |
| Program Studi | : | Teknik Informatika |
| Jenjang Pendidikan | : | Strata Satu (S1) |
| Judul TA | : | ANALISIS PERFORMA ENAM BACKEND JAVASCRIPT FRAMEWORK MENGGUNAKAN RUNTIME JAVASCRIPT BUN DENGAN METODE GET, POST, PUT DAN DELETE |

Tugas Akhir ini secara keseluruhan dinilai telah memenuhi ketentuan-ketentuan dan kaidah-kaidah dalam penulisan penelitian ilmiah serta telah diuji dan dapat dipertahankan dihadapan dewan penguji. Oleh karena itu, Tim Penguji Ujian Tugas Akhir Fakultas Teknik Universitas Islam Riau menyatakan bahwa mahasiswa yang bersangkutan dinyatakan Telah Lulus Mengikuti Ujian Tugas Akhir Pada Tanggal 31 Oktober 2024 dan disetujui serta diterima untuk memenuhi salah satu syarat guna memperoleh gelar Sarjana Strata Satu Bidang Ilmu Teknik Informatika.

Pekanbaru,31 Oktober 2024

**Dewan Penguji**

1. Pembimbing : M. Rizki Fadhillah, S.T., M.Eng ( )
2. Penguji 1 : Panji Rachmat Setiawan, S.Kom., MMSI ( )
3. Penguji 2 : Anggi Hanafiah, S.Kom., M.Kom ( )

**Disahkan Oleh :**

|  |  |
| --- | --- |
| Dekan  Fakultas Teknik | Ketua Program Studi  Teknik Informatika |
|  |  |
| **Dr. Deddy Purnomo Retno, S.T, M.T** | **Dr. Apri Siswanto, S.Kom, M.Kom** |
| **NIDN : 1005057702** | **NIDN : 1016048502** |

# PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa tugas akhir ini merupakan karya saya sendiri dan semua sumber yang tercantum didalamnya baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar sesuai ketentuan. Jika terdapat unsur penipuan atau pemalsuan data maka saya bersedia dicabut gelar yang telah saya peroleh.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | Pekanbaru, 30 Oktober 2024 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | Muhamad Faiz |
|  |  | NPM: 203510066 |

# KATA PENGANTAR

**Assalamu’alaikum Warahmatullahi Wabarakatuh.**

Segala puji dan syukur penulis panjatkan kepada Allah SWT atas rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul " **ANALISIS PERFORMA ENAM BACKEND JAVASCRIPT FRAMEWORK MENGGUNAKAN RUNTIME JAVASCRIPT BUN DENGAN METODE GET, POST, PUT DAN DELETE** " Penulisan skripsi ini merupakan bagian dari syarat untuk memperoleh gelar Sarjana Teknik di Program Studi Teknik Informatika Universitas Islam Riau. Dalam penyusunan skripsi ini, penulis menyadari bahwa banyak pihak yang telah membantu dan memberikan dukungan, baik selama proses penyusunan tugas akhir ini maupun selama perkuliahan. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. M. Rizki Fadhillah, S.T., M.Eng selaku dosen pembimbing, yang telah memberikan bimbingan, arahan, serta masukan yang berharga selama proses penulisan skripsi ini.
2. Ketua dan sekretaris prodi serta dosen-dosen yang sangat banyak membantu terkait perkuliahan, ilmu pengetahuan dan hal lain yang tidak dapat saya sebutkan satu per satu.
3. Orang tua dan keluarga yang memberikan dukungan penuh material dan moral.
4. Sahabat serta teman-teman seperjuangan dan lainnya yang tidak bisa disebutkan satu-persatu, yang telah memberikan bantuan, semangat, dan dukungan selama masa perkuliahan dan penyusunan skripsi ini.

Teriring doa saya, semoga Allah membalas segala kebaikan dari semua pihak yang telah membantu. Penulis menyadari bahwa laporan skripsi ini masih belum sempurna. Oleh karena itu, dengan penuh kerendahan hati, penulis sangat mengharapkan kritik dan saran yang membangun untuk perbaikan laporan ini. Semoga skripsi ini dapat menambah wawasan serta memberi manfaat bagi semua pembacanya.

Pekanbaru, 30 Oktober 2024

Penulis,

Muhamad Faiz

203510066

# Daftar isi

[HALAMAN PENGESAHAN TUGAS AKHIR i](#_Toc182864281)

[HALAMAN PENGESAHAN DEWAN PENGUJI TUGAS AKHIR ii](#_Toc182864282)

[PERNYATAAN KEASLIAN TUGAS AKHIR iii](#_Toc182864283)

[KATA PENGANTAR iv](#_Toc182864284)

[Daftar isi vi](#_Toc182864285)

[Daftar Gambar viii](#_Toc182864286)

[Daftar Tabel x](#_Toc182864287)

[ABSTRAK xi](#_Toc182864288)

[ABSTRACT xii](#_Toc182864289)

[BAB I PENDAHULUAN 1](#_Toc182864290)

[1.1 Latar Belakang 1](#_Toc182864291)

[1.2 Identifikasi Masalah 3](#_Toc182864292)

[1.3 Rumusan Masalah 3](#_Toc182864293)

[1.4 Batasan Masalah 3](#_Toc182864294)

[1.5 Tujuan Penelitian 4](#_Toc182864295)

[1.6 Manfaat Penelitian 4](#_Toc182864296)

[BAB II LANDASAN TEORI 5](#_Toc182864297)

[2.1 Tinjauan Pustaka 5](#_Toc182864298)

[2.2 Dasar Teori 11](#_Toc182864299)

[2.2.1 Javascript 11](#_Toc182864300)

[2.2.2 Framework Javascript Backend 11](#_Toc182864301)

[2.2.3 Bun Js 14](#_Toc182864302)

[2.2.4 HTTP (Hypertext Transfer Protocol) 14](#_Toc182864303)

[2.2.5 Parameter Pengujian 15](#_Toc182864304)

[2.2.6 *Asymptotic Notation* 19](#_Toc182864305)

[2.2.7 Weight Scoring System 21](#_Toc182864306)

[2.2.8 MinMax Normalization 22](#_Toc182864307)

[2.3 Kerangka Berpikir 22](#_Toc182864308)

[BAB III METODOLOGI PENELITIAN 24](#_Toc182864309)

[3.1 Pengembangan dan Perancangan Sistem 24](#_Toc182864310)

[3.1.1 Alur Pengerjaan Skripsi 24](#_Toc182864311)

[3.2 Uraian Metodologi 25](#_Toc182864312)

[3.2.1 Tahap Persiapan 25](#_Toc182864313)

[3.2.2 Tahap Pengujian 28](#_Toc182864314)

[3.2.3 Tahap Evaluasi 42](#_Toc182864315)

[3.3 Alat dan Bahan Penelitian 42](#_Toc182864316)

[3.4 Ringkasan skenario apache benchmark 43](#_Toc182864317)

[BAB IV HASIL DAN PEMBAHASAN 47](#_Toc182864318)

[4.1 Hasil Pengujian 47](#_Toc182864319)

[4.1.1 Apache Benchmark 47](#_Toc182864320)

[4.1.2 Mitata Js 70](#_Toc182864321)

[4.2 Pembahasan 71](#_Toc182864322)

[BAB V KESIMPULAN DAN SARAN 73](#_Toc182864323)

[5.1 Kesimpulan 73](#_Toc182864324)

[5.2 Saran 74](#_Toc182864325)

[DAFTAR PUSTAKA 76](#_Toc182864326)

[LAMPIRAN 78](#_Toc182864327)

[Lampiran 1 : data mentah pengujian parameter *time taken for test* 78](#_Toc182864328)

[Lampiran 2 : data mentah pengujian parameter *request per second* 80](#_Toc182864329)

[Lampiran 3 : data mentah pengujian parameter *time per request* 82](#_Toc182864330)

[Lampiran 4 : data mentah pengujian parameter *transfer rate* 85](#_Toc182864331)

[Lampiran 5 : data mentah hasil pengujian *tool mitata js* 87](#_Toc182864332)

# Daftar Gambar

[**Gambar 2. 1** website Apache Lounge 18](#_Toc180311953)

[**Gambar 2. 2** file ab.exe dan abs.exe 18](#_Toc180311954)

[**Gambar 2. 3** isi folder projek 19](#_Toc180311955)

[**Gambar 2. 4** Kerangka Berpikir 23](#_Toc180311956)

[**Gambar 3. 1** Alur Pengerjaan Skripsi 24](#_Toc180311957)

[**Gambar 3. 2** Tahap persiapan membuat folder Benchmark 26](#_Toc180311958)

[**Gambar 3. 3** inialisasi runtime bun pada folder Bun 26](#_Toc180311959)

[**Gambar 3. 4** Struktur Folder Bun 27](#_Toc180311960)

[**Gambar 3. 5** Isi file package.json 28](#_Toc180311961)

[**Gambar 3. 6** Contoh hasil parameter Time Taken for Test 30](#_Toc180311962)

[**Gambar 3. 7** Contoh hasil parameterRequest Per Second 31](#_Toc180311963)

[**Gambar 3. 8** Contoh hasil parameter Transfer Rate 31](#_Toc180311964)

[**Gambar 3. 9**Contoh hasil parameter *Time Per Request* 31](#_Toc180311965)

[**Gambar 3. 10** perintah pengujian HTTP POST 31](#_Toc180311966)

[**Gambar 3. 11** perintah pengujian HTTP GET 32](#_Toc180311967)

[**Gambar 3. 12** perintah pengujian HTTP PUT 32](#_Toc180311968)

[**Gambar 3. 13** perintah pengujian HTTP DELETE 32](#_Toc180311969)

[**Gambar 3. 14** Contoh hasil permintaan POST 33](#_Toc180311970)

[**Gambar 3. 15** Contoh data 33](#_Toc180311971)

[**Gambar 3. 16** Skenario permintaan HTTP POST 34](#_Toc180311972)

[**Gambar 3. 17** Contoh hasil permintaann HTTP GET 35](#_Toc180311973)

[**Gambar 3. 18** Skenario permintaan HTTP GET 35](#_Toc180311974)

[**Gambar 3. 19** Contoh hasil Permintaann HTTP PUT 36](#_Toc180311975)

[**Gambar 3. 20** Skenario permintaan HTTP GET 36](#_Toc180311976)

[**Gambar 3. 21** Contoh hasil Permintaann HTTP DELETE 37](#_Toc180311977)

[**Gambar 3. 22** Skenario permintaan HTTP DELETE 37](#_Toc180311978)

[**Gambar 3. 23** perintah install packet manager mitata 38](#_Toc180311979)

[**Gambar 3. 24** contoh source code 39](#_Toc180311980)

[**Gambar 3. 25** perintah menjalankan tool mitata 39](#_Toc180311981)

# Daftar Tabel

[**Tabel 2. 1** Penelitian terdahulu 6](#_Toc180311982)

[**Tabel 2. 2** Perbandingan ukuran waktu dalam second 16](#_Toc180311983)

[**Tabel 2. 3** Notasi analisis waktu 20](#_Toc180311984)

[**Tabel 3. 1** Skenario Beban Permintaan HTTP POST, GET, PUT dan DELETE 30](#_Toc180311985)

[**Tabel 3. 2** Spesifikasi Perangkat Keras (Hardware) 42](#_Toc180311986)

[**Tabel 3. 3** Spesifikasi Perangkat Lunak (Software) 43](#_Toc180311987)

[**Tabel 3. 4** Ringkasan skenario beban apache benchmark 44](#_Toc180311988)

[**Tabel 4. 1** Hasil pengujian Time Taken for Test 49](#_Toc180311989)

[**Tabel 4. 2** Hasil pengujian Request per Second 53](#_Toc180311990)

[**Tabel 4. 3** Hasil pengujian Time per Request 57](#_Toc180311991)

[**Tabel 4. 4** Hasil pengujian Transfer rate 61](#_Toc180311992)

[**Tabel 4. 5** Frekuensi Performa Framework Backend JavaScript Berdasarkan Jumlah Data 65](#_Toc180311993)

[**Tabel 4. 6** Hasil Weight Score 67](#_Toc180311994)

[**Tabel 4. 7** Hasil rata - rata pengujian mitata js 70](#_Toc180311995)

**ANALISIS PERFORMA ENAM BACKEND JAVASCRIPT FRAMEWORK MENGGUNAKAN RUNTIME JAVASCRIPT BUN DENGAN METODE GET, POST, PUT DAN DELETE**

**Muhamad Faiz**

Program Studi Teknik Informatika, Fakultas Teknik, Univeristas Islam Riau

*Email*: muhamadfaiz@student.uir.ac.id

# ABSTRAK

Di era modern ini, Teknologi Informasi (TI) sangatlah berperan besar pada perkembangan dunia dan tidak ada yang bisa menyangkal dampak inovasi teknologi, terutama pesatnya perkembangan teknologi informasi (TI) dalam pengembangan aplikasi berbasis web. aplikasi berbasis web di dukung oleh banyak bahasa pemrograman dan juga *framework.* Penelitian ini bertujuan untuk membandingkan performa keenam *framework backend javascript* yaitu *express js, hapi js, koa, fastify, nest* dan *elysia* yang berjalan di *runtime Bun* dengan metode permintaan HTTP GET, POST, PUT dan DELETE. Penelitian ini dilaksanakan melalui tahapan metodologi seperti tahap persiapan: mempersiapkan aplikasi dan sistem pada komputer lokal dan menginstall *resource* yang di perlukan seperti menginstall keenam *framework* tersebut, untuk melakukan pada tahap pengujian , tahapan pengujian: proses pengujian menggunakan 2 buah *tools* yatiu *apache benchmark* dan *mitata.*js dengan menggunakan skenario *BIG O Notation* untuk diujikan dan tahapan evaluasi : menganalisis hasil dari pengujian keenam *framework* tersebut, dari tahap sebelumnya hasil pengujian akan di catat ,dibandingkan dan kemudian akan dibuat kesimpulan *framework* manakah yang performanya bagus. Dengan melakukan penelitian ini, di harapkan dapat memberikan kontribusi dalam membantu para *developer* untuk memilih *framework backend javascript* yang berjalan di *runtime Bun* secara efektif.

**Kata kunci** : *framework, javascript, backend,* HTTP, *Bun*

**PERFORMANCE ANALYSIS OF SIX JAVASCRIPT FRAMEWORK BACKENDS USING JAVASCRIPT BUN RUNTIME WITH GET, POST, PUT AND DELETE METHODS**

**Muhamad Faiz**

Department of Informatics Engineering, Faculty of Engineering, Islamic University of Riau

*Email*: muhamadfaiz@student.uir.ac.id

# ABSTRACT

In this modern era, Information Technology (IT) plays a very big role in the development of the world and no one can deny the impact of technological innovation, especially the rapid development of information technology (IT) in the development of web-based applications. Web-based applications are supported by many programming languages ​​and frameworks. This study aims to compare the performance of the six backend javascript frameworks, namely express js, hapi js, koa, fastify, nest and elysia which run on the Bun runtime with the HTTP GET, POST, PUT and DELETE request methods. This study was carried out through methodological stages such as the preparation stage: preparing applications and systems on the local computer and installing the necessary resources such as installing the six frameworks, to carry out the testing stage, testing stage: the testing process uses 2 tools, namely apache benchmark and mitata.js using the BIG O Notation scenario to be tested and the evaluation stage: analyzing the results of testing the six frameworks, from the previous stage the test results will be recorded, compared and then a conclusion will be made which framework has good performance. By conducting this research, it is hoped that it can contribute to helping developers to choose a JavaScript backend framework that runs on the Bun runtime effectively.

***Keywords*** : *framework, javascript, backend,* HTTP, *Bun*

# PENDAHULUAN

## Latar Belakang

Di era modern ini, Teknologi Informasi (TI) sangatlah berperan besar pada perkembangan dunia dan tidak ada yang bisa menyangkal dampak inovasi teknologi, terutama pesatnya perkembangan teknologi informasi (TI) dalam pengembangan aplikasi berbasis web. aplikasi berbasis web di dukung oleh banyak bahasa pemrograman dan juga framework. Di dalam dunia TI pemilihan framework untuk membangun sebuah aplikasi sangatlah penting, dikarenakan pemilihan framework untuk kebutuhan bukanlah hal yang gampang bagi pengembang(del Pilar Salas-Zárate et al., 2015).

*Framework* merupakan kerangka kerja yang membantu para pengembang untuk membuat sebuah aplikasi berbasis web dengan kode yang terstruktur, mudah dan cepat. *Framework* dalam pengembangan aplikasi berbasis web terbagi menjadi dua jenis yaitu *front-end framework* dan *back-end framework*. untuk *front-end framework* kerangka tampilan pada website, sehingga website tersebut menjadi lebih menarik dan memudahkan user untuk menggunakan layanan pada website. Sedangkan *back-end framework* digunakan untuk bekerja di sisi *server* seperti pemrosesan data, parsing data, suplai data, penanganan request, routing dan validasi data.

*Back-end JavaScript framework* merupakan kerangka kerja yang digunakan untuk membangun aplikasi *server-side* menggunakan *javaScript*. *Framework* tersebut menyediakan sejumlah fitur dan alat yang memudahkan pengembangan aplikasi web, termasuk pengelolaan rute (*routing*), pemrosesan permintaan dan *response* HTTP, manajemen basis data, dan lain sebagainya. Ada beberapa *back-End framework* yang umum digunakan oleh sebagian besar pengembang saat ini, yaitu ada *Koa, Express, Elysia, Fastify, Nest dan Hapi*. pada dasarnya *framework* tersebut dijalankan dengan *runtime javascript node js*.

Pada saat ini ada beberapa jenis *runtime javascript*. Salah satu *runtime* itu adalah *Bun*. *Bun* salah satu teknologi pengembangan website berbasis *javascript* di sisi *server* berupa *runtime* dan menyediakan *built-in HTTP server*. *Bun* merupakan *javascript runtime* baru yang di bangun untuk melayani ekosistem pada *javascript* modern dan *Bun* juga dirancang untuk menjadi alternatif *javascript runtime* yang bekerja secara secepat mungkin, sehingga runtime tersebut masih tahap pengembangan, Bun juga menggunakan mesin *javascript V8* yang dimodifikasi, *Bun* juga di rancang agar tidak tergantung pada paket Node,js eksternal.

Metode GET, POST, PUT, dan DELETE merupakan metode HTTP yang digunakan untuk berinteraksi dengan sumber daya (*resource*) di server. Metode GET digunakan untuk mengambil data dari server, POST untuk mengirim data baru ke server, PUT untuk memperbarui data yang ada di server, dan DELETE untuk menghapus data dari server.

Dari beberapa uraian diatas, Penulis tertarik untuk membuat sebuah analisis untuk menguji pada enam *framework javascript back-end* yang berjalan di *Bun* dengan skripsi yang berjudul “Analisis Performa Enam *Backend Javascript Framework* menggunakan *Runtime Javascript Bun* dengan Metode GET, POST, PUT dan DELETE”.

## Identifikasi Masalah

Berdasarkan permasalahan yang telah diuraikan diatas, maka identifikasi masalah dalam penelitian ini sebagai berikut:

1. Para pengembang kesulitan memilih *framework* *back-end*
2. Kurangnya informasi yang berkaitan dengan *runtime javascript Bun*

## Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dikemukakan diatas maka dapat dirumuskan masalah sebagai berikut:

1. Bagaimana cara melakukan pengujian perbandingan performansi dari keenam *Back-End JavaScript framework* yang diujikan?
2. *Framework* mana yang memiliki nilai performansi terbaik?

## Batasan Masalah

Batasan masalah yang akan dibahas dibawah ini meliput beberapa hal pokok yaitu :

1. Sistem Operasi yang digunakan adalah Windows 11 Home v23H2
2. Runtime javascript menggunakan Bun v1.1.21
3. Metode HTTP GET, POST, PUT dan DELETE
4. Menggunakan 6 *framework back-end javascript*

## Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. mengetahui perbandingan performansi dari keenam *Back-End JavaScript framework*
2. mengetahui *framework* mana yang berjalan di *runtime bun*

## Manfaat Penelitian

Penelitian ini diharapkan mampu memberikan referensi kepada para pengembang, agar dapat mengetahui kinerja atau performa dari setiap *framework backend javascript* yang berjalan di *runtime bun*, sehingga dapat menentukan *framework backend* mana yang sesuai dengan kebutuhan

# LANDASAN TEORI

## Tinjauan Pustaka

Tujuan dari penelitian ini dimaksudkan untuk mengetahui perbandingan performa antara enam *back-end javascript*. Untuk tinjauan pustaka pada penelitian ini adalah studi literatur, dimana yang merujuk menurut penelitian terdahulu adalah sebagai berikut:

**Tabel 2. 1** Penelitian terdahulu

| No | Judul | Penulis | Metode | Hasil | Kekurangan dan Kelebihan |
| --- | --- | --- | --- | --- | --- |
| 1 | ANALISIS PERBANDINGAN PERFORMA RESTFULL API ANTARA EXPRESS.JS DENGAN LARAVEL FRAMEWORK | (Hadinata & Stianingsih, 2024) | Performance Testing | Hasil dari penelitian tersebut, terlihat bahwa Express.js memiliki waktu respon, penggunaan CPU dan penggunaan memori yang lebih efisien dari Laravel. | **Kekurangan**: Penelitian ini hanya membandingkan dua framework, Laravel dan Express.js, dan menggunakan satu kasus penggunaan tertentu  **Kelebihan**: Framework Laravel dan Express.js diuji pada dua server yang berbeda dengan spesifikasi yang sama, memastikan bahwa setiap framework dapat menangani setiap permintaan tanpa berbagi sumber daya |
| 2 | ANALISIS PERBANDINGAN PERFORMA WEB SERVICE REST MENGGUNAKAN FRAMEWORK LARAVEL, DJANGO, DAN Node JS PADA APLIKASI BERBASIS WEBSITE | (Amarulloh, 2023) | deskriptif kuantitatif | 1. NodeJS dan Django memiliki keunggulan dibandingkan laravel dengan tingkat keberhasilan permintaan data dari 3000, 5000, hingga 7000 data  2. Framework NodeJS (ExpressJS) lebih unggul dari Django dan Laravel  3. Framework Django lebih unggul dari NodeJS dan Laravel dalam jumlah permintaan per detik dan HTML Transferred. | **Kekurangan**: Kurangnya penjelasan yang mendalam tentang metode pengumpulan data, pengujian, dan analisis yang digunakan dalam penelitian  **Kelebihan**: Penelitian memberikan pemahaman tentang pentingnya efisiensi dalam penggunaan sumber daya dan sensitivitas aplikasi perangkat lunak terhadap delay. |
| 3 | Analisis Perbandingan Kinerja *Framework* Codeigniter Dengan ExpressJs Pada *Server* RESTful Api | (Mulana et al., 2022) | Performance testing | Hasil pengujian kinerja framework Codeigniter bersama dengan framework Express.js menunjukan bahwa rata rata response time dari framework Express,js adalah 420,72 ms lebih cepat dibandingkan framework Codeigniter yang memperoleh rata rata response time 555,90 ms. Tetapi disisi framework Express.js yang berbeda biasanya membutuhkan lebih banyak sumber daya. penggunaan CPU 1,95% dan penggunaan memori 2,74% lebih besar sedikit dibandingkan dengan penggunaan CPU framework Codeigniter yang memiliki penggunaan memori rata-rata 2,31% dan rata-rata 1,34%. | **Kekurangan**: Penelitian ini hanya membandingkan dua framework, Codeigniter dan Express.js, dan menggunakan satu kasus penggunaan tertentu  **Kelebihan**: penggunaan metode performance testing yang jelas dan terstruktur untuk membandingkan kinerja dua framework yang berbeda |
| 4 | Analisis perbandingan Codeigniter dan Yii framework pada perancangan website rencana anggaran biaya | (Suwarno & Afandi, 2022) | metode perbandingan menggunakan diagram blok. | 1.Yii framework lebih efisien dan cepat dalam proses perancangan aplikasi web karena memiliki Yii Generator yang membantu dalam penulisan CRUD (Create, Read, Update, Delete) secara otomatis.  2.Yii framework memiliki kecepatan yang sedikit lebih baik daripada Codeigniter framework dalam pengujian speed index.  3.Codeigniter framework memiliki keamanan yang lebih rendah dibandingkan Yii framework.  4.Codeigniter framework memiliki URL yang lebih rapi dan singkat dibandingkan Yii framework. | **Kekurangan**:  1.Codeigniter:Tidak memiliki fitur generator kode seperti Yii framework, memiliki keamanan yang lebih rendah dibandingkan Yii framework.  2.Yii : Memiliki URL yang lebih panjang, memerlukan waktu lebih lama untuk memahami proses alur dari Yii framework bagi pemula.  **Kelebihan**:  1.Codeigniter Lebih cepat dan efisien dalam proses perancangan, memiliki keamanan yang baik, URL yang rapi dan singkat.  2.Yii lebih efisien dan cepat dalam proses perancangan karena memiliki Yii Generator, memiliki keamanan yang baik. |
| 5 | PERFORMA MICROFRAMEWORK PHP PADA REST API MENGGUNAKAN METODE LOAD TESTING | (Yatini et al., 2021) | Load Testing | Hasil dari penelitian ini menunjukkan bahwa pada aplikasi uji, angka terbaik didapat pada jumlah koneksi di bawah 128 dengan rate antara 2000 - 3000 RPS | **Kekurangan**: metode pembobotan hasil pengujian belum menghasilkan angka yang lebih sesuai dan akurat  **Kelebihan**: adanya analisis perbandingan performa microframework PHP pada REST API, yang penting untuk menentukan kesesuaian pada pengembangan aplikasi berbasis API |
| 6 | Perbandingan Pengembangan Front End Menggunakan Blade Template dan Vue Js | (Chastro et al., 2020) | Audits panel pada chrome dev tools | Blade Template memiliki performa yang lebih cepat daripada Vue Js. Namun, Vue Js memberikan lebih banyak kemudahan dalam mengembangkan aplikasi website yang kompleks. Selain itu, dalam pengujian kecepatan scripting, Blade Template juga menunjukkan kecepatan yang jauh lebih tinggi dibandingkan dengan Vue Js. Oleh karena itu, kesimpulannya adalah untuk membangun aplikasi sederhana sebaiknya menggunakan Blade Template, sementara untuk aplikasi yang kompleks, Vue Js dapat menjadi pilihan yang lebih baik. | **Kekurangan**:  1.semakin kompleks aplikasi tersebut, performa Blade Template dapat menurun.  2.pada aplikasi yang sederhana, performa Vue Js kalah jika dibandingkan dengan Blade Template.  **Kelebihan**:  1. Kelebihan dari Blade Template adalah performanya yang cepat, terutama untuk aplikasi sederhana. Blade Template juga memungkinkan untuk membuat aplikasi dengan performa yang stabil  2. kelebihan dari Vue Js adalah kemampuannya dalam mengembangkan aplikasi website yang kompleks dengan lebih mudah. Vue Js juga menawarkan performa yang stabil meskipun aplikasi tersebut kompleks |
| 7 | ANALISIS PERBANDINGAN METODE SOAP DAN REST YANG DIGUNAKAN PADA FRAMEWORK FLASK UNTUK MEMBANGUN WEB SERVICE | (Putra & Putera, 2019) | SOAP dan REST | Hasil dari penelitian menunjukkan bahwa dalam pengujian web service berbasis Flask menggunakan metode SOAP dan REST, REST memiliki performa yang lebih baik daripada SOAP dalam pengiriman request dan respon | **Kekurangan**: Tidak disebutkan apakah penelitian mempertimbangkan faktor faktor lain yang dapat memengaruhi performa, seperti ukuran data, kecepatan jaringan, atau kompleksitas aplikasi.  **Kelebihan**: Penggunaan framework Flask sebagai basis penelitian memberikan wawasan praktis tentang implementasi metode SOAP dan REST dalam lingkungan nyata. |

## Dasar Teori

### Javascript

*JavaScript* merupakan bahasa pemrograman tingkat tinggi yang pertama kali dibuat pada tahun 1995, pada awalnya dimaksudkan untuk berjalan di browser atau sisi klien, tetapi seiring berjalannya waktu, sekarang dapat berjalan di komputer *server* atau *server-side*(Chastro et al., 2020; Dirjen et al., 2017)*.*

### Framework Javascript Backend

*Framework JavaScript backend* merupakan kerangka kerja pengembangan perangkat lunak yang digunakan untuk membangun bagian dari aplikasi web yang berjalan di sisi server atau *backend*(Celi-Párraga et al., 2023). *Framework* ini membantu dalam mengelola logika aplikasi, interaksi dengan *database*, dan menyediakan layanan ke sisi klien atau *frontend*(Peña-Monferrer & Diaz-Marin, 2022). Terdapat enam buah *framework* yang populer saat ini antara lain:

1. *Express js*

*Framework Express.js* adalah kerangka kerja *backend Node.js* yang memungkinkan pengembang untuk membangun aplikasi web dengan mudah dan efisien. *Express.js* membantu dalam mengelola rute (*routes*), permintaan HTTP, tanggapan, serta berbagai aspek pengembangan *backend* lainnya(Nurhayati & Agussalim, 2023).

1. *Elysia*

*Framework Elysia js* merupakan kerangka kerja web *javascript* untuk *runtime bun*. *Framework* ini dibangun dengan bahasa pemrograman *typescript* agar berfokus pada kinerja yang sederhana dan juga fleksibel. Salah satu keunggulan dari *framework elysia* ini yaitu performa tinggi dan mudah digunakan(elysiajs, 2024).

1. *Koa*

*Koa* merupakan kerangka web baru yang dikembangkan oleh tim di belakang *Express*, bertujuan untuk menjadi landasan yang lebih kecil, ekspresif, dan kuat untuk aplikasi web dan API(Koa, 2024). Fungsi asinkronnya memungkinkan menghilangkan panggilan balik dan meningkatkan penanganan kesalahan secara signifikan. *Koa* tidak menggunakan *middleware* apa pun di dalamnya, sebaliknya, Koa menawarkan berbagai metode inovatif yang membuat server penulisan cepat dan menyenangkan.

1. *Hapi*

*Framework Hapi* merupakan sebuah *framework* aplikasi web yang *open source* untuk *Node.js* yang memungkinkan pengembang untuk membuat aplikasi web yang kokoh, aman, dan skalabel. *Hapi* menekankan struktur yang konsisten dan konfigurasi yang ekstensif, serta menyediakan berbagai fitur bawaan seperti *routing*, *input validation*, *error handling*, dan *caching*(Hapi, 2024).

*Hapi* dirancang untuk mempermudah pengembangan aplikasi web dengan menyediakan alat dan metode yang jelas. Dengan pendekatan yang terstruktur dan fokus pada keamanan, *Hapi* cocok digunakan untuk membangun aplikasi web besar dan kompleks.

1. *Fastify*

*Framework fastify* merupakan kerangka kerja web yang mengutamakan penyediaan overhead seminimal mungkin dan pengalaman pengembang terbaik, disertai dengan arsitektur plugin yang tangguh(Fastify, 2024).

1. *Nest*

*Framework Nest* merupakan sebuah *framework Node.js* yang sangat kuat dan progresif untuk membangun aplikasi *server-side* yang efisien, dapat diuji, dan mudah di-maintain(nest, 2024). *Nest* menggunakan *TypeScript* sebagai bahasa utamanya dan mengikuti pola desain yang berbasis pada konsep *Dependency Injection*, sehingga memungkinkan pengembang untuk membuat kode yang terstruktur dengan baik dan mudah diuji.

*Nest* menyediakan berbagai fitur bawaan seperti modul, *middleware, decorator*, dan juga integrasi yang kuat dengan *Express.js, Fastify*, dan banyak teknologi lainnya. Dengan pendekatan yang modular dan konfigurasi yang ekstensif, *Nest* cocok digunakan untuk membangun aplikasi web kompleks dan skalabel.

### Bun Js

*Bun.js* merupakan sekumpulan alat JavaScript bersumber terbuka yang lengkap yang dengan mudah menggabungkan berbagai komponen *JavaScript* sisi *server* yang penting untuk menawarkan solusi yang berkinerja tinggi. Meskipun namanya lucu, *Bun* menggabungkan fitur beberapa alat, seperti *runtime* *Node* atau *Deno*, manajer paket seperti NPM atau pnpm, dan alat pembangunan seperti *Webpack* atau *Vite*. Awal mulanya merupakan pekerjaan sampingan satu orang, Bun dengan cepat menjadi pesaing yang kompetitif untuk pendekatan pengembangan web konvensional(Ahmod, 2023).

### HTTP (Hypertext Transfer Protocol)

*Hypertext Transfer Protocol (HTTP)* adalah protokol yang digunakan untuk komunikasi antara klien dan server di internet. Ini adalah protokol terpenting yang saat ini digunakan di internet(Nayak et al., 2022)*. Hypertext Transfer Protocol (HTTP)* merupakan protokol tingkat aplikasi untuk sistem informasi hypermedia terdistribusi, kolaboratif. Ini adalah protokol generik dan *stateless* yang dapat digunakan untuk berbagai tugas di luar hiperteks, seperti nama *server* dan sistem manajemen objek terdistribusi. Menggunakan antarmuka pengguna berbasis web, HTTP memungkinkan pengguna mengirim permintaan ke server HTTP mana pun yang terhubung ke internet, membuat aliran data antara klien dan *server* terlihat. Perangkat klien dan perangkat komputasi jarak jauh dapat terhubung dengan layanan jarak jauh melalui penggunaan HTTP. HTTP dapat digunakan untuk mengakses sumber daya di *server* dengan mengirimkan permintaan akses HTTP, untuk alasan keamanan, permintaan ini dapat dienkripsi. Selain itu, metode pemrosesan permintaan HTTP dapat digunakan untuk menghindari gangguan komunikasi dan misinformasi. Untuk metode permintaan pada HTTP sebagai berikut:

1. GET

Metode GET meminta representasi dari sumber daya yang ditentukan. Permintaan menggunakan GET seharusnya hanya mengambil data.

1. POST

Metode POST mengirimkan entitas ke sumber daya tertentu, sering kali menyebabkan perubahan status atau efek samping pada server.

1. PUT

Metode PUT menggantikan semua representasi sumber daya target saat ini dengan payload permintaan.

1. DELETE

Metode DELETE menghapus sumber daya yang ditentukan.

### Parameter Pengujian

Pada pengujian ini menggunakan 2 metode pengujian yaitu *microbenchmark* dan *stress testing.* Untuk penjelasan lebih lanjut sebagai berikut:

1. *Microbenchmark*

*Microbenchmark* merupakan jenis pengujian kinerja yang focus pada penilaian kinerja operasi atau fungsi kecil dan sangat spesifik dari sebuah sistem. Dalam konteks *runtime Bun, microbenchmarking* digunakan untuk mengukur kecepatan operasi atau bagian kecil dari kode untuk memastikan performa yang optimal.

Untuk pengujian ini akan menggunakan paket manager *mitata.js*. *mitata js* merupakan alat umum yang bagus untuk *microbenchmarking* dan berguna menguji kinerja operasi atau fungsi *javascript* yang sangat spesifik dan kecil. Untuk perbandingan waktu pada *mitata js* dapat dilihat pada tabel 2.2.

**Tabel 2. 2** Perbandingan ukuran waktu dalam second

|  |  |
| --- | --- |
| Unit | Per 1 second |
| ps (*picosecond)* | 1e-12s |
| ns (*nano second)* | 0.000000001s |
| μs (*microsecond*) | 0.000001s |
| ms (*milisecond*) | 0.001s |
| s (second) | 1 |

1. *Stress testing*

*Stress testing* adalah cara untuk mengukur bagaimana suatu keadaan dapat berdampak pada tingkat ketahanan sistem(Ginasari et al., 2021). Dalam rekayasa perangkat lunak *stress testing* merupakan teknik pengujian terhadap sebuah *software* untuk mengetahui ketahanan *software* atau sistem saat digunakan melalui batas normal. Ketahanan yang dimaksud yaitu termasuk menguji stabilitas dan *reliability* dari sebuah *software* atau sistem. Dengan *stress test* akan dapat menghadapi kondisi yang sangat berat dan memastikan bahwa *software* tidak mengalami kegagalan, terutama untuk unit sistem yang dapat memiliki konsekuensi fatal jika mengalami kegagalan dalam jangka waktu yang cukup lama. Adapun beberapa parameter pengujian yang diuji dalam sebuah kinerja *framework* yaitu:

*Time Taken for Test*

*Time Taken for Test* merupakan jumlah waktu yang diperlukan dari saat koneksi pertama ke soket hingga saat respons terakhir diterima(Apache.org, 2024). Nilai yang dihasilkan dapat dikatakan bagus apabila nilainya lebih rendah maka waktu yang dibutuhkan untuk menangani semua *request* lebih cepat.

*Request Per Second*

*Request Per Second* merupakan jumlah permintaan per detik. Apabila nilai yang dihasilkan semakin besar, itu menunjukkan bahwa *web server* yang diujikan dapat menyelesaikan banyak permintaan dalam satu detik. Ini menunjukkan nilai yang bagus pada *Request Per Second*(Azi et al., 2023).

*Transfer Rate (Kb/s)*

*Transfer Rate* merupakan metrik standar yang digunakan untuk menghitung kecepatan di mana data atau informasi bergerak dari satu tempat ke tempat lain. Apabila nilai transfer yang dihasilkan cukup besar, rate transfer dapat dianggap baik(Azi et al., 2023).

*Time Per Request (ms)*

*Time Per Request* merupakan Rata-rata waktu yang dihabiskan per permintaan. Apabila nilai *Time per Request* semakin kecil, itu menunjukkan jika *web server* dapat menangani permintaan dengan lebih cepat(Satwika & Semadi, 2020).

Cara install :

1. Install Apache Bencmark di windows terlebih dahulu
2. Versi terbaru Apache Benchmark untuk windows dapat download di <https://www.apachelounge.com/download/>, pilih sesuai versi arsitektur windows.



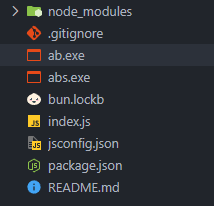
**Gambar 2. 1** website Apache Lounge

1. Pindah ke folder Apache24/bin dan salin file ab dan abs.



**Gambar 2. 2** file ab.exe dan abs.exe

1. pindahkan file ab,exe dan abs.exe ke folder projek



**Gambar 2. 3** isi folder projek

### *Asymptotic Notation*

*Asymptotic Notation* merupakan notasi matematika yang digunakan untuk menunjukkan berapa lama waktu yang dibutuhkan suatu algoritma bekerja ketika input cenderung menuju nilai tertentu. Ada 3 notasi pada Asymptotic yaitu:

1. *Big O Notation (O)*

*Big O Notation* merupakan sebuah notasi matematika yang memiliki 2 buah kegunaan. Salah satunya untuk menganalisa kompleksitas waktu dari sebuah algoritma. *Big O Notation* juga digunakan untuk mepresentasikan batas atas pada suatu algoritma yaitu kondisi terburuk atau bisa disebut *worst case*.

1. *Big Omega Notation (Ω)*

*Big Omega Notation* merupakan salah satu notasi analisis asimptotik yang digunakan untuk menyatakan batas bawah kompleksitas waktu dari suatu algoritma yaitu kondisi terbaik atau bisa disebut *best case.*

1. *Big Theta Notation (Θ)*

*Big Theta Notation* merupakan salah satu notasi asimtotik digunakan untuk menilai kinerja algoritma, mengidentifikasi kasus terbaik dan terburuknya, bisa dikatakan kondisi rata-rata dari suatu algoritma atau bisa disebut *average case.*

Daftar notasi fungsi yang biasa ditemui saat menganalisis waktu berjalan di suatu algoritma dapat dilihat pada Tabel 2.3*.*

**Tabel 2. 3** Notasi analisis waktu

| Notasi | Deskripsi | Contoh |
| --- | --- | --- |
| O(1)/konstan | Algoritma membutuhkan waktu yang sama, tidak peduli seberapa besar inputnya. | Mengakses elemen array dengan indeks tertentu, mengambil nilai variabel. |
| O(log n)/logaritma | kompleksitas meningkat satu unit untuk setiap dua kali lipat data masukan | menemukan item di pohon pencarian seimbang |
| O(n)/linear | Waktu yang dibutuhkan algoritma berbanding lurus dengan besarnya input. Semakin besar input, semakin lama waktu yang dibutuhkan. | Mencari elemen tertentu dalam daftar dengan cara iterasi (pencarian linear) |
| O(n log n)/Log Linear | kompleksitas tumbuh sebagai kombinasi linier atau logaritmik | menggabungkan pengurutan pada kumpulan item |
| O(n2)/kuadrat | waktu yang dibutuhkan sebanding dengan kuadrat jumlah elemen | memeriksa semua kemungkinan pasangan dalam array |
| O(n3)/kubik | waktu eksekusi sebanding dengan pangkat tiga jumlah elemen | perkalian matriks dari matriks nxn |
| O(2n)/ eksponensial | waktu berlipat ganda untuk setiap elemen baru yang ditambahkan | menghasilkan semua himpunan bagian dari himpunan tertentu |
| O(n!)/faktorial | kompleksitas tumbuh secara faktorial berdasarkan ukuran kumpulan data | menentukan semua permutasi dari daftar tertentu |

### Weight Scoring System

*Weight Scoring System* merupakan nilai yang diukur yang menunjukkan kinerja atau efisiensi suatu entitas (seperti teknologi, metode, atau algoritma) berdasarkan sejumlah parameter atau kriteria yang diberi bobot berdasarkan tingkat kepentingannya. Nilai untuk setiap parameter dikalikan dengan bobotnya, dan kemudian nilai total dikumpulkan. Proses ini memungkinkan evaluasi proporsional dari berbagai elemen, yang menghasilkan satu angka yang dapat dengan mudah digunakan untuk membandingkan kinerja antar entitas. Untuk membuat perbandingan lebih adil dan konsisten, normalisasi sering digunakan pada skor berat untuk menempatkan semua nilai pada rentang yang seragam (misalnya, 0% hingga 100%). Persamaan 2.1 menunjukkan persamaan untuk menghitung *weight score.*

|  |  |  |
| --- | --- | --- |
|  | *weightScoring* = (*x’1 \* bobot1) +* … + (*x’n* \* *bobotn* ) | (2.1) |

### MinMax Normalization

*MinMax Normalization* merupakan metode normalisasi yang mengubah nilai data dari 0 hingga 1. Persamaan 2.2 menunjukkan persamaan untuk menghitung MinMax Normalization(Permana & Salisah, 2022).

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

## Kerangka Berpikir

Untuk memecahkan masalah yang sedang diteliti dalam penelitian ini diterapkan kerangka berpikir sistematis yang disusun dari *problems* (permasalahan), *approach* (pendekatan), *development* (pengembangan), *implementation* (implementasi), *measurement* (pengukuran), dan *result* (hasil). Gambar dari kerangka pemikiran dapat dilihat pada berikut.

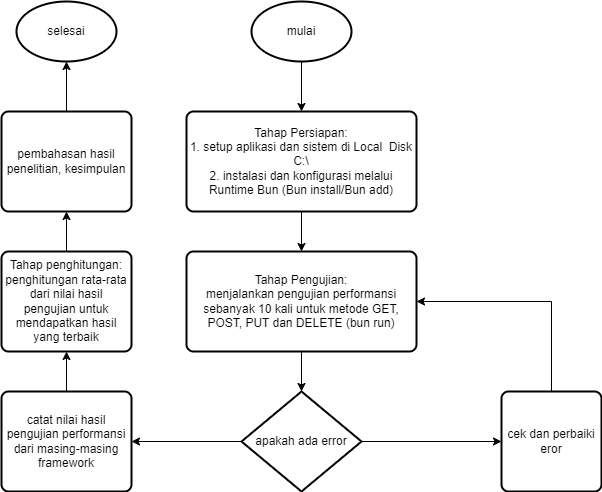
**Gambar 2. 4** Kerangka Berpikir

# METODOLOGI PENELITIAN

## Pengembangan dan Perancangan Sistem

### Alur Pengerjaan Skripsi

Penelitian dilakukan dengan mengikuti alur pengerjaan skripsi, yang menunjukkan tahapan yang akan dilakukan dari awal hingga akhir. Alur pengerjaan dapat dilihat pada gambar dibawah ini:



**Gambar 3. 1** Alur Pengerjaan Skripsi

Untuk Langkah pertama dari pengerjaan skripsi ini yaitu tahap persiapan, pada tahap ini dilakukan setup folder di direktori Local Disk C:/ dengan nama *benchmark*, didalam folder tersebut akan di instalasi *Bun* yang dibutuhkan dengan perintah *Bun add* dan *npm install* serta membuat *source code* dari tiap-tiap *framework* yang akan di ujikan.

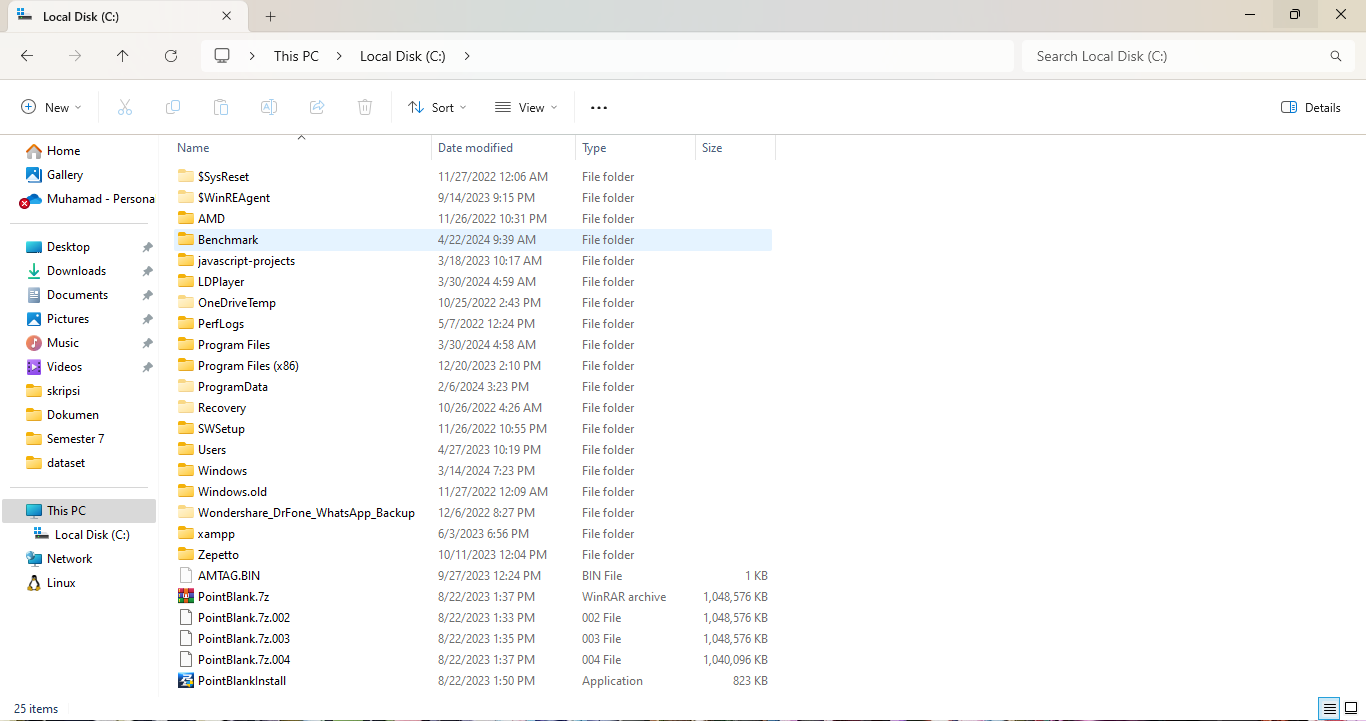
Selanjutnya untuk Langkah kedua yaitu tahap pengujian, dimana pada tahap ini akan dilakukan pengujian terhadap masing-masing *framework* dengan metode GET, POST, PUT dan DELETE, dengan menjalankan perintah *bun run* akan dilakukan sebanyak 10 kali pengujian. Jika ada *error* dalam proses pengujian berlangsung maka peneliti akan cek dan perbaiki *error* tersebut sampai pengujian berhasil di lakukan. Setelah melakukan pengujian maka nilai performa dari masing- masing *framework* akan dicatat, nilai performa yang diambil yaitu nilai rata-rata.

Langkah yang ketiga yaitu tahap penghitungan, dimana pada tahap ini akan dilakukan penghitungan hasil rata-rata dari tahap pengujian yang dilakukan sebanyak 10 kali pengujian, sehingga nantinya akan mendapatkan hasil rata-rata yang terbaik.

Untuk langkah terakhir dalam alur pengerjaan skripsi tersebut yaitu menganalisis, membahas serta membuat kesimpulan dari hasil tahap pengujian sebelumnya.

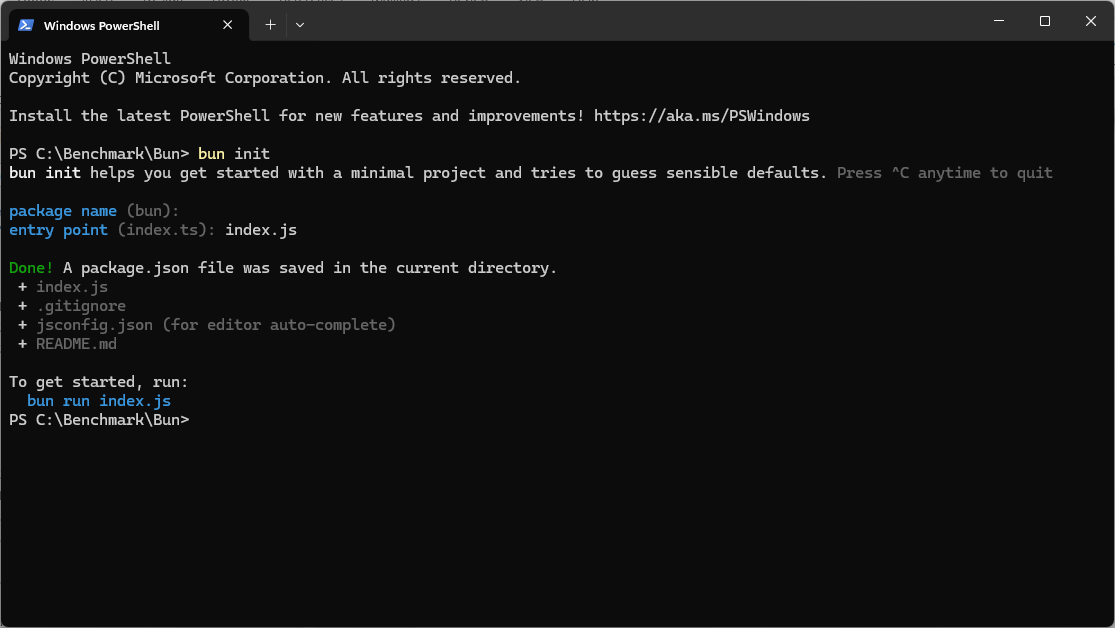
## Uraian Metodologi

### Tahap Persiapan

Langkah pertama dalam tahap persiapan adalah membuat sebuah direktori/folder yang bernama Benchmark di Local Disk C:\. Folder Benchmark tersebut akan di pasangi beberapa *packet manager* dan pembuatan *source code* untuk pengujian keenam *framework*. Contoh letak folder Benchmark ditunjukkan pada Gambar 3.2.

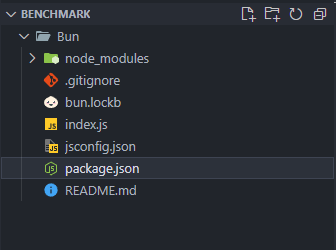
**Gambar 3. 2** Tahap persiapan membuat folder Benchmark

Langkah kedua dari tahap persiapan yaitu menginstall *resource* yang di perlukan pada tahap selanjutnya. Hal yang pertama untuk tahap pengembangan di folder Bun yaitu menginialisasikan folder tersebut dengan perintah *Bun init* pada terminal.



**Gambar 3. 3** inialisasi runtime bun pada folder Bun

Pada Gambar 3.4 dibawah ini merupakan struktur folder Bun setelah proses inialiasi.



**Gambar 3. 4** Struktur Folder Bun

Untuk membantu proses penelitian, dibutuhkan repository online NodeJS Package Manager (NPM). Melalui library yang tersedia, NPM membantu proses instalasi keenam framework berbasis NodeJS tersebut.

Untuk konfigurasi dan struktur format pertukaran data, penyimpanan data, versi, *dependencies*, *framework*, dan router dari masing-masing *framework*, disimpan pada file *JavaScript Object Notation (JSON)* bernama package.json. Isi dari file package.json sebagai berikut:



**Gambar 3. 5** Isi file package.json

#### Tahap Pengujian

Tahap pengujian merupakan tahap untuk menguji kemampuan keenam *framework*. Tahap ini akan menggunakan 2 metode yaitu *microbenchmark* dan *stress test*. *Microbenchmark* merupakan jenis pengujian kinerja yang fokus pada penilaian kinerja operasi atau fungsi kecil dan sangat spesifik dari sebuah sistem*,* sedangkan, *Stress test* merupakan teknik atau metode pengujian terhadap sebuah *software* untuk mengetahui ketahanan *software* atau sistem dimana pengujian ini akan menitik beratkan kemampuan ketahanan masing – masing *framework* diluar batas normal, Dengan menggunakan metode *stress test* akan dapat menghadapi kondisi yang sangat berat dan memastikan bahwa *software* tidak mengalami kegagalan, terutama untuk unit sistem yang dapat memiliki konsekuensi fatal jika mengalami kegagalan dalam jangka waktu yang cukup lama. Tahap ini akan menggunakan 10 data, 100 data, 500 data dan 1000 data untuk melakukan proses pengujian dengan metode GET, POST PUT dan DELETE dari masing-masing *framework*.

Untuk melakukan pengujian tersebut, peneliti akan menggunakan 2 buah *tool*  yaitu *Apache Benchmark* dan *mitata js*. Dari masing-masing *tool* tersebut akan diljalankan sebanyak 10 kali pengujian untuk mendapatkan nilai rata-rata terbaik. Untuk penjelasan 2 buah *tool* tersebut akan dijelaskan di bawah ini.

1. *Apache Benchmark*

*Apache* *benchmark* merupakan sebuah alat yang dibuat oleh *Organization* yang dapat bekerja untuk mengukur performansi pada HTTP. *Tool* tersebut dapat di analogikan jumlah pengunjung yang mencoba untuk mengakses. Pada penelitian ini akan menggunakan *request* sebanyak 50, 500, 5000 dan 10.000, dengan jumlah *concurrency* (jumlah permintaan yang dilakukan dalam 1 waktu) sebanyak 50 permintaan untuk masing – masing *framework*. Parameter yang akan dibandingkan ada 4 yaitu *Time Taken for Test, Request Per Second, Transfer Rate* dan *Time Per Request*. Untuk skenario beban pengujian permintaan setiap HTTP dan *framework*, dapat dilihat pada tabel 3.1.

**Tabel 3. 1** Skenario Beban Permintaan HTTP POST, GET, PUT dan DELETE

|  |  |  |
| --- | --- | --- |
| **Framework** | **C** | **n** |
| express | 50 | 50, 500, 5000, 10.000 |
| hapi | 50 | 50, 500, 5000, 10.000 |
| koa | 50 | 50, 500, 5000, 10.000 |
| nest | 50 | 50, 500, 5000, 10.000 |
| elysia | 50 | 50, 500, 5000, 10.000 |
| fastify | 50 | 50, 500, 5000, 10.000 |

Pada Tabel 3.1 setiap *framework* menunjukkan variabel beban pengujian berupa *concurrency* dan *request*. Untuk variabel *concurrency* berjumlah 50 sedangkan pada variabel *request* ada 4 nilai kondisi yaitu 50, 500, 5000 dan 10.000 yang digunakan untuk setiap permintaan metode HTTP.

Untuk parameter kinerja pengujian pada *tool Apache Benchmark*, akan di jelaskan dibawah ini.

*Time Taken for Test*

Merupakan jumlah waktu yang diperlukan dari saat koneksi pertama ke soket hingga saat respons terakhir diterima. Nilai yang dihasilkan dapat dikatakan bagus apabila nilainya lebih rendah maka waktu yang dibutuhkan untuk menangani semua *request* lebih cepat



**Gambar 3. 6** Contoh hasil parameter Time Taken for Test

1. *Request Per Second*

Merupakan jumlah permintaan per detik. Apabila nilai yang dihasilkan semakin besar, itu menunjukkan bahwa *framework* yang diujikan dapat menyelesaikan banyak permintaan dalam satu detik. Ini menunjukkan nilai yang bagus pada *Request Per Second.*



**Gambar 3. 7** Contoh hasil parameterRequest Per Second

1. *Transfer Rate*

Metrik standar yang digunakan untuk menghitung kecepatan di mana data atau informasi bergerak dari satu tempat ke tempat lain. Apabila nilai transfer yang dihasilkan cukup besar, rate transfer dapat dianggap baik.



**Gambar 3. 8** Contoh hasil parameter Transfer Rate

1. *Time Per Request*

Rata-rata waktu yang dihabiskan per permintaan. Apabila nilai *Time per Request* semakin kecil, itu menunjukkan jika *framework* dapat menangani permintaan dengan lebih cepat.



**Gambar 3. 9**Contoh hasil parameter *Time Per Request*

Cara penggunaan :

1. Jalankan perintah berikut di terminal projek:



**Gambar 3. 10** perintah pengujian HTTP POST



**Gambar 3. 11** perintah pengujian HTTP GET



**Gambar 3. 12** perintah pengujian HTTP PUT



**Gambar 3. 13** perintah pengujian HTTP DELETE

Keterangan sintaks:

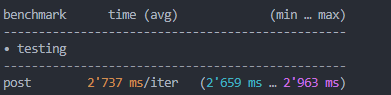
|  |  |  |
| --- | --- | --- |
| ./ab | : | Menjalankan program Apache Benchmark (ab) dari direktori lokal. ./ menunjukkan bahwa ab berada di direktori saat ini. |
|  |  |  |
| -m | : | Menentukan metode HTTP yang akan digunakan |
|  |  |  |
| -c | : | Menentukan concurrency level, yaitu jumlah permintaan yang dikirimkan secara paralel pada saat yang sama |
|  |  |  |
| -n | : | Menentukan total jumlah permintaan yang akan dikirimkan ke server |

1. *Mitata.js*

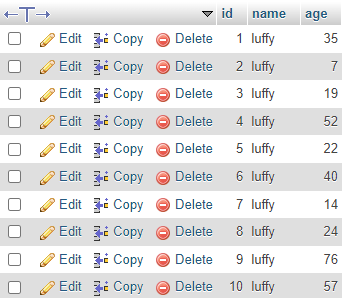
*Mitata js* merupakan alat umum yang bagus untuk *microbenchmarking* dan berguna menguji kinerja operasi atau fungsi *javascript* yang sangat spesifik dan kecil. Pada *tool* ini yang menjadi parameter pengujiannya yaitu waktu rata – rata pemrosesan dalam *second*. Untuk skenario pengujian dengan menggunakan *mitata.js* dengan setiap permintaan pada HTTP sebagai berikut:

1. POST

Skenario permintaan HTTP POST merupakan tahap untuk mengirimkan data yang bertahap di mulai dari 10, 100, 500 dan 1000 data. Untuk data yang akan dikirimkan hanya id, name dan age, data tersebut akan disimpan ke *database* MySQL, kemudian di catat berapa waktu rata-rata yang di perlukan untuk mengirimkan data tersebut. Untuk contoh hasil pengujian permintaan HTTP POST dan contoh data yang telah tersimpan di *database* ditunjukkan pada Gambar 3.17 dan Gambar 3.14.

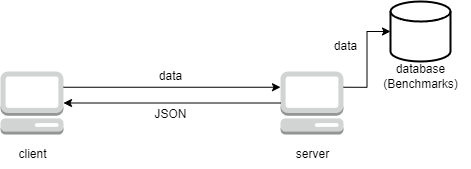


**Gambar 3. 14** Contoh hasil permintaan POST



**Gambar 3. 15** Contoh data

Skenario pengujian untuk permintaan HTTP POST dapat dilihat pada Gambar 3.16.

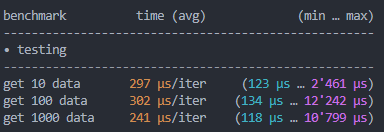


**Gambar 3. 16** Skenario permintaan HTTP POST

Gambar 3.16 menunjukkan desain skenario permintaan HTTP POST, dimana terdapat komputer *client* mengirimkan data secara bertahap sebanyak 10, 100, 500 dan 1000 data ke komputer *server*, komputer *server* tersebut telah di installkan keenam *framework*, selanjutnya data tersebut akan disimpan ke dalam *database* yang bernama *Benchmarks* dan untuk *response* ke *client* akan berbentuk *Javascript Object Notation (JSON)*.

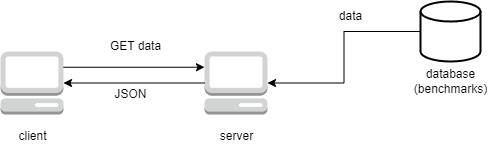
1. GET

Skenario permintaan HTTP GET merupakan tahap untuk mengambil data dari *database*. Untuk skenario pengujian ini, ada 4 skenario yaitu pertama permintaan GET sebanyak 10 data, kedua permintaan GET sebanyak 100 data, ketiga permintaan GET sebanyak 500 data dan terakhir permintaan GET sebanyak 1000 data. Dari ketiga skenario tersebut akan dicatat berapa waktu rata-rata yang diperlukan untuk mengambil data tersebut. Untuk contoh hasil pengujian permintaan HTTP GET ditunjukkan pada Gambar 3.20.



**Gambar 3. 17** Contoh hasil permintaann HTTP GET

Skenario pengujian untuk permintaan HTTP GET dapat dilihat pada Gambar 3.18.

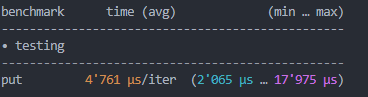


**Gambar 3. 18** Skenario permintaan HTTP GET

Gambar 3.18 menunjukkan desain skenario permintaan HTTP GET, dimulai dari permintaan 10 data, 100 data, 500 data dan 1000 data, dimana terdapat komputer *client* mengirimkan permintaan ke komputer *server* untuk memberikan *response* berupa data sebanyak dimulai dari 10, 100, 500 dan 1000 data yang berbentuk *Javascript Object Notation (JSON)*.

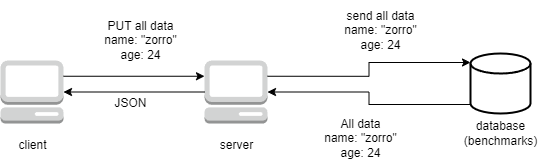
1. PUT

Skenario permintaan HTTP PUT merupakan tahap untuk mengubah data dari *database*. Pada permintaan HTTP ini merubah semua data dimana data yang akan di ubah yaitu *name* dan *age*. Untuk mengubah semua data tersebut akan di catat berapa waktu rata-rata yang dibutuhkan. Untuk contoh hasil pengujian permintaan HTTP PUT ditunjukkan pada Gambar 3.22.



**Gambar 3. 19** Contoh hasil Permintaann HTTP PUT

Skenario pengujian untuk permintaan HTTP PUT dapat dilihat pada Gambar 3.20.

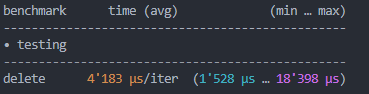


**Gambar 3. 20** Skenario permintaan HTTP GET

Gambar 3.20 menunjukkan desain skenario permintaan HTTP PUT, dimana terdapat komputer *client* mengirimkan sebuah payload ke komputer *server* untuk diubah datanya, kemudian komputer *server* akan mengirimkan *query* untuk mengubah semua data dari *client* dan kemudian data tersebut diperbaharui di *database*. Setelah berhasil di perbaharui komputer *server* akan mengirimkan *response* ke *client* akan berbentuk *Javascript Object Notation (JSON)*.

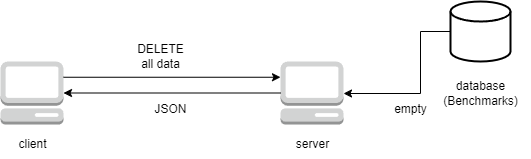
1. DELETE

Skenario permintaan HTTP DELETE merupakan tahap untuk menghapus data dari *database*. Pada permintaan HTTP ini semua data akan di hapus dari *database*. Kemudian akan dicatat berapa waktu rata – rata yang dibutuhkan untuk menghapus semua data tersebut. Untuk contoh hasil pengujian permintaan HTTP DELETE ditunjukkan pada Gambar 3.21.



**Gambar 3. 21** Contoh hasil Permintaann HTTP DELETE

Skenario pengujian untuk permintaan HTTP DELETE dapat dilihat pada Gambar 3.22.

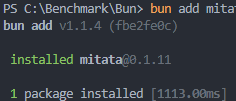


**Gambar 3. 22** Skenario permintaan HTTP DELETE

Gambar 3.22 menunjukkan desain skenario permintaan HTTP DELETE, dimana terdapat komputer *client* mengirimkan sebuah permintaan ke komputer *server* untuk dihapus semua datanya dari *database*, kemudian komputer *server* akan mengirimkan *query* untuk menghapus semua data ke *database* dan kemudian data tersebut dihapus dari *database* dan setelah berhasil dihapus, komputer *server* akan mengirimkan *response* ke *client* akan berbentuk *Javascript Object Notation (JSON)*.

Cara penggunaan:

1. install *packet manager mitata.js* di projek terlebih dahulu.
2. Buka terminal dan pindah ke direktori projek yang bernama Benchmark, selanjutnya tuliskan perintah seperti ini:



**Gambar 3. 23** perintah install packet manager mitata

1. Buatlah sebuah file yang bernama mitata,js(nama boleh bebas), untuk pembuatan source code untuk melakukan pengujian. Contoh source code ada pada gambar 3.24:



**Gambar 3. 24** contoh source code

1. Jalankan file mitata.js dengan perintah:



**Gambar 3. 25** perintah menjalankan tool mitata

Untuk mengetahui kompleksitas dan efisiensi waktu dari setiap skenario masing – masing *framework* dalam pengujian ini, peneliti akan menghitung dan menganalisa algoritma tersebut. Strategi yang akan digunakan yaitu semakin rendah waktu yang dibutuhkan maka akan semakin baik skenario atau algoritma tersebut meskipun ukuran input yang dibutuhkan semakin tinggi. Penelitian pengujian ini akan menggunakan konsep *Asymptotic Notation.*

Untuk mengetahui kondisi waktu dari ketiga notasi asimtotik pada pengujian ini, di bawah ini adalah contoh *pseudocode* dan notasi *asymptotic* terutama untuk permintaan HTTP GET:

1. Skenario GET *route* ‘/’

1. BEGIN

2. IMPORT framework library

3. INITIALIZE framework app

4. SET port = 3000

5. DEFINE HTTP GET route '/':

WHEN route is requested:

RESPOND with HTTP status 200 and data as JSON ("hello world") **//O(1)**

6. START server on specified port:

PRINT "Server running at “<http://localhost:3000/>"

7. END

*Pseudocode* diatas merupakan contoh skenario untuk permintaan HTTP GET. Pada Langkah 5 pada *pseudocode* diatas, peneliti melabelkan notasi. Dalam konsep *Asymptotic Notation*, langkah 5diatas termasuk ke dalam notasi O(1)/konstan dan menggunakan *Big Omega Notation* sebagai kasus terbaiknya(*Best Case)*. *Pseudocode* tersebut hanya memberikan *response* dengan angka 200 untuk status *success* apabila melakukan *request* dan memberikan *respons* berbentuk JSON dengan string “Hello World” pada permintaan HTTP GET dengan *route* ‘/’.

1. Skenario GET route ‘/users’

1. BEGIN

2. IMPORT framework library

3. IMPORT mysql library

4. INITIALIZE framework app

5. SET port = 3000

6. CREATE MySQL connection object:

SET host = 'localhost'

SET user = 'root'

SET password = 'password'

SET database = 'your\_database'

7. ATTEMPT to connect to MySQL database:

IF connection fails THEN

PRINT "Error connecting to MySQL:", error details

STOP execution

ELSE

PRINT "Connected to MySQL"

8. DEFINE HTTP GET route '/users':

WHEN route is requested:

CREATE SQL query to select all user data :

"SELECT \* FROM users"

EXECUTE SQL query:

IF query fails THEN

RESPOND with HTTP status 500 and message "Database query error" **//O(1)**

STOP execution

ELSE

RESPOND with HTTP status 200 and user data as JSON //**O(1)**

9. START server on specified port:

PRINT "Server running at http://localhost:3000/"

10. END

*Pseudocode* diatas merupakan contoh skenario GET *route* ‘/users’, Pada Langkah 8 pada *pseudocode* diatas, peneliti melabelkan notasi. Dalam konsep *Asymptotic Notation* untuk notasinya yaitu O(1) atau konstan dan menggunakan *Big Omega Notation* sebagai kasus terbaiknya(*Best Case)*. *Pseudocode* tersebut memberikan *response* dengan angka 200 untuk status *success* apabila melakukan *request* dan memberikan *respons* berbentuk JSON dengan menampilkan seluruh data *users.* pada permintaan HTTP GET dengan *route* ‘/users’.

### Tahap Evaluasi

Tahap evaluasi merupakan tahap menganalisis hasil dari pengujian keenam *framework* tersebut, dari tahap sebelumnya hasil pengujian akan dibuat kesimpulan *framework* manakah yang performanya bagus.

## Alat dan Bahan Penelitian

Dalam menganalisa performa enam *backend javascript framework* dengan *runtime* *bun.js* dalam pengujian metode GET, POST, PUT dan DELETE ini, memerlukan beberapa perangkat lunak dan perangkat keras untuk mendukung Analisa pengujian ini.

1. Spesifikasi Perangkat Keras(*Hardware*)

Perangkat Keras (*Hardware*) yang digunakan dalam Analisa performa ini adalah sebagai berikut:

**Tabel 3. 2** Spesifikasi Perangkat Keras (Hardware)

|  |  |  |
| --- | --- | --- |
| No | Spesifikasi | Keterangan |
| 1 | Tipe Laptop | HP Laptop 15-ef0xxx |
| 2 | CPU | AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx |
| 3 | Harddisk | 500GB |
| 4 | RAM | 8 GB |
| 5 | Tipe Sistem Operasi | Sistem Operasi 64-bit |

1. Spesifikasi Perangkat Lunak(Software)

Perangkat Lunak (*Software*) yang digunakan dalam analisis performa ini adalah sebagai berikut:

**Tabel 3. 3** Spesifikasi Perangkat Lunak (Software)

|  |  |  |
| --- | --- | --- |
| No | Spesifikasi | Keterangan |
| 1 | Sistem Operasi | Windows 11 Home v23H2 |
| 2 | Framework | Express js, Hapi js, Koa, Nest, Elysia, Fastify |
| 3 | Javascript Runtime | Bun v1.1.21 |
| 4 | Database | Mysql |
| 5 | Text Editor | Visual Studio Code |
| 6 | Packet Manager | NPM |
| 7 | Tool Benchmark | Apache Benchmark v2.4, mitata.js v0.0.6 |

## Ringkasan skenario apache benchmark

Dibawah ini merupakan tabel ringkasan skenario *apache benchmark* Dimana menyajikan rangkuman skenario pengujian performa enam *framework backend JavaScript* (*Express, Hapi, Koa, Nest, Elysia, dan Fastify*) menggunakan *apache benchmark*. Pengujian dilakukan pada beberapa skenario HTTP, yaitu permintaan POST, GET, PUT, dan DELETE, yang disimulasikan dalam berbagai kondisi beban. Setiap *framework* diuji untuk mengukur kemampuan dalam menangani *time taken for test, request per second, time per request* dan *transfer rate* di bawah beban permintaan yang berbeda-beda. Data ini akan memberikan gambaran mengenai performa masing-masing framework dalam lingkungan yang serupa, sehingga dapat membantu dalam analisis dan pemilihan framework yang paling efisien untuk keperluan pengembangan aplikasi.berikut ringkasan skenario dapat dilihat pada Tabel 3.4.

**Tabel 3. 4** Ringkasan skenario beban apache benchmark

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) |
| --- | --- | --- | --- | --- | --- |
|
| GET | konstan | O(1) | 0 | 50 | 50 |
| GET | konstan | O(1) | 0 | 50 | 500 |
| GET | konstan | O(1) | 0 | 50 | 5000 |
| GET | konstan | O(1) | 0 | 50 | 10000 |
| GET | linear | O(n) | 10 | 50 | 50 |
| GET | linear | O(n) | 10 | 50 | 500 |
| GET | linear | O(n) | 10 | 50 | 5000 |
| GET | linear | O(n) | 10 | 50 | 10000 |
| GET | linear | O(n) | 100 | 50 | 50 |
| GET | linear | O(n) | 100 | 50 | 500 |
| GET | linear | O(n) | 100 | 50 | 5000 |
| GET | linear | O(n) | 100 | 50 | 10000 |
| GET | linear | O(n) | 500 | 50 | 50 |
| GET | linear | O(n) | 500 | 50 | 500 |
| GET | linear | O(n) | 500 | 50 | 5000 |
| GET | linear | O(n) | 500 | 50 | 10000 |
| GET | linear | O(n) | 1000 | 50 | 50 |
| GET | linear | O(n) | 1000 | 50 | 500 |
| GET | linear | O(n) | 1000 | 50 | 5000 |
| GET | linear | O(n) | 1000 | 50 | 10000 |
| POST | linear | O(n) | 10 | 50 | 50 |
| POST | linear | O(n) | 10 | 50 | 500 |
| POST | linear | O(n) | 10 | 50 | 5000 |
| POST | linear | O(n) | 10 | 50 | 10000 |
| POST | linear | O(n) | 100 | 50 | 50 |
| POST | linear | O(n) | 100 | 50 | 500 |
| POST | linear | O(n) | 100 | 50 | 5000 |
| POST | linear | O(n) | 100 | 50 | 10000 |
| POST | linear | O(n) | 500 | 50 | 50 |
| POST | linear | O(n) | 500 | 50 | 500 |
| POST | linear | O(n) | 500 | 50 | 5000 |
| POST | linear | O(n) | 500 | 50 | 10000 |
| POST | linear | O(n) | 1000 | 50 | 50 |
| POST | linear | O(n) | 1000 | 50 | 500 |
| POST | linear | O(n) | 1000 | 50 | 5000 |
| POST | linear | O(n) | 1000 | 50 | 10000 |
| PUT | linear | O(n) | 10 | 50 | 50 |
| PUT | linear | O(n) | 10 | 50 | 500 |
| PUT | linear | O(n) | 10 | 50 | 5000 |
| PUT | linear | O(n) | 10 | 50 | 10000 |
| PUT | linear | O(n) | 100 | 50 | 50 |
| PUT | linear | O(n) | 100 | 50 | 500 |
| PUT | linear | O(n) | 100 | 50 | 5000 |
| PUT | linear | O(n) | 100 | 50 | 10000 |
| PUT | linear | O(n) | 500 | 50 | 50 |
| PUT | linear | O(n) | 500 | 50 | 500 |
| PUT | linear | O(n) | 500 | 50 | 5000 |
| PUT | linear | O(n) | 500 | 50 | 10000 |
| PUT | linear | O(n) | 1000 | 50 | 50 |
| PUT | linear | O(n) | 1000 | 50 | 500 |
| PUT | linear | O(n) | 1000 | 50 | 5000 |
| PUT | linear | O(n) | 1000 | 50 | 10000 |
| DELETE | linear | O(n) | 10 | 50 | 50 |
| DELETE | linear | O(n) | 10 | 50 | 500 |
| DELETE | linear | O(n) | 10 | 50 | 5000 |
| DELETE | linear | O(n) | 10 | 50 | 10000 |
| DELETE | linear | O(n) | 100 | 50 | 50 |
| DELETE | linear | O(n) | 100 | 50 | 500 |
| DELETE | linear | O(n) | 100 | 50 | 5000 |
| DELETE | linear | O(n) | 100 | 50 | 10000 |
| DELETE | linear | O(n) | 500 | 50 | 50 |
| DELETE | linear | O(n) | 500 | 50 | 500 |
| DELETE | linear | O(n) | 500 | 50 | 5000 |
| DELETE | linear | O(n) | 500 | 50 | 10000 |
| DELETE | linear | O(n) | 1000 | 50 | 50 |
| DELETE | linear | O(n) | 1000 | 50 | 500 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 |
| DELETE | linear | O(n) | 1000 | 50 | 10000 |

Pada Tabel 3.4, skenario di atas dapat diambil contoh sebuah sistem yang menguji performa beberapa metode HTTP, yaitu GET, POST, PUT, dan DELETE, dalam berbagai kondisi beban. Skenario ini melibatkan pengujian kompleksitas yang konstan (O(1)) dan linear (O(n)) untuk masing-masing metode, dengan jumlah data mulai dari 0 data, 10 data, 100 data, 500 data, hingga 1000 data. Setiap pengujian dilakukan dengan jumlah *concurrency* tetap, yaitu 50 *request*, yang merupakan kapasitas maksimum server untuk menangani permintaan secara bersamaan pada suatu waktu, dan variasi jumlah *request* (n) sebanyak 50 *request*, 500 *request*, 5000 *request*, hingga 10000 *request*. Dalam pengujian ini, nilai n (jumlah total permintaan) dan c (*concurrency*) saling berhubungan karena c menentukan berapa banyak permintaan yang dapat diproses server secara paralel pada suatu waktu. Jika nilai n jauh lebih besar daripada c, maka permintaan akan dikelola dalam *batch*, dengan server memproses hingga batas *concurrency* sebelum melanjutkan ke permintaan berikutnya, yang dapat memengaruhi waktu total pengujian dan performa sistem secara keseluruhan. Pengujian ini bertujuan untuk mengukur respons server terhadap peningkatan jumlah data dan permintaan, serta mengevaluasi efisiensi sistem dalam menangani beban yang berbeda-beda.

# HASIL DAN PEMBAHASAN

## Hasil Pengujian

Hasil pengujian performa enam *framework backend javascript* yang menggunakan 2 *tools* yaitu *apache benchmark* dan *mitata js.* Untuk *apache benchmark* itu menggunakan 4 parameter yaitu *time taken for test, request per second, time per request* dan *transfer rate,* sedangkan *mitata js* menggunakan perbandingan waktu dalam *second*.

### Apache Benchmark

Hasil pengujian performa enam *framework backend javascript* yang menggunakan *apache benchmark* untuk beban pengujian yaitu *request* sebanyak 50 *request*, 500 *request*, 5000 *request* dan 10.000 *request*, dengan jumlah *concurrency* (kapasitas maksimum server untuk menangani permintaan secara bersamaan pada suatu waktu) sebanyak 50 permintaan untuk masing – masing *framework* dan menggunakan 10 data, 100 data, 500 data dan 1000 data secara bertahap dan di ujikan sebanyak 10 kali. Empat parameter yang diukur dalam pengujian ini meliputi: time taken for test, requests per second, time per request, dan transfer rate, yang data mentahnya disajikan secara lengkap pada Lampiran 1, Lampiran 2, Lampiran 3, dan Lampiran 4.

Dibawah ini adalah hasil pengujian dengan metode HTTP dengan 4 parameter yaitu *time taken for test, request per second, time per request* dan *transfer rate* yang telah dirata – rata kan . Dapat dilihat dari Tabel 4.1 sampai 4.4. Tabel ini menggunakan skema warna untuk memudahkan interpretasi hasil, di mana warna hijau menandakan performa terbaik dan warna merah menandakan performa terburuk.

**Tabel 4. 1** Hasil pengujian Time Taken for Test

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | framework | | | | | | framework terbaik |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 0,0384 | 0,0432 | 0,0362 | 0,3472 | 0,0178 | 0,0261 | elysia |
| GET | konstan | O(1) | 0 | 50 | 500 | 0,1732 | 0,1937 | 0,1499 | 0,1792 | 0,0977 | 0,1346 | elysia |
| GET | konstan | O(1) | 0 | 50 | 5000 | 1,0356 | 1,1085 | 1,0811 | 1,1155 | 0,9387 | 0,9972 | elysia |
| GET | konstan | O(1) | 0 | 50 | 10000 | 2,0241 | 2,1423 | 2,0293 | 2,3141 | 2,0306 | 2,4848 | express |
| GET | linear | O(n) | 10 | 50 | 50 | 0,1055 | 0,0676 | 0,067 | 0,074 | 0,0628 | 0,1862 | elysia |
| GET | linear | O(n) | 10 | 50 | 500 | 0,3955 | 0,2713 | 0,375 | 0,3815 | 0,2575 | 0,8175 | elysia |
| GET | linear | O(n) | 10 | 50 | 5000 | 6,2847 | 4,4187 | 3,272 | 3,9064 | 2,3397 | 8,066 | elysia |
| GET | linear | O(n) | 10 | 50 | 10000 | 6,6488 | 4,6135 | 6,56 | 7,3896 | 4,0687 | 16,309 | elysia |
| GET | linear | O(n) | 100 | 50 | 50 | 0,0986 | 0,0649 | 0,0726 | 0,0931 | 0,0625 | 0,1169 | elysia |
| GET | linear | O(n) | 100 | 50 | 500 | 0,9791 | 0,3288 | 0,2625 | 0,3095 | 0,8907 | 0,2436 | fastify |
| GET | linear | O(n) | 100 | 50 | 5000 | 2,7864 | 2,8259 | 2,6282 | 5,7289 | 1,7411 | 2,2599 | elysia |
| GET | linear | O(n) | 100 | 50 | 10000 | 5,5246 | 5,8409 | 4,9663 | 5,8616 | 3,3213 | 4,5906 | elysia |
| GET | linear | O(n) | 500 | 50 | 50 | 0,0768 | 0,0686 | 0,0701 | 0,0709 | 0,0403 | 0,0618 | elysia |
| GET | linear | O(n) | 500 | 50 | 500 | 0,7244 | 0,7002 | 0,6904 | 0,7051 | 0,3866 | 0,6227 | elysia |
| GET | linear | O(n) | 500 | 50 | 5000 | 7,037 | 7,1106 | 6,5683 | 3,9601 | 4,4601 | 5,3009 | nest |
| GET | linear | O(n) | 500 | 50 | 10000 | 15,1498 | 15,0468 | 12,9822 | 15,4216 | 7,7453 | 12,6532 | elysia |
| GET | linear | O(n) | 1000 | 50 | 50 | 0,0767 | 0,122 | 0,1308 | 0,0693 | 0,0965 | 0,1139 | nest |
| GET | linear | O(n) | 1000 | 50 | 500 | 0,5252 | 0,6889 | 0,5293 | 0,4017 | 0,6622 | 0,3519 | fastify |
| GET | linear | O(n) | 1000 | 50 | 5000 | 7,2569 | 11,2259 | 6,1407 | 11,508 | 5,7801 | 7,7804 | elysia |
| GET | linear | O(n) | 1000 | 50 | 10000 | 19.0784 | 26.1924 | 12.9943 | 15.3358 | 12.3229 | 20.0929 | elysia |
| POST | linear | O(n) | 10 | 50 | 50 | 0,0862 | 0,0885 | 0,0603 | 0,0798 | 0,1709 | 0,1207 | koa |
| POST | linear | O(n) | 10 | 50 | 500 | 0,3108 | 0,3497 | 0,278 | 0,2884 | 0,7163 | 0,7209 | koa |
| POST | linear | O(n) | 10 | 50 | 5000 | 5,4394 | 6,184 | 2,6513 | 3,9802 | 6,9071 | 7,3528 | koa |
| POST | linear | O(n) | 10 | 50 | 10000 | 6,4087 | 7,7166 | 4,7534 | 8,349 | 12,9583 | 14,324 | koa |
| POST | linear | O(n) | 100 | 50 | 50 | 0,1818 | 0,1009 | 0,1074 | 0,1162 | 0,7135 | 0,1366 | hapi |
| POST | linear | O(n) | 100 | 50 | 500 | 2,8715 | 2,5154 | 1,9916 | 3,9155 | 4,3498 | 1,2768 | fastify |
| POST | linear | O(n) | 100 | 50 | 5000 | 14,9127 | 16,218 | 18,398 | 18,325 | 43,377 | 8,2718 | fastify |
| POST | linear | O(n) | 100 | 50 | 10000 | 39,2985 | 25,176 | 24,851 | 52,465 | 92,451 | 22,206 | fastify |
| POST | linear | O(n) | 500 | 50 | 50 | 0,5756 | 0,4018 | 0,3456 | 0,6917 | 2,3103 | 0,3465 | koa |
| POST | linear | O(n) | 500 | 50 | 500 | 6,4699 | 5,5429 | 3,9396 | 9,6702 | 20,1139 | 3,7567 | fastify |
| POST | linear | O(n) | 500 | 50 | 5000 | 64,2844 | 40,0783 | 38,5355 | 89,4311 | 228,3872 | 38,0486 | fastify |
| POST | linear | O(n) | 500 | 50 | 10000 | 185,6257 | 107,7004 | 105,8857 | 251,0277 | 426,4329 | 102,4271 | fastify |
| POST | linear | O(n) | 1000 | 50 | 50 | 1,0312 | 0,6117 | 0,7646 | 1,6652 | 4,2894 | 0,6172 | hapi |
| POST | linear | O(n) | 1000 | 50 | 500 | 11,5806 | 6,6481 | 6,4082 | 14,4813 | 44,0291 | 6,552 | koa |
| POST | linear | O(n) | 1000 | 50 | 5000 | 181,4216 | 102,0725 | 103,7445 | 237,4862 | 427,4554 | 92,9336 | fastify |
| POST | linear | O(n) | 1000 | 50 | 10000 | 20.5586 | 22.5931 | 11.809 | 36.5536 | 36.9568 | 36.7367 | koa |
| PUT | linear | O(n) | 10 | 50 | 50 | 0,1299 | 0,0658 | 0,059 | 0,0678 | 0,062 | 0,1438 | koa |
| PUT | linear | O(n) | 10 | 50 | 500 | 0,3653 | 0,2776 | 0,3502 | 0,3827 | 0,2481 | 0,8124 | elysia |
| PUT | linear | O(n) | 10 | 50 | 5000 | 6,0905 | 4,4363 | 3,1399 | 3,8777 | 2,4496 | 8,0784 | elysia |
| PUT | linear | O(n) | 10 | 50 | 10000 | 6,2601 | 5,611 | 5,8541 | 6,8517 | 4,4626 | 15,3879 | elysia |
| PUT | linear | O(n) | 100 | 50 | 50 | 0,0824 | 0,0666 | 0,0729 | 0,0818 | 0,0562 | 0,1088 | elysia |
| PUT | linear | O(n) | 100 | 50 | 500 | 0,3917 | 0,4082 | 0,3712 | 0,4549 | 1,187 | 0,4197 | koa |
| PUT | linear | O(n) | 100 | 50 | 5000 | 3,4631 | 3,501 | 3,184 | 3,3049 | 2,6686 | 3,0724 | elysia |
| PUT | linear | O(n) | 100 | 50 | 10000 | 7,3002 | 7,3347 | 6,7047 | 6,8126 | 5,5099 | 6,3682 | elysia |
| PUT | linear | O(n) | 500 | 50 | 50 | 0,0937 | 0,1023 | 0,1 | 0,0911 | 0,0746 | 0,0905 | elysia |
| PUT | linear | O(n) | 500 | 50 | 500 | 0,9429 | 0,8681 | 0,8687 | 0,862 | 0,7131 | 0,8906 | elysia |
| PUT | linear | O(n) | 500 | 50 | 5000 | 8,9936 | 8,8102 | 8,535 | 4,6834 | 7,9674 | 7,4194 | nest |
| PUT | linear | O(n) | 500 | 50 | 10000 | 18,3836 | 19,3959 | 17,4142 | 18,5596 | 14,2634 | 17,2779 | elysia |
| PUT | linear | O(n) | 1000 | 50 | 50 | 0,0936 | 0,175 | 0,186 | 0,0788 | 0,1724 | 0,1762 | nest |
| PUT | linear | O(n) | 1000 | 50 | 500 | 0,6632 | 0,8921 | 0,647 | 0,4543 | 1,3958 | 0,4268 | fastify |
| PUT | linear | O(n) | 1000 | 50 | 5000 | 16,3948 | 15,5577 | 7,9287 | 15,7515 | 12,9913 | 10,5984 | koa |
| PUT | linear | O(n) | 1000 | 50 | 10000 | 13.6412 | 15.3966 | 10.8566 | 28.3287 | 27.972 | 31.2664 | koa |
| DELETE | linear | O(n) | 10 | 50 | 50 | 0,076 | 0,0996 | 0,0636 | 0,0663 | 0,0616 | 0,1185 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 500 | 0,341 | 0,263 | 0,3008 | 0,3576 | 0,238 | 0,7494 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 5000 | 5,6222 | 4,0197 | 2,8396 | 3,4277 | 2,2613 | 7,6703 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 10000 | 5,7024 | 4,8311 | 5,2303 | 6,7789 | 4,1684 | 14,9275 | elysia |
| DELETE | linear | O(n) | 100 | 50 | 50 | 0,0644 | 0,0598 | 0,0536 | 0,0607 | 0,0608 | 0,1009 | koa |
| DELETE | linear | O(n) | 100 | 50 | 500 | 0,1644 | 0,2144 | 0,1578 | 0,2208 | 1,0237 | 0,1569 | fastify |
| DELETE | linear | O(n) | 100 | 50 | 5000 | 1,603 | 1,9807 | 1,2922 | 1,7247 | 1,0747 | 1,1904 | elysia |
| DELETE | linear | O(n) | 100 | 50 | 10000 | 3,1585 | 4,1924 | 2,5493 | 3,4477 | 2,1126 | 2,3119 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 50 | 0,0175 | 0,0176 | 0,0158 | 0,0174 | 0,0129 | 0,0151 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 500 | 0,1836 | 0,188 | 0,1412 | 0,1816 | 0,119 | 0,1311 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 5000 | 1,8331 | 2,1985 | 1,5835 | 1,9486 | 1,1679 | 1,3868 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 10000 | 3,5723 | 4,2543 | 3,047 | 3,8001 | 2,1801 | 2,5312 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 50 | 0,0507 | 0,0765 | 0,0503 | 0,058 | 0,0302 | 0,0403 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 500 | 0,304 | 0,3937 | 0,2845 | 0,3272 | 0,1671 | 0,2415 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 1,8498 | 2,2215 | 1,6216 | 1,9885 | 1,1233 | 1,3558 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 10000 | 2.2015 | 3.2846 | 1.9365 | 3.101 | 1.7002 | 2.0524 | elysia |

**Tabel 4. 2** Hasil pengujian Request per Second

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | framework | | | | | | framework terbaik |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 1315.306 | 1175.617 | 1384.068 | 1057.595 | 2843.194 | 1919.902 | elysia |
| GET | konstan | O(1) | 0 | 50 | 500 | 2980.221 | 2637.051 | 3342.378 | 2791.919 | 5118.516 | 3777.866 | elysia |
| GET | konstan | O(1) | 0 | 50 | 5000 | 4829.794 | 4517.503 | 4645.198 | 4482.676 | 5345.166 | 5014.793 | elysia |
| GET | konstan | O(1) | 0 | 50 | 10000 | 4941.392 | 4674.919 | 4928.42 | 4351.637 | 4950.376 | 4047.429 | elysia |
| GET | linear | O(n) | 10 | 50 | 50 | 533.04 | 760.262 | 807.34 | 693.234 | 823.437 | 369.925 | elysia |
| GET | linear | O(n) | 10 | 50 | 500 | 1274.616 | 1860.573 | 1347.155 | 1322.885 | 1954.636 | 614.507 | elysia |
| GET | linear | O(n) | 10 | 50 | 5000 | 798.551 | 1213.248 | 1533.331 | 1283.327 | 2147.208 | 620.79 | elysia |
| GET | linear | O(n) | 10 | 50 | 10000 | 1506.245 | 2186.374 | 1535.388 | 1364.73 | 2466.304 | 617.829 | elysia |
| GET | linear | O(n) | 100 | 50 | 50 | 553.993 | 778.008 | 693.912 | 580.791 | 829.401 | 429.127 | elysia |
| GET | linear | O(n) | 100 | 50 | 500 | 512.889 | 1528.853 | 1918.343 | 1618.275 | 561.594 | 2066.301 | fastify |
| GET | linear | O(n) | 100 | 50 | 5000 | 1795.43 | 1775.761 | 1909.564 | 1542.747 | 2903.373 | 2213.512 | elysia |
| GET | linear | O(n) | 100 | 50 | 10000 | 1810.256 | 1716.721 | 2014.489 | 1706.338 | 3021.902 | 2180.548 | elysia |
| GET | linear | O(n) | 500 | 50 | 50 | 663.52 | 757.298 | 724.674 | 729.824 | 1269.141 | 837.155 | elysia |
| GET | linear | O(n) | 500 | 50 | 500 | 693.65 | 716.839 | 726.751 | 710.659 | 1298.032 | 806.241 | elysia |
| GET | linear | O(n) | 500 | 50 | 5000 | 710.758 | 703.262 | 761.736 | 1262.953 | 1123.551 | 945.178 | nest |
| GET | linear | O(n) | 500 | 50 | 10000 | 660.721 | 664.685 | 770.474 | 649.439 | 1291.272 | 790.397 | elysia |
| GET | linear | O(n) | 1000 | 50 | 50 | 653.108 | 412.419 | 382.593 | 722.14 | 518.556 | 439.473 | nest |
| GET | linear | O(n) | 1000 | 50 | 500 | 952.066 | 726.267 | 954.181 | 1244.827 | 755.149 | 1421.032 | fastify |
| GET | linear | O(n) | 1000 | 50 | 5000 | 690.667 | 445.541 | 814.776 | 434.799 | 865.071 | 643.084 | elysia |
| GET | linear | O(n) | 1000 | 50 | 10000 | 524.645 | 382.52 | 770.877 | 653.199 | 812.042 | 497.816 | elysia |
| POST | linear | O(n) | 10 | 50 | 50 | 657.047 | 646 | 900.245 | 659.773 | 365.167 | 431.533 | koa |
| POST | linear | O(n) | 10 | 50 | 500 | 1664 | 1469.607 | 1904.15 | 1910.254 | 714.249 | 701.165 | nest |
| POST | linear | O(n) | 10 | 50 | 5000 | 924.124 | 878.963 | 1890.071 | 1262.316 | 728.014 | 683.527 | koa |
| POST | linear | O(n) | 10 | 50 | 10000 | 1566.004 | 1350.139 | 2118.72 | 1231.936 | 772.596 | 703.949 | koa |
| POST | linear | O(n) | 100 | 50 | 50 | 349.359 | 509.487 | 493.711 | 442.162 | 74.627 | 377.532 | hapi |
| POST | linear | O(n) | 100 | 50 | 500 | 181.924 | 248.937 | 266.088 | 145.359 | 115.165 | 415.93 | fastify |
| POST | linear | O(n) | 100 | 50 | 5000 | 348.085 | 324.625 | 295.98 | 232.5368 | 115.823 | 611.405 | fastify |
| POST | linear | O(n) | 100 | 50 | 10000 | 274.453 | 418.234 | 421.797 | 204.579 | 108.72 | 507.853 | fastify |
| POST | linear | O(n) | 500 | 50 | 50 | 128.084 | 186.314 | 188.251 | 110.497 | 21.811 | 191.549 | fastify |
| POST | linear | O(n) | 500 | 50 | 500 | 78.241 | 95.731 | 130.587 | 55.515 | 24.917 | 136.554 | fastify |
| POST | linear | O(n) | 500 | 50 | 5000 | 77.811 | 126.838 | 129.863 | 56.029 | 22.056 | 132.422 | fastify |
| POST | linear | O(n) | 500 | 50 | 10000 | 53.982 | 93.226 | 94.696 | 39.958 | 23.52 | 98.143 | fastify |
| POST | linear | O(n) | 1000 | 50 | 50 | 48.733 | 82.014 | 66.593 | 33.905 | 11.692 | 81.803 | hapi |
| POST | linear | O(n) | 1000 | 50 | 500 | 43.291 | 76.201 | 78.267 | 34.554 | 11.374 | 77.253 | koa |
| POST | linear | O(n) | 1000 | 50 | 5000 | 27.805 | 49.405 | 48.386 | 21.086 | 11.732 | 53.941 | fastify |
| POST | linear | O(n) | 1000 | 50 | 10000 | 487.283 | 444.384 | 847.807 | 273.584 | 270.645 | 272.232 | koa |
| PUT | linear | O(n) | 10 | 50 | 50 | 553.135 | 780.455 | 860.293 | 753.08 | 859.031 | 409.197 | koa |
| PUT | linear | O(n) | 10 | 50 | 500 | 1369.86 | 1810.872 | 1434.392 | 1307.467 | 2015.583 | 617.389 | elysia |
| PUT | linear | O(n) | 10 | 50 | 5000 | 823.182 | 1283.128 | 1596.548 | 1319.6 | 2049.5 | 620.143 | elysia |
| PUT | linear | O(n) | 10 | 50 | 10000 | 1600.769 | 1857.452 | 1714.344 | 1470.404 | 2253.683 | 651.793 | elysia |
| PUT | linear | O(n) | 100 | 50 | 50 | 620.535 | 773.856 | 688.877 | 646.206 | 892.745 | 464.923 | elysia |
| PUT | linear | O(n) | 100 | 50 | 500 | 1277.494 | 1235.78 | 1352.144 | 1099.425 | 421.379 | 1195.3 | koa |
| PUT | linear | O(n) | 100 | 50 | 5000 | 1443.885 | 1429.65 | 1570.361 | 1512.894 | 1874.221 | 1627.322 | elysia |
| PUT | linear | O(n) | 100 | 50 | 10000 | 1370.024 | 1368.549 | 1492.049 | 1468.19 | 1814.845 | 1570.225 | elysia |
| PUT | linear | O(n) | 500 | 50 | 50 | 534.089 | 493.066 | 513.528 | 550.66 | 671.208 | 552.55 | elysia |
| PUT | linear | O(n) | 500 | 50 | 500 | 530.3 | 576.058 | 575.723 | 580.042 | 701.251 | 561.683 | elysia |
| PUT | linear | O(n) | 500 | 50 | 5000 | 556.001 | 567.649 | 585.856 | 1067.745 | 629.466 | 673.957 | nest |
| PUT | linear | O(n) | 500 | 50 | 10000 | 543.983 | 530.667 | 574.242 | 541.926 | 701.193 | 578.834 | elysia |
| PUT | linear | O(n) | 1000 | 50 | 50 | 534.797 | 289.745 | 268.795 | 635.008 | 290.054 | 285.772 | nest |
| PUT | linear | O(n) | 1000 | 50 | 500 | 753.953 | 560.524 | 772.774 | 1100.606 | 358.193 | 1181.633 | fastify |
| PUT | linear | O(n) | 1000 | 50 | 5000 | 304.988 | 321.393 | 630.764 | 317.43 | 384.887 | 471.782 | koa |
| PUT | linear | O(n) | 1000 | 50 | 10000 | 734.657 | 652.699 | 921.449 | 353.082 | 359.991 | 319.874 | koa |
| DELETE | linear | O(n) | 10 | 50 | 50 | 675.401 | 756.685 | 841.366 | 776.558 | 884.848 | 449.647 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 500 | 1470.731 | 1907.738 | 1666.107 | 1400.199 | 2102.916 | 669.449 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 5000 | 891.499 | 1413.273 | 1763.357 | 1467.404 | 2231.346 | 654.362 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 10000 | 1756.208 | 2128.694 | 1921.712 | 1503.596 | 2417.831 | 672.847 | elysia |
| DELETE | linear | O(n) | 100 | 50 | 50 | 820.1 | 843.545 | 936.192 | 831.127 | 947.72 | 496.233 | elysia |
| DELETE | linear | O(n) | 100 | 50 | 500 | 3047.162 | 2370.245 | 3210.388 | 2290.032 | 488.577 | 3197.288 | koa |
| DELETE | linear | O(n) | 100 | 50 | 5000 | 3029.318 | 2359.429 | 3174.31 | 2265.247 | 488.6577 | 3162.44 | koa |
| DELETE | linear | O(n) | 100 | 50 | 10000 | 3165.969 | 2389.997 | 3924.773 | 2900.382 | 4733.479 | 4327.483 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 50 | 2894.093 | 2853.471 | 3224.312 | 2887.623 | 3948.987 | 3353.328 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 500 | 2738.712 | 2664.866 | 3549.013 | 2759.9 | 4211.031 | 3824.288 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 5000 | 2728.363 | 2274.163 | 3157.45 | 2565.875 | 4292.73 | 3605.446 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 10000 | 2799.374 | 2351.451 | 3282.025 | 2634.755 | 4586.693 | 3950.629 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 50 | 989.4 | 657.992 | 995.356 | 864.26 | 1663.187 | 1243.19 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 500 | 1646.867 | 1269.967 | 1757.576 | 1528.559 | 2995.356 | 2071.627 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 2702.896 | 2250.773 | 3083.503 | 2517.138 | 4451.336 | 3687.854 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 10000 | 4563.842 | 3105.609 | 5264.371 | 3259.72 | 5978.755 | 4948.348 | elysia |

**Tabel 4. 3** Hasil pengujian Time per Request

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | framework | | | | | | framework terbaik |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 38.4412 | 43.2053 | 36.2039 | 43.67211 | 17.7997 | 26.1025 | elysia |
| GET | konstan | O(1) | 0 | 50 | 500 | 17.3212 | 19.3713 | 14.9911 | 17.9211 | 9.771 | 13.4609 | elysia |
| GET | konstan | O(1) | 0 | 50 | 5000 | 10.357 | 11.086 | 10.812 | 11.156 | 9.388 | 9.973 | elysia |
| GET | konstan | O(1) | 0 | 50 | 10000 | 10.1215 | 10.7125 | 10.1475 | 11.5715 | 10.154 | 12.425 | express |
| GET | linear | O(n) | 10 | 50 | 50 | 105.318 | 67.5456 | 66.761 | 73.8352 | 62.7428 | 186.0135 | elysia |
| GET | linear | O(n) | 10 | 50 | 500 | 39.5487 | 27.1377 | 37.4591 | 38.1607 | 25.7405 | 81.7477 | elysia |
| GET | linear | O(n) | 10 | 50 | 5000 | 62.846 | 44.1885 | 32.7152 | 39.0642 | 23.3971 | 80.6588 | elysia |
| GET | linear | O(n) | 10 | 50 | 10000 | 33.2439 | 23.0671 | 32.7994 | 36.948 | 20.3429 | 81.5462 | elysia |
| GET | linear | O(n) | 100 | 50 | 50 | 98.6309 | 64.8966 | 72.5778 | 93.0828 | 62.4654 | 116.9006 | elysia |
| GET | linear | O(n) | 100 | 50 | 500 | 99.1975 | 32.8787 | 26.2459 | 30.9502 | 89.0745 | 24.3521 | fastify |
| GET | linear | O(n) | 100 | 50 | 5000 | 27.866 | 28.261 | 26.284 | 57.291 | 17.4121 | 22.601 | elysia |
| GET | linear | O(n) | 100 | 50 | 10000 | 27.625 | 29.2065 | 24.8335 | 29.31 | 16.6075 | 22.955 | elysia |
| GET | linear | O(n) | 500 | 50 | 50 | 76.8052 | 68.6053 | 70.1051 | 70.9051 | 40.3033 | 61.8043 | elysia |
| GET | linear | O(n) | 500 | 50 | 500 | 72.4473 | 70.0251 | 69.0452 | 70.5151 | 38.6629 | 62.2742 | elysia |
| GET | linear | O(n) | 500 | 50 | 5000 | 70.3672 | 71.1011 | 65.6867 | 39.6025 | 44.6024 | 53.013 | nest |
| GET | linear | O(n) | 500 | 50 | 10000 | 75.7498 | 75.2345 | 64.911 | 77.1082 | 38.7245 | 63.2659 | elysia |
| GET | linear | O(n) | 1000 | 50 | 50 | 76.7079 | 122.0096 | 130.81 | 69.3054 | 96.5071 | 113.9096 | nest |
| GET | linear | O(n) | 1000 | 50 | 500 | 52.524 | 68.8952 | 52.9342 | 40.173 | 66.225 | 35.1928 | fastify |
| GET | linear | O(n) | 1000 | 50 | 5000 | 72.5674 | 112.2572 | 61.409 | 115.079 | 57.805 | 77.8013 | elysia |
| GET | linear | O(n) | 1000 | 50 | 10000 | 95.3927 | 130.9614 | 64.9716 | 76.6789 | 61.6145 | 100.4641 | elysia |
| POST | linear | O(n) | 10 | 50 | 50 | 86.1366 | 88.5044 | 60.3006 | 79.6498 | 170.7597 | 120.69 | koa |
| POST | linear | O(n) | 10 | 50 | 500 | 31.0675 | 34.9771 | 27.7913 | 28.8426 | 71.6283 | 72.0851 | koa |
| POST | linear | O(n) | 10 | 50 | 5000 | 54.3948 | 61.8403 | 26.5122 | 39.802 | 69.0725 | 73.5266 | koa |
| POST | linear | O(n) | 10 | 50 | 10000 | 32.0431 | 38.5832 | 23.7667 | 41.7447 | 64.7914 | 71.6215 | koa |
| POST | linear | O(n) | 100 | 50 | 50 | 181.7176 | 100.9029 | 107.4014 | 116.1545 | 713.6845 | 136.4966 | hapi |
| POST | linear | O(n) | 100 | 50 | 500 | 287.174 | 251.5397 | 199.1591 | 391.54 | 435.004 | 127.6952 | fastify |
| POST | linear | O(n) | 100 | 50 | 5000 | 149.1268 | 162.1789 | 183.9801 | 183.2494 | 433.7731 | 82.7141 | fastify |
| POST | linear | O(n) | 100 | 50 | 10000 | 196.4929 | 125.8804 | 124.2567 | 262.3261 | 462.2548 | 111.0282 | fastify |
| POST | linear | O(n) | 500 | 50 | 50 | 575.6409 | 401.8286 | 345.6242 | 691.7527 | 2310.414 | 346.5254 | koa |
| POST | linear | O(n) | 500 | 50 | 500 | 647.0233 | 554.3109 | 393.989 | 966.9937 | 2011.374 | 375.6885 | fastify |
| POST | linear | O(n) | 500 | 50 | 5000 | 642.8435 | 400.7808 | 385.3544 | 894.3092 | 2283.874 | 380.484 | fastify |
| POST | linear | O(n) | 500 | 50 | 10000 | 928.128 | 538.5012 | 529.4286 | 1255.139 | 2132.165 | 512.1354 | fastify |
| POST | linear | O(n) | 1000 | 50 | 50 | 1031.276 | 611.745 | 764.6567 | 1665.316 | 4289.632 | 617.2457 | hapi |
| POST | linear | O(n) | 1000 | 50 | 500 | 1158.053 | 664.8338 | 640.8563 | 1448.137 | 4402.927 | 655.2285 | koa |
| POST | linear | O(n) | 1000 | 50 | 5000 | 1814.217 | 1020.725 | 1037.444 | 2374.861 | 4274.555 | 929.3359 | fastify |
| POST | linear | O(n) | 1000 | 50 | 10000 | 102.7927 | 112.9651 | 59.045 | 182.7687 | 184.7844 | 183.6835 | koa |
| PUT | linear | O(n) | 10 | 50 | 50 | 129.7951 | 65.8052 | 58.8989 | 67.8829 | 62.0177 | 143.7095 | koa |
| PUT | linear | O(n) | 10 | 50 | 500 | 36.5292 | 27.7576 | 35.032 | 38.2723 | 24.8244 | 81.2457 | elysia |
| PUT | linear | O(n) | 10 | 50 | 5000 | 60.9041 | 44.3629 | 31.3988 | 38.7778 | 24.4953 | 80.7824 | elysia |
| PUT | linear | O(n) | 10 | 50 | 10000 | 31.3003 | 28.0546 | 29.2702 | 34.258 | 22.3128 | 76.9392 | elysia |
| PUT | linear | O(n) | 100 | 50 | 50 | 79.0424 | 66.6004 | 72.8989 | 100.3804 | 56.2525 | 108.843 | elysia |
| PUT | linear | O(n) | 100 | 50 | 500 | 39.173 | 40.8273 | 37.1272 | 45.4932 | 118.7088 | 41.9731 | koa |
| PUT | linear | O(n) | 100 | 50 | 5000 | 34.6337 | 35.013 | 31.842 | 33.0512 | 26.6878 | 30.726 | elysia |
| PUT | linear | O(n) | 100 | 50 | 10000 | 36.499 | 36.6716 | 33.5242 | 34.0618 | 27.5515 | 31.843 | elysia |
| PUT | linear | O(n) | 500 | 50 | 50 | 93.7065 | 102.3077 | 99.9749 | 91.1054 | 74.6049 | 90.5392 | elysia |
| PUT | linear | O(n) | 500 | 50 | 500 | 94.2981 | 86.8164 | 86.8763 | 86.2063 | 71.3151 | 89.0696 | elysia |
| PUT | linear | O(n) | 500 | 50 | 5000 | 89.9334 | 88.0987 | 85.346 | 46.8364 | 79.6727 | 74.1894 | nest |
| PUT | linear | O(n) | 500 | 50 | 10000 | 91.9194 | 96.9815 | 87.0721 | 92.7984 | 71.317 | 86.3905 | elysia |
| PUT | linear | O(n) | 1000 | 50 | 50 | 93.6086 | 175.0182 | 186.0571 | 78.8064 | 172.4136 | 176.2132 | nest |
| PUT | linear | O(n) | 1000 | 50 | 500 | 66.3215 | 89.2167 | 64.705 | 45.4334 | 139.5914 | 42.6831 | fastify |
| PUT | linear | O(n) | 1000 | 50 | 5000 | 163.9488 | 155.5785 | 79.283 | 157.5172 | 129.9125 | 105.9835 | koa |
| PUT | linear | O(n) | 1000 | 50 | 10000 | 68.2054 | 76.983 | 54.2824 | 141.6429 | 139.8597 | 156.3331 | koa |
| DELETE | linear | O(n) | 10 | 50 | 50 | 75.9143 | 99.6234 | 63.5773 | 66.2266 | 61.523 | 118.4625 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 500 | 34.1147 | 26.2941 | 30.0777 | 35.4543 | 23.7967 | 74.93 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 5000 | 56.2209 | 40.1979 | 53.6917 | 34.277 | 22.6129 | 76.7028 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 10000 | 28.5116 | 24.1553 | 26.1514 | 33.8942 | 20.842 | 74.6371 | elysia |
| DELETE | linear | O(n) | 100 | 50 | 50 | 64.3611 | 59.8193 | 53.5987 | 60.7229 | 60.7462 | 100.995 | koa |
| DELETE | linear | O(n) | 100 | 50 | 500 | 16.4564 | 21.4379 | 15.7836 | 22.0817 | 102.3776 | 15.6935 | fastify |
| DELETE | linear | O(n) | 100 | 50 | 5000 | 16.0308 | 19.8084 | 12.923 | 17.248 | 10.7483 | 11.905 | elysia |
| DELETE | linear | O(n) | 100 | 50 | 10000 | 15.7935 | 20.9637 | 12.7475 | 17.2395 | 10.564 | 11.5605 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 50 | 17.5015 | 17.6022 | 15.8011 | 17.4013 | 12.9009 | 15.1 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 500 | 18.3659 | 18.8013 | 14.1181 | 18.1614 | 11.8962 | 13.111 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 5000 | 18.3319 | 21.987 | 15.836 | 19.4866 | 11.68 | 13.869 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 10000 | 17.8624 | 21.2735 | 15.2359 | 19.0016 | 10.9015 | 12.657 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 50 | 50.6694 | 76.5077 | 50.3049 | 58.0059 | 30.2035 | 40.3037 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 500 | 30.3976 | 39.373 | 28.452 | 32.7171 | 16.7111 | 24.152 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 18.499 | 22.2166 | 16.2171 | 19.8865 | 11.2339 | 13.559 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 10000 | 11.0073 | 16.4222 | 9.6818 | 15.5053 | 8.5012 | 10.2623 | elysia |

**Tabel 4. 4** Hasil pengujian Transfer rate

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | framework | | | | | | framework terbaik |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 245.335 | 199.763 | 173.01 | 197.266 | 352.621 | 239.988 | elysia |
| GET | konstan | O(1) | 0 | 50 | 500 | 555.881 | 448.092 | 417.796 | 520.757 | 634.816 | 472.234 | elysia |
| GET | konstan | O(1) | 0 | 50 | 5000 | 900.871 | 767.622 | 580.65 | 836.125 | 662.927 | 626.848 | express |
| GET | konstan | O(1) | 0 | 50 | 10000 | 921.686 | 794.372 | 616.051 | 811.681 | 613.963 | 505.927 | express |
| GET | linear | O(n) | 10 | 50 | 50 | 257.345 | 195.263 | 358.47 | 352.11 | 365.051 | 162.08 | elysia |
| GET | linear | O(n) | 10 | 50 | 500 | 647.626 | 477.862 | 597.63 | 358.732 | 866.602 | 273.03 | elysia |
| GET | linear | O(n) | 10 | 50 | 5000 | 406.451 | 311.606 | 680.3 | 651.651 | 951.59 | 275.424 | elysia |
| GET | linear | O(n) | 10 | 50 | 10000 | 707.906 | 561.541 | 682.08 | 694.301 | 1092.478 | 274.393 | elysia |
| GET | linear | O(n) | 100 | 50 | 50 | 1726.378 | 199.821 | 2375.468 | 2025.617 | 2839.28 | 1469.157 | elysia |
| GET | linear | O(n) | 100 | 50 | 500 | 1763.593 | 5333.071 | 6560.576 | 5638.68 | 120.656 | 7066.593 | fastify |
| GET | linear | O(n) | 100 | 50 | 5000 | 6268.227 | 6204.758 | 6543.612 | 5383.045 | 9937.812 | 7576.52 | elysia |
| GET | linear | O(n) | 100 | 50 | 10000 | 6305.842 | 5988.407 | 6909.068 | 5957.183 | 10328.77 | 7463.696 | elysia |
| GET | linear | O(n) | 500 | 50 | 50 | 11392.75 | 12965.77 | 12343.53 | 12497.3 | 21631.15 | 14261.96 | elysia |
| GET | linear | O(n) | 500 | 50 | 500 | 11877.74 | 12263.96 | 12378.91 | 12166.56 | 22121.07 | 13754.92 | elysia |
| GET | linear | O(n) | 500 | 50 | 5000 | 12170.34 | 12031.68 | 12974.8 | 8279.497 | 19140.95 | 13438.34 | elysia |
| GET | linear | O(n) | 500 | 50 | 10000 | 11315.49 | 11387.27 | 13123.62 | 11114.03 | 22010.87 | 13476.88 | elysia |
| GET | linear | O(n) | 1000 | 50 | 50 | 3323.58 | 6864.134 | 9355.239 | 1303.233 | 17672.95 | 10360.67 | elysia |
| GET | linear | O(n) | 1000 | 50 | 500 | 6308.364 | 9287.563 | 6943.899 | 2281.777 | 25735.59 | 4118.777 | elysia |
| GET | linear | O(n) | 1000 | 50 | 5000 | 11822.97 | 14595.37 | 12391.9 | 14839.66 | 29463.92 | 14162.31 | elysia |
| GET | linear | O(n) | 1000 | 50 | 10000 | 17913.23 | 13055.41 | 26276.12 | 22304.45 | 27668.11 | 16969.46 | elysia |
| POST | linear | O(n) | 10 | 50 | 50 | 148.861 | 121.125 | 146.816 | 149.481 | 26.747 | 70.375 | nest |
| POST | linear | O(n) | 10 | 50 | 500 | 377.001 | 275.554 | 310.54 | 491.241 | 52.315 | 114.35 | nest |
| POST | linear | O(n) | 10 | 50 | 5000 | 209.372 | 164.807 | 308.245 | 285.993 | 55.06 | 111.473 | koa |
| POST | linear | O(n) | 10 | 50 | 10000 | 354.797 | 221.24 | 345.533 | 279.11 | 58.256 | 114.806 | express |
| POST | linear | O(n) | 100 | 50 | 50 | 79.152 | 95.53 | 80.518 | 100.177 | 5.466 | 61.571 | nest |
| POST | linear | O(n) | 100 | 50 | 500 | 41.216 | 46.675 | 43.395 | 32.931 | 24.742 | 67.833 | fastify |
| POST | linear | O(n) | 100 | 50 | 5000 | 78.863 | 60.867 | 48.272 | 62.713 | 18.21 | 99.71 | fastify |
| POST | linear | O(n) | 100 | 50 | 10000 | 62.18 | 78.417 | 68.789 | 46.352 | 17.092 | 82.823 | fastify |
| POST | linear | O(n) | 500 | 50 | 50 | 29.019 | 34.934 | 30.701 | 25.034 | 3.429 | 31.237 | hapi |
| POST | linear | O(n) | 500 | 50 | 500 | 17.726 | 17.95 | 21.297 | 12.577 | 3.918 | 22.269 | fastify |
| POST | linear | O(n) | 500 | 50 | 5000 | 17.63 | 23.782 | 21.179 | 12.695 | 3.467 | 21.595 | hapi |
| POST | linear | O(n) | 500 | 50 | 10000 | 12.228 | 17.48 | 15.444 | 9.054 | 3.696 | 16.006 | hapi |
| POST | linear | O(n) | 1000 | 50 | 50 | 11.042 | 15.378 | 10.86 | 7.681 | 1.839 | 13.34 | hapi |
| POST | linear | O(n) | 1000 | 50 | 500 | 9.807 | 14.288 | 12.765 | 7.829 | 1.788 | 12.597 | hapi |
| POST | linear | O(n) | 1000 | 50 | 5000 | 6.299 | 9.263 | 7.891 | 4.775 | 1.843 | 8.796 | hapi |
| POST | linear | O(n) | 1000 | 50 | 10000 | 118.492 | 90.699 | 144.889 | 61.984 | 42.553 | 44.396 | koa |
| PUT | linear | O(n) | 10 | 50 | 50 | 122.403 | 142.525 | 136.101 | 166.942 | 135.062 | 64.906 | nest |
| PUT | linear | O(n) | 10 | 50 | 500 | 303.67 | 330.696 | 226.925 | 301.461 | 316.904 | 97.672 | hapi |
| PUT | linear | O(n) | 10 | 50 | 5000 | 182.483 | 234.321 | 252.579 | 292.53 | 322.235 | 98.109 | elysia |
| PUT | linear | O(n) | 10 | 50 | 10000 | 354.858 | 339.202 | 271.215 | 325.957 | 354.338 | 103.116 | express |
| PUT | linear | O(n) | 100 | 50 | 50 | 137.559 | 141.32 | 108.983 | 150.7129 | 140.363 | 73.553 | nest |
| PUT | linear | O(n) | 100 | 50 | 500 | 283.195 | 225.675 | 213.913 | 243.72 | 90.531 | 189.101 | express |
| PUT | linear | O(n) | 100 | 50 | 5000 | 320.078 | 261.078 | 248.436 | 335.378 | 294.677 | 257.446 | nest |
| PUT | linear | O(n) | 100 | 50 | 10000 | 245.335 | 199.763 | 173.01 | 197.266 | 352.621 | 239.988 | elysia |
| PUT | linear | O(n) | 500 | 50 | 50 | 118.397 | 90.041 | 81.241 | 122.071 | 105.531 | 87.414 | nest |
| PUT | linear | O(n) | 500 | 50 | 500 | 117.558 | 105.197 | 91.082 | 128.584 | 110.254 | 88.86 | nest |
| PUT | linear | O(n) | 500 | 50 | 5000 | 123.254 | 103.664 | 92.683 | 236.697 | 98.969 | 106.622 | nest |
| PUT | linear | O(n) | 500 | 50 | 10000 | 120.591 | 96.911 | 90.846 | 120.136 | 110.247 | 91.574 | express |
| PUT | linear | O(n) | 1000 | 50 | 50 | 118.553 | 52.912 | 42.526 | 140.769 | 45.605 | 45.212 | nest |
| PUT | linear | O(n) | 1000 | 50 | 500 | 167.135 | 102.361 | 122.255 | 243.981 | 56.318 | 186.938 | nest |
| PUT | linear | O(n) | 1000 | 50 | 5000 | 67.609 | 58.692 | 99.788 | 70.368 | 60.514 | 74.637 | koa |
| PUT | linear | O(n) | 1000 | 50 | 10000 | 175.055 | 130.029 | 152.975 | 78.27 | 56.599 | 50.607 | express |
| DELETE | linear | O(n) | 10 | 50 | 50 | 149.723 | 138.184 | 133.107 | 172.148 | 139.122 | 71.428 | nest |
| DELETE | linear | O(n) | 10 | 50 | 500 | 326.03 | 348.385 | 263.582 | 311.998 | 330.635 | 105.908 | hapi |
| DELETE | linear | O(n) | 10 | 50 | 5000 | 197.628 | 258.087 | 278.97 | 325.294 | 350.827 | 103.521 | elysia |
| DELETE | linear | O(n) | 10 | 50 | 10000 | 389.315 | 388.736 | 304.02 | 333.318 | 380.148 | 106.447 | express |
| DELETE | linear | O(n) | 100 | 50 | 50 | 181.801 | 154.045 | 148.109 | 184.244 | 149.008 | 78.503 | nest |
| DELETE | linear | O(n) | 100 | 50 | 500 | 675.493 | 432.849 | 507.893 | 507.656 | 104.967 | 505.822 | express |
| DELETE | linear | O(n) | 100 | 50 | 5000 | 691.654 | 463.222 | 612.21 | 642.736 | 731.52 | 664.993 | elysia |
| DELETE | linear | O(n) | 100 | 50 | 10000 | 701.832 | 436.455 | 620.912 | 642.956 | 744.23 | 684.621 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 50 | 641.561 | 521.092 | 510.096 | 640.129 | 620.885 | 530.505 | express |
| DELETE | linear | O(n) | 500 | 50 | 500 | 607.116 | 486.651 | 561.464 | 611.814 | 662.087 | 605.016 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 5000 | 604.823 | 415.3 | 499.518 | 568.801 | 674.931 | 570.391 | elysia |
| DELETE | linear | O(n) | 500 | 50 | 10000 | 620.565 | 429.415 | 519.227 | 584.071 | 721.149 | 625.003 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 50 | 219.33 | 120.161 | 157.469 | 191.589 | 261.495 | 196.677 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 500 | 349.614 | 218.822 | 266.049 | 323.037 | 451.712 | 314.137 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 599.176 | 411.029 | 487.819 | 557.997 | 699.868 | 583.429 | elysia |
| DELETE | linear | O(n) | 1000 | 50 | 10000 | 1011.711 | 567.139 | 868.827 | 722.615 | 940.017 | 782.844 | express |

Dari hasil Tabel 4.1 sampai Tabel 4.4 diatas untuk menampilkan *framework* mana yang unggul, maka menggunakan frekuensi untuk menetukannya. Dibawah ini adalah tabel menampilkan frekuensi berdasarkan jumlah data yang diujikan dimulai dari 10 data, 100 data, 500 data, 1000 data dan juga menampilkan frekuensi berdasarkan jumlah permintaan (n) sebesar 50, 500, 5000 dan 10000 dengan *concurrency* (c) sebesar 50. Dapat dilihat dari Tabel 4.5 dan Tabel 4.6.

**Tabel 4. 5** Frekuensi Performa Framework Backend JavaScript Berdasarkan Jumlah Data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| framework | express | hapi | koa | nest | elysia | fastify | Total frekuensi |
| Frekuensi terbaik 0 data | 4 | 0 | 0 | 0 | 12 | 0 | 16 |
| Frekuensi terbaik 10 data | 2 | 1 | 15 | 4 | 38 | 0 | 60 |
| Frekuensi terbaik 100 data | 2 | 3 | 8 | 4 | 30 | 21 | 68 |
| Frekuensi terbaik 500 data | 2 | 3 | 2 | 9 | 37 | 11 | 64 |
| frekuensi terbaik1000 data | 2 | 6 | 14 | 8 | 25 | 9 | 64 |
| Total poin frekuensi *framework* | 12 | 13 | 39 | 25 | 142 | 41 | 272 |

Berdasarkan Tabel 4.5 *Elysia* mendominasi dengan 142 poin, menunjukkan performa terbaik pada semua jumlah data, terutama pada 10, 100, dan 500 data. *Fastify* menempati posisi kedua dengan 41 poin, tampil baik pada 100 dan 500 data. *Koa* berada di posisi ketiga dengan 39 poin, dengan performa unggul pada pengujian 10 dan 1000 data. *Nest* berada di posisi keempat dengan total 25 poin, menunjukkan performa yang cukup baik pada 500 data. *Hapi* dan *Express* berada di posisi terbawah, dengan masing-masing 13 dan 12 poin, *Hapi* menonjol pada pengujian 1000 data, sementara *Express* memiliki performa terbaik pada 0 data.

**Tabel 4. 6** Frekuensi Performa Framework Backend JavaScript Berdasarkan jumlah permintaan dan concurrency

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| n (users) | c | express | hapi | koa | nest | elysia | fastify |
| 50 | 50 | 0 | 2 | 10 | 13 | 37 | 6 |
| 500 | 50 | 0 | 0 | 5 | 14 | 41 | 8 |
| 5000 | 50 | 0 | 0 | 4 | 14 | 26 | 24 |
| 10000 | 50 | 0 | 0 | 12 | 7 | 44 | 5 |
| Total | | 0 | 0 | 2 | 31 | 48 | 148 |

Tabel 4.6 menunjukkan frekuensi performa dari enam framework JavaScript (Express, Hapi, Koa, Nest, Elysia, dan Fastify) berdasarkan jumlah permintaan (n) sebesar 50, 500, 5000, dan 10000, dengan concurrency (c) yang sama sebesar 50. Dari tabel tersebut, terlihat bahwa Elysia memiliki frekuensi performa tertinggi yaitu sebesar 148 poin, diikuti oleh Nest dengan 48 poin, Fastify dengan 43 poin, dan Koa dengan 31 poin. Hapi hanya mengumpulkan 2 poin, sementara Express tidak memiliki poin sama sekali. Hal ini menunjukkan bahwa Elysia secara konsisten memberikan hasil performa terbaik di berbagai skenario pengujian, sedangkan Express menunjukkan performa paling rendah. Warna pada tabel juga memberikan visualisasi kualitas performa, di mana warna hijau menunjukkan performa terbaik, kuning menengah, dan merah terendah

Pengujian selanjutnya menggunakan *weight score*, di mana empat parameter utama diberikan bobot yang berbeda: *time taken for test* dengan bobot 25%, *requests per second* dengan bobot 40%, *time per request* dengan bobot 20%, dan *transfer rate* dengan bobot 15%. Dengan menggunakan bobot ini, hasil pengujian menunjukkan framework mana yang lebih unggul dalam menangani berbagai beban permintaan. Hasilnya dapat dilihat pada Tabel 4.7.

**Tabel 4. 7** Hasil Weight Score

| metode http | skenario | kompleksitas | jumlah data | c | n (users) | weight score | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 0.293 | 0.264 | 0.229 | 0.470 | 0.550 | 0.320 |
| GET | konstan | O(1) | 0 | 50 | 500 | 0.505 | 0.471 | 0.358 | 0.478 | 0.550 | 0.394 |
| GET | konstan | O(1) | 0 | 50 | 5000 | 0.558 | 0.536 | 0.438 | 0.570 | 0.439 | 0.417 |
| GET | konstan | O(1) | 0 | 50 | 10000 | 0.546 | 0.497 | 0.435 | 0.528 | 0.445 | 0.450 |
| GET | linear | O(n) | 10 | 50 | 50 | 0.370 | 0.386 | 0.546 | 0.466 | 0.550 | 0.450 |
| GET | linear | O(n) | 10 | 50 | 500 | 0.403 | 0.435 | 0.395 | 0.333 | 0.550 | 0.450 |
| GET | linear | O(n) | 10 | 50 | 5000 | 0.386 | 0.327 | 0.402 | 0.380 | 0.550 | 0.450 |
| GET | linear | O(n) | 10 | 50 | 10000 | 0.367 | 0.412 | 0.365 | 0.361 | 0.550 | 0.450 |
| GET | linear | O(n) | 100 | 50 | 50 | 0.510 | 0.369 | 0.472 | 0.508 | 0.550 | 0.522 |
| GET | linear | O(n) | 100 | 50 | 500 | 0.485 | 0.426 | 0.512 | 0.444 | 0.405 | 0.550 |
| GET | linear | O(n) | 100 | 50 | 5000 | 0.221 | 0.218 | 0.246 | 0.450 | 0.550 | 0.328 |
| GET | linear | O(n) | 100 | 50 | 10000 | 0.434 | 0.451 | 0.418 | 0.450 | 0.550 | 0.421 |
| GET | linear | O(n) | 500 | 50 | 50 | 0.450 | 0.434 | 0.422 | 0.437 | 0.550 | 0.422 |
| GET | linear | O(n) | 500 | 50 | 500 | 0.450 | 0.439 | 0.434 | 0.440 | 0.550 | 0.417 |
| GET | linear | O(n) | 500 | 50 | 5000 | 0.499 | 0.502 | 0.479 | 0.400 | 0.522 | 0.436 |
| GET | linear | O(n) | 500 | 50 | 10000 | 0.444 | 0.441 | 0.410 | 0.450 | 0.550 | 0.408 |
| GET | linear | O(n) | 1000 | 50 | 50 | 0.391 | 0.472 | 0.524 | 0.400 | 0.509 | 0.476 |
| GET | linear | O(n) | 1000 | 50 | 500 | 0.387 | 0.495 | 0.398 | 0.365 | 0.581 | 0.412 |
| GET | linear | O(n) | 1000 | 50 | 5000 | 0.354 | 0.461 | 0.386 | 0.476 | 0.550 | 0.371 |
| GET | linear | O(n) | 1000 | 50 | 10000 | 0.401 | 0.450 | 0.519 | 0.445 | 0.550 | 0.400 |
| POST | linear | O(n) | 10 | 50 | 50 | 0.473 | 0.440 | 0.547 | 0.449 | 0.450 | 0.349 |
| POST | linear | O(n) | 10 | 50 | 500 | 0.463 | 0.403 | 0.486 | 0.561 | 0.450 | 0.471 |
| POST | linear | O(n) | 10 | 50 | 5000 | 0.438 | 0.468 | 0.550 | 0.456 | 0.422 | 0.483 |
| POST | linear | O(n) | 10 | 50 | 10000 | 0.472 | 0.404 | 0.545 | 0.430 | 0.405 | 0.479 |
| POST | linear | O(n) | 100 | 50 | 50 | 0.429 | 0.543 | 0.509 | 0.499 | 0.450 | 0.394 |
| POST | linear | O(n) | 100 | 50 | 500 | 0.380 | 0.436 | 0.370 | 0.455 | 0.450 | 0.550 |
| POST | linear | O(n) | 100 | 50 | 5000 | 0.384 | 0.349 | 0.331 | 0.305 | 0.450 | 0.550 |
| POST | linear | O(n) | 100 | 50 | 10000 | 0.378 | 0.469 | 0.449 | 0.357 | 0.450 | 0.550 |
| POST | linear | O(n) | 500 | 50 | 50 | 0.425 | 0.551 | 0.522 | 0.391 | 0.450 | 0.533 |
| POST | linear | O(n) | 500 | 50 | 500 | 0.379 | 0.418 | 0.526 | 0.343 | 0.450 | 0.550 |
| POST | linear | O(n) | 500 | 50 | 5000 | 0.369 | 0.535 | 0.523 | 0.313 | 0.450 | 0.534 |
| POST | linear | O(n) | 500 | 50 | 10000 | 0.372 | 0.531 | 0.514 | 0.353 | 0.450 | 0.534 |
| POST | linear | O(n) | 1000 | 50 | 50 | 0.364 | 0.550 | 0.431 | 0.320 | 0.450 | 0.527 |
| POST | linear | O(n) | 1000 | 50 | 500 | 0.349 | 0.541 | 0.532 | 0.308 | 0.450 | 0.525 |
| POST | linear | O(n) | 1000 | 50 | 5000 | 0.361 | 0.519 | 0.484 | 0.342 | 0.450 | 0.541 |
| POST | linear | O(n) | 1000 | 50 | 10000 | 0.418 | 0.384 | 0.550 | 0.473 | 0.450 | 0.450 |
| PUT | linear | O(n) | 10 | 50 | 50 | 0.588 | 0.480 | 0.505 | 0.502 | 0.518 | 0.450 |
| PUT | linear | O(n) | 10 | 50 | 500 | 0.441 | 0.515 | 0.398 | 0.436 | 0.541 | 0.450 |
| PUT | linear | O(n) | 10 | 50 | 5000 | 0.404 | 0.436 | 0.432 | 0.440 | 0.550 | 0.450 |
| PUT | linear | O(n) | 10 | 50 | 10000 | 0.461 | 0.489 | 0.423 | 0.436 | 0.550 | 0.450 |
| PUT | linear | O(n) | 100 | 50 | 50 | 0.481 | 0.509 | 0.421 | 0.609 | 0.530 | 0.450 |
| PUT | linear | O(n) | 100 | 50 | 500 | 0.529 | 0.476 | 0.496 | 0.457 | 0.450 | 0.436 |
| PUT | linear | O(n) | 100 | 50 | 5000 | 0.566 | 0.472 | 0.405 | 0.569 | 0.480 | 0.412 |
| PUT | linear | O(n) | 100 | 50 | 10000 | 0.503 | 0.472 | 0.405 | 0.431 | 0.550 | 0.448 |
| PUT | linear | O(n) | 500 | 50 | 50 | 0.539 | 0.482 | 0.458 | 0.547 | 0.489 | 0.415 |
| PUT | linear | O(n) | 500 | 50 | 500 | 0.558 | 0.472 | 0.419 | 0.558 | 0.481 | 0.421 |
| PUT | linear | O(n) | 500 | 50 | 5000 | 0.482 | 0.451 | 0.425 | 0.550 | 0.407 | 0.392 |
| PUT | linear | O(n) | 500 | 50 | 10000 | 0.542 | 0.481 | 0.378 | 0.551 | 0.498 | 0.381 |
| PUT | linear | O(n) | 1000 | 50 | 50 | 0.469 | 0.443 | 0.450 | 0.550 | 0.421 | 0.431 |
| PUT | linear | O(n) | 1000 | 50 | 500 | 0.391 | 0.351 | 0.356 | 0.523 | 0.450 | 0.504 |
| PUT | linear | O(n) | 1000 | 50 | 5000 | 0.483 | 0.426 | 0.550 | 0.474 | 0.374 | 0.405 |
| PUT | linear | O(n) | 1000 | 50 | 10000 | 0.487 | 0.417 | 0.523 | 0.441 | 0.411 | 0.450 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 0.438 | 0.682 | 0.468 | 0.488 | 0.501 | 0.450 |
| DELETE | linear | O(n) | 10 | 50 | 500 | 0.450 | 0.518 | 0.431 | 0.435 | 0.539 | 0.450 |
| DELETE | linear | O(n) | 10 | 50 | 5000 | 0.397 | 0.433 | 0.529 | 0.438 | 0.550 | 0.450 |
| DELETE | linear | O(n) | 10 | 50 | 10000 | 0.462 | 0.511 | 0.435 | 0.420 | 0.545 | 0.450 |
| DELETE | linear | O(n) | 100 | 50 | 50 | 0.536 | 0.474 | 0.489 | 0.514 | 0.568 | 0.450 |
| DELETE | linear | O(n) | 100 | 50 | 500 | 0.530 | 0.393 | 0.506 | 0.404 | 0.450 | 0.503 |
| DELETE | linear | O(n) | 100 | 50 | 5000 | 0.769 | 0.729 | 0.591 | 0.688 | 0.150 | 0.568 |
| DELETE | linear | O(n) | 100 | 50 | 10000 | 0.488 | 0.450 | 0.446 | 0.477 | 0.550 | 0.495 |
| DELETE | linear | O(n) | 500 | 50 | 50 | 0.605 | 0.463 | 0.413 | 0.592 | 0.526 | 0.416 |
| DELETE | linear | O(n) | 500 | 50 | 500 | 0.544 | 0.450 | 0.437 | 0.540 | 0.550 | 0.480 |
| DELETE | linear | O(n) | 500 | 50 | 5000 | 0.490 | 0.450 | 0.405 | 0.487 | 0.550 | 0.449 |
| DELETE | linear | O(n) | 500 | 50 | 10000 | 0.480 | 0.450 | 0.401 | 0.482 | 0.550 | 0.463 |
| DELETE | linear | O(n) | 1000 | 50 | 50 | 0.436 | 0.450 | 0.369 | 0.428 | 0.550 | 0.412 |
| DELETE | linear | O(n) | 1000 | 50 | 500 | 0.443 | 0.450 | 0.377 | 0.445 | 0.550 | 0.395 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 0.478 | 0.450 | 0.395 | 0.479 | 0.550 | 0.446 |
| DELETE | linear | O(n) | 1000 | 50 | 10000 | 0.495 | 0.450 | 0.469 | 0.472 | 0.526 | 0.429 |

Berdasarkan Tabel 4.6, *Elysia* dan *Fastify* secara konsisten menunjukkan performa terbaik dalam berbagai skenario pengujian, terutama pada metode GET, PUT, dan DELETE dengan jumlah data besar dan jumlah request tinggi. Keduanya memiliki skor tertinggi dalam berbagai kombinasi jumlah data dan *request*, menandakan kemampuan mereka menangani beban permintaan dengan efisien. *Express* dan *Hapi* menunjukkan performa yang lebih fluktuatif, dengan hasil yang baik pada jumlah request rendah tetapi mengalami penurunan signifikan pada jumlah *request* yang lebih tinggi. *Koa* dan *Nest* umumnya memiliki skor lebih rendah, terutama pada skenario PUT dan DELETE dengan jumlah data besar, menunjukkan bahwa keduanya kurang efisien dalam menangani permintaan dengan beban tinggi.

### Mitata Js

Hasil pengujian performa enam *framework backend javascript* yang menggunakan *mitata js* untuk pengujiannya yaitu berdasarkan waktu dalam *seconds*. Hasil data mentahnya disajikan secara lengkap pada Lampiran 5.

Dibawah ini adalah hasil rata – rata dari pengujian yang dilakukan, bisa dilihat pada Tabel 4.7. Tabel ini menggunakan skema warna untuk memudahkan interpretasi hasil, di mana warna hijau menandakan performa terbaik dan warna merah menandakan performa terburuk.

**Tabel 4. 8** Hasil rata - rata pengujian mitata js

| metode http | skenario | kompleksitas | jumlah data | framework | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 682.11 µs | 869.68 µs | 866.57 µs | 919.31 µs | 715.95 µs | 770.63 µs |
| GET | linear | O(n) | 10 | 1.30 ms | 1.52 ms | 1.28 ms | 1.43 ms | 1.30 ms | 1.44 ms |
| GET | linear | O(n) | 100 | 1.51 ms | 4.97 ms | 1.48 ms | 1.68 ms | 1.49 ms | 1.61 ms |
| GET | linear | O(n) | 500 | 2.56 ms | 8.95 ms | 2.56 ms | 2.94 ms | 2.20 ms | 2.71 ms |
| GET | linear | O(n) | 1000 | 2.54 ms | 9.11 ms | 3.59 ms | 3.86 ms | 2.82 ms | 3.86 ms |
| POST | linear | O(n) | 10 | 1.33 ms | 2.24 ms | 1.39 ms | 1.58 ms | 2.53 ms | 1.45 ms |
| POST | linear | O(n) | 100 | 1.82 ms | 2.81 ms | 2.01 ms | 2.45 ms | 14.93 ms | 2.03 ms |
| POST | linear | O(n) | 500 | 4.84 ms | 8.74 ms | 4.80 ms | 6.90 ms | 60.66 ms | 5.23 ms |
| POST | linear | O(n) | 1000 | 7.36 ms | 11.35 ms | 6.58 ms | 10.78 ms | 112.38 ms | 8.47 ms |
| PUT | linear | O(n) | 10 | 1.78 ms | 2.17 ms | 1.79 ms | 2.03 ms | 1.61 ms | 2.27 ms |
| PUT | linear | O(n) | 100 | 2.00 ms | 2.67 ms | 2.17 ms | 2.29 ms | 1.87 ms | 2.15 ms |
| PUT | linear | O(n) | 500 | 3.09 ms | 4.30 ms | 3.11 ms | 3.42 ms | 2.89 ms | 3.70 ms |
| PUT | linear | O(n) | 1000 | 4.78 ms | 8.68 ms | 4.80 ms | 5.13 ms | 4.33 ms | 5.33 ms |
| DELETE | linear | O(n) | 10 | 1.10 ms | 1.45 ms | 1.16 ms | 1.38 ms | 1.14 ms | 1.26 ms |
| DELETE | linear | O(n) | 100 | 1.05 ms | 1.67 ms | 1.14 ms | 1.37 ms | 1.16 ms | 1.23 ms |
| DELETE | linear | O(n) | 500 | 1.08 ms | 1.90 ms | 1.02 ms | 1.25 ms | 1.03 ms | 1.16 ms |
| DELETE | linear | O(n) | 1000 | 1.04 ms | 1.71 ms | 1.05 ms | 1.31 ms | 1.13 ms | 1.17 ms |

Berdasarkan Tabel 4.7, Hasil menunjukkan bahwa *express* dan *fastify* secara konsisten memberikan performa terbaik, terutama pada skenario GET dan DELETE dengan waktu respon yang lebih cepat. *elysia* juga memberikan performa baik namun mengalami penurunan signifikan dalam POST dengan data besar. Sebaliknya, *hapi* dan *nest* cenderung lebih lambat, terutama saat menangani jumlah data besar pada metode PUT dan POST, sedangkan *koa* memberikan performa yang solid meski tidak selalu menjadi yang tercepat.

## Pembahasan

Berdasarkan semua hasil pengujian yang sudah dilakukan sebelumnya, maka dapat diambil kesimpulan sebagai berikut:

1. Pada pengujian *apache benchmark* dengan menggunakan frekuensi berdasarkan jumlah data , *Elysia* adalah *framework* dengan performa paling unggul dalam pengujian ini, mengumpulkan skor tertinggi dan menunjukkan kinerja terbaik pada berbagai ukuran data, terutama pada 10, 100, dan 500 data. *Fastify* berada di posisi kedua, diikuti oleh *Koa* dan *Nest*, yang masing-masing memiliki keunggulan di data tertentu. Sementara itu, *Hapi* dan *Express* berada di peringkat terbawah dengan skor terendah, di mana *Hapi* hanya menonjol pada data 1000, dan *Express* tidak menunjukkan performa menonjol pada data apapun.
2. Pada pengujian *apache benchmark* dengan menggunakan frekuensi berdasarkan jumlah permintaan (n) sebesar 50, 500, 5000, dan 10000 dengan concurrency (c) yang sama sebesar 50, *Elysia* adalah *framework* dengan performa tertinggi, mencapai 134 poin dan menunjukkan konsistensi terbaik di berbagai skenario pengujian. *Nest* dan *Fastify* mengikuti dengan selisih cukup signifikan, masing-masing 48 dan 43 poin.
3. Pada pengujian *apache benchmark* dengan menggunakan *weight scoring system*, *Elysia* dan *Fastify* yang paling menonjol sebagai *framework* dengan performa paling konsisten dan efisien dalam berbagai skenario pengujian, terutama pada metode GET, PUT, dan DELETE dengan jumlah data dan *request* yang besar. Hal ini menunjukkan kemampuan keduanya dalam menangani beban permintaan tinggi dengan baik.
4. Pada pengujian *mitata js,* Hasil pengujian menunjukkan bahwa *Express* dan *Fastify* konsisten memberikan performa terbaik, terutama pada skenario GET dan DELETE dengan waktu respons yang cepat.

# KESIMPULAN DAN SARAN

## Kesimpulan

Berdasarkan penelitian yang dilakukan , dapat disimpulkan bahwa:

1. Untuk melakukan pengujian performa 6 *framework backend javascript*(*express, hapi, koa, nest, elysia* dan *fastify)* di *runtime bun* dan menggunakan metode HTTP GET, POST, PUT dan DELETE,penelitian ini menggunakan 2 buah tools yaitu *apache benchmark* dan *mitata js,* Dimana untuk *tool apache benchmark* menggunakan beban *concurrency* sebesar 50 dan untuk *n* sebesar 50, 500, 5000 dan 10000 dengan menggunakan parameter *time taken for test, request per second, time per request* dan *transfer rate,* untuk skenario pada pengujian ini menggunakan *Big O Notation* dengan skenario konstan(O(1)) dan linear(O(n)). Penelitian ini juga menggunakan 1000 data yang disimpan dengan mysql.
2. Penelitian ini menunjukkan bahwa *framework elysia* dan *fastify* yang menunjukkan performa terbaik di semua 3 sudut pandang yang berbeda yaitu sudut pandang frekuensi terbaik berdasarkan jumlah data, sudut pandang frekuensi terbaik berdasarkan jumlah permintaan dan *concurrency*, sudut pandang penghitungan dengan *weight score* pada pengujian *apache benchmark* dan untuk di pengujian *mitata js, express js* dan *fastify js* konsisten memberikan performa terbaik. Jadi secara keseluruhan *framework elysia* dan *fastify* yang paling unggul.

## Saran

Saran dari penelitian yang akan dilakukan pada penelitian selanjutnya besar harapan guna bisa dikembangkan yakni :

1. Menambahkan *framework backend JavaScript* lainnya dapat memperkaya penelitian ini dengan memberikan perspektif baru pada performa *backend* dalam berbagai skenario pengujian. *Framework* tambahan, seperti *Sails.js, Feathers.js*, atau *Strapi*, yang membawa pendekatan unik dalam pengelolaan API, *query*, atau manajemen data *real-time*, akan membantu mengeksplorasi kinerja framework dalam menangani beban dan kompleksitas yang berbeda. Melalui variasi *framework* dengan arsitektur dan pendekatan berbeda, penelitian ini akan memperoleh wawasan lebih luas mengenai keunggulan dan kelemahan masing-masing *framework* saat digunakan dalam skenario kompleksitas yang beragam, mulai dari aplikasi ringan hingga aplikasi berskala besar.
2. Menggunakan *Node.js* sebagai *runtime* tambahan dalam pengujian dapat memperkaya analisis performa *framework backend JavaScript* dalam penelitian ini. *Node.js*, sebagai *runtime* yang populer dan stabil, akan memberikan perbandingan yang relevan terhadap *runtime Bun* yang digunakan dalam penelitian ini. Melalui pengujian dengan *Node.js*, akan terlihat bagaimana masing-masing framework beradaptasi terhadap perbedaan optimisasi, manajemen memori, dan efisiensi eksekusi antara kedua *runtime* tersebut.
3. Melakukan pengujian dengan *tools benchmark* lain dapat memberikan perspektif yang lebih luas dan mendalam mengenai performa *framework backend JavaScript* yang diuji. Dengan menggunakan alat *benchmark* tambahan seperti *JMeter, Artillery*, atau *K6*, penelitian ini dapat mengevaluasi performa setiap *framework* dari berbagai metrik, metode pengujian, dan simulasi beban yang berbeda.

# DAFTAR PUSTAKA

Ahmod, M. F. (2023). *JAVASCRIPT RUNTIME PERFORMANCE ANALYSIS: NODE AND BUN*.

Amarulloh, A. (2023). *ANALISIS PERBANDINGAN PERFORMA WEB SERVICE REST MENGGUNAKAN FRAMEWORK LARAVEL, DJANGO, DAN Node JS PADA APLIKASI BERBASIS WEBSITE*.

Apache.org. (2024). *ab - Apache HTTP server benchmarking tool - Apache HTTP Server Version 2.4*.

Azi, M. N. A., Arifwidodo, B., & Wahyudi, E. (2023). Analisis Performansi Web Server Saat Menangani Permintaan Client Menggunakan Metode Reserve Proxy Caching dan Varnish. *Journal of Telecommunication, Electronics, and Control Engineering (JTECE)*, *5*(1), 14–21. https://doi.org/10.20895/jtece.v5i1.843

Celi-Párraga, R. J., Boné-Andrade, M. F., & Olivero, A. P. M. (2023). *Programación Web del Frontend al Backend*. Editorial Grupo AEA. https://doi.org/10.55813/egaea.l.2022.18

Chastro, C., Darmawan, E., Kom, S., & #2, M. T. (2020). *Perbandingan Pengembangan Front End Menggunakan Blade Template dan Vue Js* (Vol. 2).

del Pilar Salas-Zárate, M., Alor-Hernández, G., Valencia-García, R., Rodríguez-Mazahua, L., Rodríguez-González, A., & Cuadrado, J. L. L. (2015). Analyzing best practices on Web development frameworks: The lift approach. *Science of Computer Programming*, *102*, 1–19. https://doi.org/10.1016/j.scico.2014.12.004

Dirjen, S. K., Riset, P., Pengembangan, D., Dikti, R., Putu, I., & Pratama, A. E. (2017). Terakreditasi SINTA Peringkat 2 Pengujian Performansi Lima Back-End JavaScript Framework Menggunakan Metode GET dan POST. *Jurnal Resti*, *1*(3), 1216–1225.

elysiajs. (2024). *Ergonomic Framework for Humans*.

Fastify. (2024). *Fast and low overhead web framework, for Node.js*.

Ginasari, N. L. A. S., Wibawa, K. S., & Wirdiani, N. K. A. (2021). Pengujian Stress Testing API Sistem Pelayanan dengan Apache JMeter. *JITTER : Jurnal Ilmiah Teknologi Dan Komputer*, *2*(3), 552. https://doi.org/10.24843/JTRTI.2021.v02.i03.p14

Hadinata, W., & Stianingsih, L. (2024). ANALISIS PERBANDINGAN PERFORMA RESTFULL API ANTARA EXPRESS.JS DENGAN LARAVEL FRAMEWORK. *Jurnal Informatika Dan Teknik Elektro Terapan*, *12*(1). https://doi.org/10.23960/jitet.v12i1.3845

Hapi. (2024). *The Simple, Secure Framework Developers Trust*.

Koa. (2024). *Koa: Next Generation Web Framework for NodeJS*.

Mulana, L., Prihandani, K., Rizal, A., Singaperbanga, U., & Abstract, K. (2022). Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api. *Jurnal Ilmiah Wahana Pendidikan*, *8*(16), 316–326. https://doi.org/10.5281/zenodo.7067707

Nayak, Mr. P., T, C. P., S, C. A., S, D., & Rakesh, C. H. (2022). Exploring the Hypertext Transfer Protocol. *International Journal of Advanced Research in Science, Communication and Technology*, 736–738. https://doi.org/10.48175/IJARSCT-7044

nest. (2024). *Hello, nest! A progressive Node.js framework for building efficient, reliable and scalable server-side applications.*

Nurhayati, E., & Agussalim, A. (2023). Rancang Bangun Back-end API pada Aplikasi Mobile AyamHub Menggunakan Framework Node JS Express. *Jurnal Sistem Dan Teknologi Informasi (JustIN)*, *11*(3), 524. https://doi.org/10.26418/justin.v11i3.66823

Peña-Monferrer, C., & Diaz-Marin, C. (2022). rom.js/cfd.xyz: An open-source framework for generating and visualizing parametric CFD results. *OpenFOAM® Journal*, *2*, 143–148. https://doi.org/10.51560/ofj.v2.83

Permana, I., & Salisah, F. N. S. (2022). Pengaruh Normalisasi Data Terhadap Performa Hasil Klasifikasi Algoritma Backpropagation. *Indonesian Journal of Informatic Research and Software Engineering (IJIRSE)*, *2*(1), 67–72. https://doi.org/10.57152/ijirse.v2i1.311

Putra, M. G. L., & Putera, M. I. A. (2019). ANALISIS PERBANDINGAN METODE SOAP DAN REST YANG DIGUNAKAN PADA FRAMEWORK FLASK UNTUK MEMBANGUN WEB SERVICE. *SCAN - Jurnal Teknologi Informasi Dan Komunikasi*, *14*(2). https://doi.org/10.33005/scan.v14i2.1480

Satwika, I. K. S., & Semadi, K. N. (2020). PERBANDINGAN PERFORMANSI WEB SERVER APACHE DAN NGINX DENGAN MENGGUNAKAN IPV6. *SCAN - Jurnal Teknologi Informasi Dan Komunikasi*, *15*(1). https://doi.org/10.33005/scan.v15i1.1847

Suwarno, S., & Afandi. (2022). Analisis perbandingan Codeigniter dan Yii framework pada perancangan website rencana anggaran biaya. *Jurnal CoSciTech (Computer Science and Information Technology)*, *3*(3), 249–258. https://doi.org/10.37859/coscitech.v3i3.4338

Yatini, I., Nurwiyati, F. W., & Anam, K. (2021). *PERFORMA MICROFRAMEWORK PHP PADA REST API MENGGUNAKAN METODE LOAD TESTING* (Vol. 19, Issue 2, p. 12).

# LAMPIRAN

Lampiran 1 : data mentah pengujian parameter *time taken for test*

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | pengujian ke- | framework | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 1 | 0.05 | 0.043 | 0.034 | 0.079 | 0.018 | 0.027 |
| GET | konstan | O(1) | 0 | 50 | 50 | 2 | 0.04 | 0.042 | 0.038 | 0.042 | 0.017 | 0.025 |
| GET | konstan | O(1) | 0 | 50 | 50 | 3 | 0.036 | 0.041 | 0.035 | 0.042 | 0.018 | 0.027 |
| GET | konstan | O(1) | 0 | 50 | 50 | 4 | 0.036 | 0.04 | 0.036 | 0.042 | 0.024 | 0.024 |
| GET | konstan | O(1) | 0 | 50 | 50 | 5 | 0.035 | 0.038 | 0.038 | 0.042 | 0.016 | 0.024 |
| GET | konstan | O(1) | 0 | 50 | 50 | 6 | 0.037 | 0.042 | 0.038 | 0.042 | 0.017 | 0.027 |
| GET | konstan | O(1) | 0 | 50 | 50 | 7 | 0.039 | 0.044 | 0.036 | 0.066 | 0.018 | 0.026 |
| GET | konstan | O(1) | 0 | 50 | 50 | 8 | 0.034 | 0.042 | 0.037 | 0.038 | 0.016 | 0.027 |
| GET | konstan | O(1) | 0 | 50 | 50 | 9 | 0.04 | 0.061 | 0.037 | 0.038 | 0.017 | 0.027 |
| GET | konstan | O(1) | 0 | 50 | 50 | 10 | 0.037 | 0.039 | 0.033 | 0.041 | 0.017 | 0.027 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| GET | linear | O(n) | 10 | 50 | 50 | 1 | 0.236 | 0.075 | 0.089 | 0.107 | 0.102 | 0.236 |
| GET | linear | O(n) | 10 | 50 | 50 | 2 | 0.101 | 0.095 | 0.068 | 0.087 | 0.066 | 0.153 |
| GET | linear | O(n) | 10 | 50 | 50 | 3 | 0.099 | 0.066 | 0.061 | 0.065 | 0.058 | 0.115 |
| GET | linear | O(n) | 10 | 50 | 50 | 4 | 0.114 | 0.074 | 0.056 | 0.067 | 0.061 | 0.634 |
| GET | linear | O(n) | 10 | 50 | 50 | 5 | 0.082 | 0.06 | 0.052 | 0.068 | 0.061 | 0.201 |
| GET | linear | O(n) | 10 | 50 | 50 | 6 | 0.075 | 0.058 | 0.125 | 0.073 | 0.058 | 0.11 |
| GET | linear | O(n) | 10 | 50 | 50 | 7 | 0.069 | 0.076 | 0.057 | 0.069 | 0.055 | 0.101 |
| GET | linear | O(n) | 10 | 50 | 50 | 8 | 0.078 | 0.057 | 0.054 | 0.063 | 0.053 | 0.107 |
| GET | linear | O(n) | 10 | 50 | 50 | 9 | 0.084 | 0.056 | 0.055 | 0.072 | 0.054 | 0.105 |
| GET | linear | O(n) | 10 | 50 | 50 | 10 | 0.117 | 0.059 | 0.051 | 0.069 | 0.06 | 0.1 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| POST | linear | O(n) | 10 | 50 | 50 | 1 | 0.182 | 0.165 | 0.123 | 0.118 | 0.513 | 0.163 |
| POST | linear | O(n) | 10 | 50 | 50 | 2 | 0.129 | 0.076 | 0.06 | 0.104 | 0.219 | 0.144 |
| POST | linear | O(n) | 10 | 50 | 50 | 3 | 0.076 | 0.083 | 0.06 | 0.079 | 0.133 | 0.103 |
| POST | linear | O(n) | 10 | 50 | 50 | 4 | 0.076 | 0.066 | 0.045 | 0.089 | 0.131 | 0.121 |
| POST | linear | O(n) | 10 | 50 | 50 | 5 | 0.071 | 0.075 | 0.054 | 0.074 | 0.129 | 0.15 |
| POST | linear | O(n) | 10 | 50 | 50 | 6 | 0.08 | 0.074 | 0.064 | 0.059 | 0.138 | 0.111 |
| POST | linear | O(n) | 10 | 50 | 50 | 7 | 0.054 | 0.165 | 0.043 | 0.064 | 0.131 | 0.1 |
| POST | linear | O(n) | 10 | 50 | 50 | 8 | 0.07 | 0.064 | 0.046 | 0.056 | 0.099 | 0.133 |
| POST | linear | O(n) | 10 | 50 | 50 | 9 | 0.061 | 0.061 | 0.055 | 0.075 | 0.105 | 0.096 |
| POST | linear | O(n) | 10 | 50 | 50 | 10 | 0.063 | 0.056 | 0.053 | 0.08 | 0.111 | 0.086 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| PUT | linear | O(n) | 10 | 50 | 50 | 1 | 0.108 | 0.086 | 0.075 | 0.095 | 0.122 | 0.181 |
| PUT | linear | O(n) | 10 | 50 | 50 | 2 | 0.095 | 0.062 | 0.063 | 0.081 | 0.057 | 0.117 |
| PUT | linear | O(n) | 10 | 50 | 50 | 3 | 0.083 | 0.063 | 0.067 | 0.066 | 0.062 | 0.136 |
| PUT | linear | O(n) | 10 | 50 | 50 | 4 | 0.086 | 0.062 | 0.057 | 0.068 | 0.061 | 0.374 |
| PUT | linear | O(n) | 10 | 50 | 50 | 5 | 0.54 | 0.063 | 0.056 | 0.063 | 0.058 | 0.126 |
| PUT | linear | O(n) | 10 | 50 | 50 | 6 | 0.07 | 0.065 | 0.051 | 0.066 | 0.051 | 0.104 |
| PUT | linear | O(n) | 10 | 50 | 50 | 7 | 0.076 | 0.084 | 0.055 | 0.061 | 0.05 | 0.105 |
| PUT | linear | O(n) | 10 | 50 | 50 | 8 | 0.071 | 0.056 | 0.052 | 0.06 | 0.055 | 0.105 |
| PUT | linear | O(n) | 10 | 50 | 50 | 9 | 0.08 | 0.069 | 0.059 | 0.061 | 0.051 | 0.097 |
| PUT | linear | O(n) | 10 | 50 | 50 | 10 | 0.09 | 0.048 | 0.055 | 0.057 | 0.053 | 0.093 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| DELETE | linear | O(n) | 10 | 50 | 50 | 1 | 0.106 | 0.073 | 0.068 | 0.097 | 0.135 | 0.14 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 2 | 0.072 | 0.074 | 0.057 | 0.078 | 0.058 | 0.106 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 3 | 0.077 | 0.068 | 0.078 | 0.064 | 0.059 | 0.102 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 4 | 0.076 | 0.052 | 0.048 | 0.058 | 0.055 | 0.219 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 5 | 0.088 | 0.064 | 0.057 | 0.062 | 0.054 | 0.125 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 6 | 0.074 | 0.442 | 0.05 | 0.065 | 0.056 | 0.103 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 7 | 0.065 | 0.064 | 0.118 | 0.055 | 0.047 | 0.111 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 8 | 0.062 | 0.056 | 0.053 | 0.053 | 0.051 | 0.093 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 9 | 0.061 | 0.053 | 0.055 | 0.068 | 0.052 | 0.092 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 10 | 0.079 | 0.05 | 0.052 | 0.063 | 0.049 | 0.094 |

Lampiran 2 : data mentah pengujian parameter *request per second*

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | pengujian ke- | framework | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 1 | 992.38 | 1162.66 | 1470.42 | 16.24 | 2777.78 | 1851.65 |
| GET | konstan | O(1) | 0 | 50 | 50 | 2 | 1249.91 | 1190.36 | 1315.65 | 1190.36 | 2941.35 | 1999.84 |
| GET | konstan | O(1) | 0 | 50 | 50 | 3 | 1388.66 | 1219.3 | 1428.45 | 1190.25 | 2777.78 | 1851.51 |
| GET | konstan | O(1) | 0 | 50 | 50 | 4 | 1389.16 | 1249.75 | 1388.7 | 1190.36 | 2083.16 | 2083.07 |
| GET | konstan | O(1) | 0 | 50 | 50 | 5 | 1428.37 | 1315.69 | 1315.72 | 1190.22 | 3124.61 | 2083.16 |
| GET | konstan | O(1) | 0 | 50 | 50 | 6 | 1351.24 | 1190.39 | 1315.55 | 1190.33 | 2941.18 | 1851.65 |
| GET | konstan | O(1) | 0 | 50 | 50 | 7 | 1282.02 | 1136.21 | 1388.77 | 757.5 | 2778.7 | 1922.93 |
| GET | konstan | O(1) | 0 | 50 | 50 | 8 | 1470.37 | 1190.36 | 1351.24 | 1315.65 | 3125.2 | 1851.78 |
| GET | konstan | O(1) | 0 | 50 | 50 | 9 | 1249.78 | 819.56 | 1351.21 | 1315.62 | 2941.18 | 1851.58 |
| GET | konstan | O(1) | 0 | 50 | 50 | 10 | 1351.17 | 1281.89 | 1514.97 | 1219.42 | 2941 | 1851.85 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| GET | linear | O(n) | 10 | 50 | 50 | 1 | 211.58 | 665.2 | 559.97 | 466.96 | 488.1 | 212.26 |
| GET | linear | O(n) | 10 | 50 | 50 | 2 | 497 | 528.55 | 735.27 | 577.67 | 757.6 | 327.42 |
| GET | linear | O(n) | 10 | 50 | 50 | 3 | 503.69 | 757.52 | 819.55 | 767 | 869.08 | 435.09 |
| GET | linear | O(n) | 10 | 50 | 50 | 4 | 440.36 | 675.55 | 892.87 | 746.08 | 826.16 | 78.89 |
| GET | linear | O(n) | 10 | 50 | 50 | 5 | 613.07 | 833.06 | 958.02 | 740.51 | 819.52 | 248.8 |
| GET | linear | O(n) | 10 | 50 | 50 | 6 | 667.8 | 862.32 | 399.97 | 682.23 | 868.86 | 456 |
| GET | linear | O(n) | 10 | 50 | 50 | 7 | 729.36 | 657.54 | 877.01 | 729.27 | 909.01 | 494.12 |
| GET | linear | O(n) | 10 | 50 | 50 | 8 | 644.69 | 877.22 | 933.67 | 798.76 | 938.14 | 469.17 |
| GET | linear | O(n) | 10 | 50 | 50 | 9 | 595.09 | 892.94 | 916.74 | 694.54 | 924.47 | 475.26 |
| GET | linear | O(n) | 10 | 50 | 50 | 10 | 427.76 | 852.72 | 980.33 | 729.32 | 833.43 | 502.24 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| POST | linear | O(n) | 10 | 50 | 50 | 1 | 274.38 | 303.64 | 405.18 | 423.6 | 97.43 | 306.68 |
| POST | linear | O(n) | 10 | 50 | 50 | 2 | 388.85 | 657.78 | 833.33 | 482.91 | 228.46 | 347.12 |
| POST | linear | O(n) | 10 | 50 | 50 | 3 | 661.38 | 602.36 | 833.44 | 636.68 | 376.82 | 484.24 |
| POST | linear | O(n) | 10 | 50 | 50 | 4 | 656.29 | 757.6 | 1111.43 | 564.64 | 381.83 | 413.23 |
| POST | linear | O(n) | 10 | 50 | 50 | 5 | 705.05 | 666.45 | 925.84 | 674.96 | 388.59 | 333.4 |
| POST | linear | O(n) | 10 | 50 | 50 | 6 | 623 | 675.73 | 781.14 | 847.52 | 362.5 | 450.99 |
| POST | linear | O(n) | 10 | 50 | 50 | 7 | 933.85 | 302.24 | 1162.87 | 780.99 | 383.05 | 499.26 |
| POST | linear | O(n) | 10 | 50 | 50 | 8 | 710.94 | 784.83 | 1098.35 | 892.94 | 507.34 | 376.74 |
| POST | linear | O(n) | 10 | 50 | 50 | 9 | 817.41 | 819.55 | 909.21 | 665.33 | 475.21 | 518.92 |
| POST | linear | O(n) | 10 | 50 | 50 | 10 | 799.32 | 889.82 | 941.66 | 628.16 | 450.44 | 584.75 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| PUT | linear | O(n) | 10 | 50 | 50 | 1 | 463 | 584.45 | 665.24 | 524.28 | 408.18 | 276.7 |
| PUT | linear | O(n) | 10 | 50 | 50 | 2 | 526.07 | 806.43 | 793.63 | 615.93 | 883.02 | 427.3 |
| PUT | linear | O(n) | 10 | 50 | 50 | 3 | 603.65 | 793.56 | 746.25 | 757.44 | 803.47 | 367.63 |
| PUT | linear | O(n) | 10 | 50 | 50 | 4 | 584.41 | 806.3 | 876.67 | 735.16 | 825.83 | 133.66 |
| PUT | linear | O(n) | 10 | 50 | 50 | 5 | 92.59 | 793.61 | 892.81 | 793.31 | 861.94 | 396.06 |
| PUT | linear | O(n) | 10 | 50 | 50 | 6 | 718.87 | 769.38 | 988.96 | 755.46 | 980.32 | 482.86 |
| PUT | linear | O(n) | 10 | 50 | 50 | 7 | 657.17 | 592.01 | 917.08 | 819.67 | 995.36 | 478.23 |
| PUT | linear | O(n) | 10 | 50 | 50 | 8 | 702.73 | 892.84 | 961.48 | 832.65 | 909.01 | 475.79 |
| PUT | linear | O(n) | 10 | 50 | 50 | 9 | 628.33 | 724.3 | 851.69 | 819.75 | 979.68 | 517.72 |
| PUT | linear | O(n) | 10 | 50 | 50 | 10 | 554.53 | 1041.67 | 909.12 | 877.15 | 943.5 | 536.02 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| DELETE | linear | O(n) | 10 | 50 | 50 | 1 | 472.28 | 684.74 | 734.73 | 514.66 | 369.95 | 358.18 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 2 | 697.37 | 675.59 | 877.24 | 643.8 | 869.01 | 471.56 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 3 | 648.99 | 735.23 | 637.22 | 786.96 | 847.11 | 490.17 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 4 | 654.18 | 954.67 | 1042.71 | 868.76 | 909.17 | 228.24 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 5 | 568.08 | 781.25 | 877.21 | 806.28 | 934.18 | 401.17 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 6 | 675.77 | 113.11 | 999.98 | 768.99 | 892.87 | 487.59 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 7 | 774.82 | 781.29 | 425.14 | 904.14 | 1063.81 | 449.47 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 8 | 806.41 | 892.92 | 948.78 | 943.33 | 980.33 | 536.82 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 9 | 819.65 | 943.41 | 909.09 | 735.21 | 961.54 | 542.08 |
| DELETE | linear | O(n) | 10 | 50 | 50 | 10 | 636.46 | 1004.64 | 961.56 | 793.45 | 1020.51 | 531.19 |

Lampiran 3 : data mentah pengujian parameter *time per request*

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | pengujian ke- | framework | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 1 | 50.384 | 43.005 | 34.004 | 3,078,59 | 18 | 27.003 |
| GET | konstan | O(1) | 0 | 50 | 50 | 2 | 40.003 | 42.004 | 38.004 | 42.004 | 16.999 | 25.002 |
| GET | konstan | O(1) | 0 | 50 | 50 | 3 | 36.006 | 41.007 | 35.003 | 42.008 | 18 | 27.005 |
| GET | konstan | O(1) | 0 | 50 | 50 | 4 | 35.993 | 40.008 | 36.005 | 42.004 | 24.002 | 24.003 |
| GET | konstan | O(1) | 0 | 50 | 50 | 5 | 35.005 | 38.003 | 38.002 | 42.009 | 16.002 | 24.002 |
| GET | konstan | O(1) | 0 | 50 | 50 | 6 | 37.003 | 42.003 | 38.007 | 42.005 | 17 | 27.003 |
| GET | konstan | O(1) | 0 | 50 | 50 | 7 | 39.001 | 44.006 | 36.003 | 66.007 | 17.994 | 26.002 |
| GET | konstan | O(1) | 0 | 50 | 50 | 8 | 34.005 | 42.004 | 37.003 | 38.004 | 15.999 | 27.001 |
| GET | konstan | O(1) | 0 | 50 | 50 | 9 | 40.007 | 61.008 | 37.004 | 38.005 | 17 | 27.004 |
| GET | konstan | O(1) | 0 | 50 | 50 | 10 | 37.005 | 39.005 | 33.004 | 41.003 | 17.001 | 27 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| GET | linear | O(n) | 10 | 50 | 50 | 1 | 236.319 | 75.165 | 89.291 | 107.076 | 102.439 | 235.562 |
| GET | linear | O(n) | 10 | 50 | 50 | 2 | 100.603 | 94.599 | 68.002 | 86.554 | 65.998 | 152.709 |
| GET | linear | O(n) | 10 | 50 | 50 | 3 | 99.267 | 66.005 | 61.009 | 65.189 | 57.532 | 114.918 |
| GET | linear | O(n) | 10 | 50 | 50 | 4 | 113.543 | 74.014 | 55.999 | 67.017 | 60.521 | 633.81 |
| GET | linear | O(n) | 10 | 50 | 50 | 5 | 81.557 | 60.02 | 52.191 | 67.521 | 61.011 | 200.964 |
| GET | linear | O(n) | 10 | 50 | 50 | 6 | 74.873 | 57.983 | 125.01 | 73.289 | 57.547 | 109.65 |
| GET | linear | O(n) | 10 | 50 | 50 | 7 | 68.553 | 76.041 | 57.012 | 68.562 | 55.005 | 101.191 |
| GET | linear | O(n) | 10 | 50 | 50 | 8 | 77.557 | 56.998 | 53.552 | 62.597 | 53.297 | 106.571 |
| GET | linear | O(n) | 10 | 50 | 50 | 9 | 84.021 | 55.995 | 54.541 | 71.99 | 54.085 | 105.206 |
| GET | linear | O(n) | 10 | 50 | 50 | 10 | 116.887 | 58.636 | 51.003 | 68.557 | 59.993 | 99.554 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| POST | linear | O(n) | 10 | 50 | 50 | 1 | 182.228 | 164.667 | 123.402 | 118.036 | 513.194 | 163.038 |
| POST | linear | O(n) | 10 | 50 | 50 | 2 | 128.585 | 76.013 | 60 | 103.538 | 218.858 | 144.042 |
| POST | linear | O(n) | 10 | 50 | 50 | 3 | 75.6 | 83.007 | 59.992 | 78.533 | 132.691 | 103.255 |
| POST | linear | O(n) | 10 | 50 | 50 | 4 | 76.186 | 65.998 | 44.987 | 88.552 | 130.949 | 120.998 |
| POST | linear | O(n) | 10 | 50 | 50 | 5 | 70.917 | 75.024 | 54.005 | 74.078 | 128.671 | 149.971 |
| POST | linear | O(n) | 10 | 50 | 50 | 6 | 80.257 | 73.994 | 64.009 | 58.996 | 137.931 | 110.867 |
| POST | linear | O(n) | 10 | 50 | 50 | 7 | 53.542 | 165.433 | 42.997 | 64.021 | 130.531 | 100.149 |
| POST | linear | O(n) | 10 | 50 | 50 | 8 | 70.329 | 63.708 | 45.523 | 55.995 | 98.554 | 132.719 |
| POST | linear | O(n) | 10 | 50 | 50 | 9 | 61.169 | 61.009 | 54.993 | 75.151 | 105.216 | 96.354 |
| POST | linear | O(n) | 10 | 50 | 50 | 10 | 62.553 | 56.191 | 53.098 | 79.598 | 111.002 | 85.507 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| PUT | linear | O(n) | 10 | 50 | 50 | 1 | 107.992 | 85.55 | 75.161 | 95.369 | 122.496 | 180.699 |
| PUT | linear | O(n) | 10 | 50 | 50 | 2 | 95.045 | 62.002 | 63.002 | 81.178 | 56.624 | 117.013 |
| PUT | linear | O(n) | 10 | 50 | 50 | 3 | 82.829 | 63.007 | 67.002 | 66.012 | 62.23 | 136.005 |
| PUT | linear | O(n) | 10 | 50 | 50 | 4 | 85.556 | 62.012 | 57.034 | 68.012 | 60.545 | 374.087 |
| PUT | linear | O(n) | 10 | 50 | 50 | 5 | 539.997 | 63.003 | 56.003 | 63.027 | 58.009 | 126.242 |
| PUT | linear | O(n) | 10 | 50 | 50 | 6 | 69.554 | 64.987 | 50.558 | 66.185 | 51.004 | 103.55 |
| PUT | linear | O(n) | 10 | 50 | 50 | 7 | 76.084 | 84.458 | 54.521 | 61 | 50.233 | 104.552 |
| PUT | linear | O(n) | 10 | 50 | 50 | 8 | 71.151 | 56.001 | 52.003 | 60.049 | 55.005 | 105.089 |
| PUT | linear | O(n) | 10 | 50 | 50 | 9 | 79.576 | 69.032 | 58.707 | 60.994 | 51.037 | 96.578 |
| PUT | linear | O(n) | 10 | 50 | 50 | 10 | 90.167 | 48 | 54.998 | 57.003 | 52.994 | 93.28 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 1 | 18.391 | 22.258 | 16.142 | 19.459 | 11.49 | 13.541 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 2 | 18.631 | 22.192 | 16.151 | 21.152 | 11.131 | 13.451 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 3 | 18.411 | 22.222 | 16.731 | 21.322 | 11.271 | 13.661 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 4 | 18.471 | 22.272 | 16.091 | 19.762 | 11.211 | 13.541 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 5 | 18.491 | 22.022 | 16.111 | 19.522 | 11.231 | 13.751 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 6 | 18.371 | 22.082 | 16.171 | 19.582 | 11.371 | 13.551 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 7 | 18.661 | 22.382 | 16.131 | 19.462 | 11.241 | 13.351 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 8 | 18.351 | 22.682 | 16.131 | 19.551 | 11.091 | 13.731 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 9 | 18.651 | 21.952 | 16.331 | 19.381 | 11.161 | 13.501 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 10 | 18.561 | 22.102 | 16.181 | 19.672 | 11.141 | 13.511 |

Lampiran 4 : data mentah pengujian parameter *transfer rate*

| metode HTTP | skenario | kompleksitas | jumlah data | c | n  (users) | pengujian ke- | framework | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 50 | 50 | 1 | 185.1 | 197.56 | 183.8 | 3.03 | 344.51 | 231.46 |
| GET | konstan | O(1) | 0 | 50 | 50 | 2 | 233.14 | 202.27 | 164.46 | 222.03 | 364.8 | 249.98 |
| GET | konstan | O(1) | 0 | 50 | 50 | 3 | 259.02 | 207.19 | 178.56 | 222.01 | 344.51 | 231.44 |
| GET | konstan | O(1) | 0 | 50 | 50 | 4 | 259.11 | 212.36 | 173.59 | 222.03 | 258.36 | 260.38 |
| GET | konstan | O(1) | 0 | 50 | 50 | 5 | 266.42 | 223.56 | 164.47 | 222 | 387.52 | 260.39 |
| GET | konstan | O(1) | 0 | 50 | 50 | 6 | 252.04 | 202.27 | 164.44 | 222.03 | 364.77 | 231.46 |
| GET | konstan | O(1) | 0 | 50 | 50 | 7 | 239.13 | 193.07 | 173.6 | 141.29 | 344.62 | 240.37 |
| GET | konstan | O(1) | 0 | 50 | 50 | 8 | 274.26 | 202.27 | 168.91 | 245.4 | 387.6 | 231.47 |
| GET | konstan | O(1) | 0 | 50 | 50 | 9 | 233.11 | 139.26 | 168.9 | 245.39 | 364.77 | 231.45 |
| GET | konstan | O(1) | 0 | 50 | 50 | 10 | 252.02 | 217.82 | 189.37 | 227.45 | 364.75 | 231.48 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| GET | linear | O(n) | 10 | 50 | 50 | 1 | 107.65 | 170.85 | 248.8 | 236.67 | 215.45 | 93.9 |
| GET | linear | O(n) | 10 | 50 | 50 | 2 | 252.38 | 135.75 | 327.4 | 294.48 | 336.63 | 144.85 |
| GET | linear | O(n) | 10 | 50 | 50 | 3 | 256.76 | 194.56 | 361 | 389.49 | 384.47 | 193.75 |
| GET | linear | O(n) | 10 | 50 | 50 | 4 | 224.05 | 173.5 | 397.6 | 378.87 | 367.09 | 13.48 |
| GET | linear | O(n) | 10 | 50 | 50 | 5 | 312.52 | 213.96 | 425.7 | 376.76 | 363.34 | 110.79 |
| GET | linear | O(n) | 10 | 50 | 50 | 6 | 339.77 | 221.48 | 177.7 | 347.11 | 386.06 | 202.62 |
| GET | linear | O(n) | 10 | 50 | 50 | 7 | 370.38 | 168.88 | 390.5 | 368.91 | 401.24 | 219.07 |
| GET | linear | O(n) | 10 | 50 | 50 | 8 | 328.64 | 225.3 | 414.9 | 407.18 | 416.85 | 208.47 |
| GET | linear | O(n) | 10 | 50 | 50 | 9 | 302.77 | 229.34 | 406.5 | 352.7 | 409.87 | 210.71 |
| GET | linear | O(n) | 10 | 50 | 50 | 10 | 78.53 | 219.01 | 434.6 | 368.93 | 369.51 | 223.16 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| POST | linear | O(n) | 10 | 50 | 50 | 1 | 62.16 | 56.93 | 66.08 | 95.97 | 7.14 | 50.01 |
| POST | linear | O(n) | 10 | 50 | 50 | 2 | 88.1 | 123.33 | 135.9 | 109.41 | 16.73 | 56.61 |
| POST | linear | O(n) | 10 | 50 | 50 | 3 | 149.84 | 112.94 | 135.92 | 144.25 | 27.6 | 78.97 |
| POST | linear | O(n) | 10 | 50 | 50 | 4 | 148.69 | 142.05 | 181.26 | 127.93 | 27.97 | 67.39 |
| POST | linear | O(n) | 10 | 50 | 50 | 5 | 159.74 | 124.96 | 150.99 | 152.92 | 28.46 | 54.37 |
| POST | linear | O(n) | 10 | 50 | 50 | 6 | 141.15 | 126.7 | 127.39 | 192.02 | 26.55 | 73.55 |
| POST | linear | O(n) | 10 | 50 | 50 | 7 | 211.57 | 56.67 | 189.65 | 176.94 | 28.06 | 81.42 |
| POST | linear | O(n) | 10 | 50 | 50 | 8 | 161.07 | 147.16 | 179.12 | 202.31 | 37.16 | 61.44 |
| POST | linear | O(n) | 10 | 50 | 50 | 9 | 185.19 | 153.67 | 148.28 | 150.74 | 34.81 | 84.63 |
| POST | linear | O(n) | 10 | 50 | 50 | 10 | 181.1 | 166.84 | 153.57 | 142.32 | 32.99 | 95.36 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| PUT | linear | O(n) | 10 | 50 | 50 | 1 | 102.64 | 106.73 | 105.24 | 116.22 | 64.18 | 43.78 |
| PUT | linear | O(n) | 10 | 50 | 50 | 2 | 116.62 | 147.27 | 125.55 | 136.54 | 138.83 | 67.6 |
| PUT | linear | O(n) | 10 | 50 | 50 | 3 | 133.82 | 144.92 | 118.06 | 167.91 | 126.33 | 58.16 |
| PUT | linear | O(n) | 10 | 50 | 50 | 4 | 129.55 | 147.24 | 138.69 | 162.97 | 129.84 | 22.84 |
| PUT | linear | O(n) | 10 | 50 | 50 | 5 | 20.53 | 144.93 | 141.25 | 175.86 | 135.52 | 62.66 |
| PUT | linear | O(n) | 10 | 50 | 50 | 6 | 159.36 | 140.5 | 156.46 | 167.47 | 154.13 | 76.39 |
| PUT | linear | O(n) | 10 | 50 | 50 | 7 | 145.68 | 108.11 | 145.08 | 181.7 | 156.5 | 75.66 |
| PUT | linear | O(n) | 10 | 50 | 50 | 8 | 155.78 | 163.05 | 152.11 | 184.58 | 142.92 | 75.27 |
| PUT | linear | O(n) | 10 | 50 | 50 | 9 | 139.29 | 132.27 | 134.74 | 181.72 | 154.03 | 81.9 |
| PUT | linear | O(n) | 10 | 50 | 50 | 10 | 120.76 | 190.23 | 143.83 | 194.45 | 148.34 | 84.8 |
| … |  |  |  |  |  |  |  |  |  |  |  |  |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 1 | 602.67 | 410.22 | 490.02 | 569.61 | 684.19 | 584.14 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 2 | 594.91 | 411.45 | 489.76 | 524.03 | 706.27 | 588.07 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 3 | 602.02 | 410.9 | 472.78 | 519.85 | 697.49 | 579.03 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 4 | 600.06 | 409.98 | 491.58 | 560.89 | 701.23 | 584.16 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 5 | 599.41 | 414.63 | 490.97 | 567.78 | 699.98 | 575.25 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 6 | 603.33 | 413.5 | 489.15 | 566.04 | 691.36 | 583.73 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 7 | 593.95 | 407.96 | 490.36 | 569.53 | 699.35 | 592.48 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 8 | 603.99 | 402.57 | 490.36 | 566.91 | 708.81 | 576.08 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 9 | 594.27 | 415.95 | 484.36 | 571.88 | 704.37 | 585.89 |
| DELETE | linear | O(n) | 1000 | 50 | 5000 | 10 | 597.15 | 413.13 | 488.85 | 563.45 | 705.63 | 585.46 |

Lampiran 5 : data mentah hasil pengujian *tool mitata js*

| metode http | skenario | kompleksitas | jumlah data | pengujian ke - | framework | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| express | hapi | koa | nest | elysia | fastify |
| GET | konstan | O(1) | 0 | 1 | 1022089 | 1451571.2 | 1511593 | 1400932 | 1731125.895 | 1304782 |
| GET | konstan | O(1) | 0 | 2 | 737195.1 | 1115308.45 | 1222000 | 1055878 | 1100928.647 | 1100732 |
| GET | konstan | O(1) | 0 | 3 | 573451 | 981204.496 | 1108466 | 1036733 | 767457.4209 | 906029.7 |
| GET | konstan | O(1) | 0 | 4 | 574366.3 | 994818.367 | 843112 | 1356157 | 885033.0056 | 678109.6 |
| GET | konstan | O(1) | 0 | 5 | 726227.4 | 711490.26 | 919230.3 | 686755.6 | 485634.6451 | 1015471 |
| GET | konstan | O(1) | 0 | 6 | 932256.4 | 680608.515 | 847378 | 740515.7 | 567179.0279 | 570695.2 |
| GET | konstan | O(1) | 0 | 7 | 808155.2 | 604713.714 | 567265 | 866927.9 | 484291.5615 | 566336.6 |
| GET | konstan | O(1) | 0 | 8 | 486026.3 | 741682.583 | 510289.7 | 841602 | 387834.6344 | 571569.5 |
| GET | konstan | O(1) | 0 | 9 | 480916.1 | 616377.162 | 482082.2 | 609920.6 | 378183.2432 | 477823.4 |
| GET | konstan | O(1) | 0 | 10 | 480451.8 | 798989.139 | 654273.5 | 597691.8 | 371816.7552 | 514791.5 |
| GET | linear | o(n) | 10 | 1 | 1745188 | 1843197.96 | 1711867 | 1994651 | 1873858.929 | 1564636 |
| GET | linear | o(n) | 10 | 2 | 1191228 | 1356112.23 | 1241243 | 1644357 | 1399636.062 | 1384190 |
| GET | linear | o(n) | 10 | 3 | 1143444 | 1308935.48 | 1056458 | 1297083 | 1055858.319 | 1865091 |
| GET | linear | o(n) | 10 | 4 | 1121941 | 1251613.89 | 1114973 | 1184838 | 1040197.039 | 1146974 |
| GET | linear | o(n) | 10 | 5 | 1362206 | 1350446 | 1038251 | 1190457 | 1153851.46 | 1269726 |
| GET | linear | o(n) | 10 | 6 | 1169045 | 1919960.79 | 1755972 | 1626921 | 1420005.869 | 1465767 |
| GET | linear | o(n) | 10 | 7 | 1620133 | 1470555.06 | 1457419 | 1282861 | 930570.1031 | 1808229 |
| GET | linear | o(n) | 10 | 8 | 1366846 | 1544464.43 | 1113528 | 1560139 | 1669574.536 | 1451317 |
| GET | linear | o(n) | 10 | 9 | 1110567 | 1553256.42 | 1125526 | 1214150 | 1453296.552 | 1161783 |
| GET | linear | o(n) | 10 | 10 | 1129196 | 1564848.38 | 1191409 | 1259354 | 1025044.822 | 1233592 |
| GET | linear | o(n) | 100 | 1 | 1500491 | 3137687.44 | 1995418 | 1560783 | 1698398.365 | 1535404 |
| GET | linear | o(n) | 100 | 2 | 1413177 | 2833362.5 | 1388123 | 2479963 | 1865168.142 | 1422277 |
| GET | linear | o(n) | 100 | 3 | 1334930 | 2800368.02 | 1341850 | 1652931 | 1268141.851 | 1759639 |
| GET | linear | o(n) | 100 | 4 | 1353284 | 3065638.31 | 1337115 | 1472717 | 1384226.316 | 1388089 |
| GET | linear | o(n) | 100 | 5 | 1552721 | 3815719.02 | 1363957 | 1531143 | 1171425.693 | 1921773 |
| GET | linear | o(n) | 100 | 6 | 1349093 | 5299945.95 | 1992460 | 1459909 | 2317845.956 | 1949018 |
| GET | linear | o(n) | 100 | 7 | 1400998 | 5139377.78 | 1404876 | 1645932 | 1146307.26 | 1794717 |
| GET | linear | o(n) | 100 | 8 | 2438545 | 5430602.86 | 1306883 | 1748946 | 1380006.114 | 1453539 |
| GET | linear | o(n) | 100 | 9 | 1361025 | 3440073.18 | 1276670 | 1564033 | 1120938.475 | 1566518 |
| GET | linear | o(n) | 100 | 10 | 1364888 | 14703190.2 | 1425112 | 1635643 | 1532836.253 | 1346238 |
| GET | linear | o(n) | 500 | 1 | 2847469 | 4129577.12 | 2847469 | 2955054 | 2334543.911 | 2713429 |
| GET | linear | o(n) | 500 | 2 | 2375859 | 4129685.43 | 2375859 | 3184886 | 2337074.908 | 2369358 |
| GET | linear | o(n) | 500 | 3 | 2336005 | 4131611.33 | 2336005 | 2640856 | 2107947.315 | 2673079 |
| GET | linear | o(n) | 500 | 4 | 2409943 | 4598822.63 | 2409943 | 2779375 | 2141144.595 | 2458363 |
| GET | linear | o(n) | 500 | 5 | 2411775 | 6911602.27 | 2411775 | 3369574 | 2333898.148 | 2467245 |
| GET | linear | o(n) | 500 | 6 | 3005861 | 5417648.28 | 3005861 | 3185122 | 2157766.552 | 3692408 |
| GET | linear | o(n) | 500 | 7 | 2689077 | 7099725.88 | 2689077 | 2735348 | 1868667.164 | 2865127 |
| GET | linear | o(n) | 500 | 8 | 2677505 | 7672623.75 | 2677505 | 2637462 | 2733072.174 | 2535688 |
| GET | linear | o(n) | 500 | 9 | 2422019 | 35293386.7 | 2422019 | 2767520 | 1884888.393 | 2889401 |
| GET | linear | o(n) | 500 | 10 | 2459482 | 10133750 | 2459482 | 3111814 | 2120367.785 | 2402720 |
| GET | linear | o(n) | 1000 | 1 | 2684608 | 6007742.72 | 3618978 | 3913756 | 2607874.689 | 3872518 |
| GET | linear | o(n) | 1000 | 2 | 2402068 | 5561804.42 | 3403071 | 3949163 | 3914523.899 | 3721488 |
| GET | linear | o(n) | 1000 | 3 | 2571919 | 6305117.53 | 3382538 | 3624738 | 2840209.009 | 3638955 |
| GET | linear | o(n) | 1000 | 4 | 2734327 | 6197101.01 | 3458013 | 3651102 | 2665765.823 | 3364883 |
| GET | linear | o(n) | 1000 | 5 | 2580640 | 7517893.83 | 3431011 | 3720275 | 2648675.847 | 3614397 |
| GET | linear | o(n) | 1000 | 6 | 2379252 | 9729567.8 | 3851492 | 3796790 | 2544198.78 | 4596930 |
| GET | linear | o(n) | 1000 | 7 | 2520589 | 14015390 | 3613298 | 4033816 | 2729821.121 | 3932199 |
| GET | linear | o(n) | 1000 | 8 | 2586063 | 9462445.31 | 3539024 | 4144540 | 2956436.493 | 4313855 |
| GET | linear | o(n) | 1000 | 9 | 2453200 | 13142279.5 | 4113889 | 3708541 | 2618673.967 | 3652468 |
| GET | linear | o(n) | 1000 | 10 | 2450017 | 13174908.9 | 3533669 | 4040821 | 2720334.483 | 3869806 |
| POST | linear | o(n) | 10 | 1 | 2360683 | 2554957.49 | 2248058 | 2441370 | 3871061.35 | 2222896 |
| POST | linear | o(n) | 10 | 2 | 1242179 | 1491171.43 | 1254704 | 1764534 | 2885071.233 | 1696384 |
| POST | linear | o(n) | 10 | 3 | 1164320 | 1386509.03 | 1149231 | 1568127 | 2077535.855 | 1343358 |
| POST | linear | o(n) | 10 | 4 | 1121534 | 1490788.84 | 1148415 | 1222495 | 2032146.154 | 1168464 |
| POST | linear | o(n) | 10 | 5 | 1218878 | 1612220.37 | 1084076 | 1258373 | 2654718.487 | 1308066 |
| POST | linear | o(n) | 10 | 6 | 1105730 | 2260635.9 | 1909274 | 1774124 | 3027334.928 | 1691986 |
| POST | linear | o(n) | 10 | 7 | 1558377 | 1586754.66 | 1201583 | 1508608 | 1883272.619 | 1716037 |
| POST | linear | o(n) | 10 | 8 | 1382068 | 1576802.3 | 1617739 | 1607915 | 1934087.156 | 1252970 |
| POST | linear | o(n) | 10 | 9 | 1092651 | 1662759.79 | 1109941 | 1415160 | 2647197.468 | 1026722 |
| POST | linear | o(n) | 10 | 10 | 1100180 | 6798695.35 | 1178327 | 1208215 | 2290785.507 | 1075977 |
| POST | linear | o(n) | 100 | 1 | 2364484 | 2345116.21 | 2655592 | 2489374 | 10043040.98 | 2565847 |
| POST | linear | o(n) | 100 | 2 | 1800140 | 2479981.36 | 1851944 | 2519466 | 17390719.44 | 1710391 |
| POST | linear | o(n) | 100 | 3 | 1605947 | 2375945.34 | 1604798 | 1942899 | 15012587.8 | 2659638 |
| POST | linear | o(n) | 100 | 4 | 1590880 | 1997967.85 | 1558374 | 2048319 | 10102927.87 | 1617556 |
| POST | linear | o(n) | 100 | 5 | 1797497 | 3204611.64 | 1594067 | 2000218 | 10778457.89 | 2076816 |
| POST | linear | o(n) | 100 | 6 | 1718179 | 2499969.39 | 3193348 | 3893724 | 29203450 | 1771393 |
| POST | linear | o(n) | 100 | 7 | 2061296 | 3015373.82 | 2499668 | 1985561 | 9480710.769 | 2722113 |
| POST | linear | o(n) | 100 | 8 | 2050699 | 2389899.15 | 1960992 | 3262642 | 12256842 | 1799159 |
| POST | linear | o(n) | 100 | 9 | 1596191 | 3691134.21 | 1569732 | 2247179 | 12565270.83 | 1615866 |
| POST | linear | o(n) | 100 | 10 | 1611107 | 4067088.19 | 1587273 | 2069809 | 22438292.59 | 1752120 |
| POST | linear | o(n) | 500 | 1 | 5515451 | 5248995.24 | 5614207 | 7323624 | 46831308.33 | 7064774 |
| POST | linear | o(n) | 500 | 2 | 4835200 | 4171768.33 | 4501981 | 10626791 | 83099433.33 | 4239765 |
| POST | linear | o(n) | 500 | 3 | 4537080 | 3919813.67 | 4383363 | 6656665 | 53596890 | 7464390 |
| POST | linear | o(n) | 500 | 4 | 4519936 | 4805655.86 | 4858396 | 4646066 | 57481960 | 4606090 |
| POST | linear | o(n) | 500 | 5 | 4172679 | 6837329.35 | 3798694 | 6063528 | 82782233.33 | 4293150 |
| POST | linear | o(n) | 500 | 6 | 4528689 | 11250374 | 5955735 | 8223207 | 50136509.09 | 4089973 |
| POST | linear | o(n) | 500 | 7 | 6058143 | 10310371.4 | 4949077 | 6785120 | 46605725 | 6110124 |
| POST | linear | o(n) | 500 | 8 | 5494119 | 6367488.51 | 4287977 | 7225029 | 81486380 | 4543213 |
| POST | linear | o(n) | 500 | 9 | 3673773 | 10108432.3 | 5171571 | 6413416 | 42279307.69 | 5388917 |
| POST | linear | o(n) | 500 | 10 | 5029723 | 24392070 | 4507094 | 5072592 | 62256237.5 | 4544408 |
| POST | linear | o(n) | 1000 | 1 | 6615739 | 7785329.33 | 6972481 | 9838232 | 95708220 | 9556506 |
| POST | linear | o(n) | 1000 | 2 | 8695049 | 8328417.24 | 5868297 | 12321184 | 242627733.3 | 7009428 |
| POST | linear | o(n) | 1000 | 3 | 5920370 | 8625360 | 5313586 | 9669974 | 89330100 | 7206624 |
| POST | linear | o(n) | 1000 | 4 | 7019403 | 9079180.6 | 7881101 | 7758211 | 142809940 | 9235884 |
| POST | linear | o(n) | 1000 | 5 | 6892157 | 8219859.15 | 5291622 | 13154996 | 87349750 | 7263225 |
| POST | linear | o(n) | 1000 | 6 | 8650115 | 14458876.7 | 9287461 | 15628820 | 85221816.67 | 8462142 |
| POST | linear | o(n) | 1000 | 7 | 8868700 | 14436397.6 | 7026241 | 12081707 | 89272500 | 15058500 |
| POST | linear | o(n) | 1000 | 8 | 8668173 | 9588114.06 | 5313105 | 8901875 | 100399160 | 8208195 |
| POST | linear | o(n) | 1000 | 9 | 6417368 | 18102477.4 | 6944401 | 8564428 | 99880320 | 6431778 |
| POST | linear | o(n) | 1000 | 10 | 5872503 | 14898721.6 | 5897463 | 9927194 | 91161500 | 6307782 |
| PUT | linear | o(n) | 10 | 1 | 2177612 | 2650134.73 | 2076886 | 3218051 | 1897798.503 | 1979880 |
| PUT | linear | o(n) | 10 | 2 | 1631222 | 1933612.23 | 1725023 | 2235377 | 1414874.944 | 1651647 |
| PUT | linear | o(n) | 10 | 3 | 1607678 | 1877157.66 | 1597222 | 1867180 | 1993194.304 | 2465012 |
| PUT | linear | o(n) | 10 | 4 | 1600653 | 1838181.4 | 1578377 | 1704027 | 1293662.012 | 1852707 |
| PUT | linear | o(n) | 10 | 5 | 1884124 | 2460152.36 | 1536518 | 1605859 | 1323000.837 | 1900170 |
| PUT | linear | o(n) | 10 | 6 | 1590631 | 2224813.17 | 2236294 | 1896488 | 1859570.175 | 2221316 |
| PUT | linear | o(n) | 10 | 7 | 2102187 | 1925708.81 | 2222716 | 1843259 | 1582187.563 | 3636249 |
| PUT | linear | o(n) | 10 | 8 | 2136594 | 2046734.74 | 1629944 | 2635908 | 1883938.69 | 2260790 |
| PUT | linear | o(n) | 10 | 9 | 1521262 | 2036026.13 | 1609421 | 1679914 | 1330946.234 | 1729347 |
| PUT | linear | o(n) | 10 | 10 | 1547997 | 2732293.39 | 1644210 | 1575380 | 1564404.938 | 3012424 |
| PUT | linear | o(n) | 100 | 1 | 2077447 | 2444898.83 | 3084948 | 2182351 | 1872255 | 2093154 |
| PUT | linear | o(n) | 100 | 2 | 1953665 | 2445164.06 | 1967033 | 3150218 | 1877990.855 | 2021735 |
| PUT | linear | o(n) | 100 | 3 | 1855259 | 2328597.77 | 1905692 | 2252624 | 1695166.667 | 2415468 |
| PUT | linear | o(n) | 100 | 4 | 1775899 | 2221017.56 | 1840809 | 1920283 | 1678149.604 | 2252738 |
| PUT | linear | o(n) | 100 | 5 | 1930159 | 2965945.05 | 1861976 | 2452182 | 1841541.279 | 2101825 |
| PUT | linear | o(n) | 100 | 6 | 1868764 | 2642122.18 | 3649480 | 2057997 | 2827926.697 | 2105955 |
| PUT | linear | o(n) | 100 | 7 | 1905151 | 2700985.65 | 1935574 | 1978114 | 1914784.592 | 2477600 |
| PUT | linear | o(n) | 100 | 8 | 2623550 | 2399189.16 | 1831654 | 2463988 | 1973360.313 | 1808924 |
| PUT | linear | o(n) | 100 | 9 | 2201094 | 3630406.29 | 1759023 | 2010800 | 1429574.887 | 1833148 |
| PUT | linear | o(n) | 100 | 10 | 1829991 | 2874050 | 1884766 | 2391176 | 1597737.688 | 2402082 |
| PUT | linear | o(n) | 500 | 1 | 3119708 | 3585241.14 | 3711659 | 3514837 | 2764206.637 | 4056310 |
| PUT | linear | o(n) | 500 | 2 | 3051020 | 3562263.64 | 2940936 | 3658264 | 3301353.968 | 3625926 |
| PUT | linear | o(n) | 500 | 3 | 2972487 | 3517848.02 | 2978952 | 3067822 | 2579629.675 | 3602545 |
| PUT | linear | o(n) | 500 | 4 | 3495287 | 3598403.49 | 2917928 | 3012062 | 2781225.893 | 3028308 |
| PUT | linear | o(n) | 500 | 5 | 3039636 | 3833032.12 | 2902071 | 3734777 | 2966275.943 | 3623091 |
| PUT | linear | o(n) | 500 | 6 | 3010055 | 4255033.81 | 3209569 | 3429867 | 2742697.835 | 3617921 |
| PUT | linear | o(n) | 500 | 7 | 3222810 | 6399397.94 | 3410610 | 3216383 | 2485537.795 | 4635447 |
| PUT | linear | o(n) | 500 | 8 | 3125750 | 3809962.05 | 3029081 | 3163982 | 3837968.293 | 4356571 |
| PUT | linear | o(n) | 500 | 9 | 2977275 | 5543506.8 | 2878865 | 3048730 | 2463335.019 | 3314411 |
| PUT | linear | o(n) | 500 | 10 | 2933331 | 4918362.99 | 3122675 | 4375111 | 2978155.66 | 3175868 |
| PUT | linear | o(n) | 1000 | 1 | 4980470 | 5502902.63 | 4666195 | 6299972 | 4029728.387 | 4772773 |
| PUT | linear | o(n) | 1000 | 2 | 4671138 | 5465197.37 | 4611892 | 5188408 | 6418528.571 | 5779825 |
| PUT | linear | o(n) | 1000 | 3 | 4465259 | 7124024.42 | 4643145 | 5305938 | 4486117.266 | 6148970 |
| PUT | linear | o(n) | 1000 | 4 | 4923296 | 6295136.36 | 4665651 | 4654251 | 4673079.545 | 4809427 |
| PUT | linear | o(n) | 1000 | 5 | 4728027 | 6453775.53 | 4748764 | 4777167 | 3738679.042 | 4753044 |
| PUT | linear | o(n) | 1000 | 6 | 4596480 | 9984506.56 | 4841053 | 4695374 | 3743964.497 | 5671556 |
| PUT | linear | o(n) | 1000 | 7 | 4892084 | 8298100 | 4924188 | 4689250 | 3888736.646 | 4826560 |
| PUT | linear | o(n) | 1000 | 8 | 5210774 | 7763732.89 | 4803358 | 4963069 | 4178039.735 | 5871916 |
| PUT | linear | o(n) | 1000 | 9 | 4654441 | 17693896.9 | 5413560 | 4925416 | 4334300 | 4909494 |
| PUT | linear | o(n) | 1000 | 10 | 4636570 | 12168826 | 4654476 | 5812220 | 3832905.455 | 5768765 |
| DELETE | linear | o(n) | 10 | 1 | 1199796 | 1621570.51 | 1367682 | 1479293 | 1554257.882 | 1297259 |
| DELETE | linear | o(n) | 10 | 2 | 1027873 | 1338869.96 | 1148799 | 1530586 | 1147808.073 | 1040869 |
| DELETE | linear | o(n) | 10 | 3 | 1005717 | 1227454.56 | 957109.4 | 1180401 | 1385486.245 | 1219158 |
| DELETE | linear | o(n) | 10 | 4 | 947034.4 | 1263616.83 | 979347.7 | 1151160 | 828413.4289 | 953484.4 |
| DELETE | linear | o(n) | 10 | 5 | 1232744 | 1423184.34 | 934981.2 | 1077323 | 952745.6964 | 1434548 |
| DELETE | linear | o(n) | 10 | 6 | 986252.8 | 1485699.76 | 1341922 | 1236675 | 1450172.874 | 1194611 |
| DELETE | linear | o(n) | 10 | 7 | 1311878 | 1352847.06 | 1719928 | 1149781 | 1053622.259 | 1960678 |
| DELETE | linear | o(n) | 10 | 8 | 1431429 | 1273264.9 | 946996.6 | 2568552 | 1045522.517 | 1066120 |
| DELETE | linear | o(n) | 10 | 9 | 935531.9 | 1351059.17 | 1182401 | 1345377 | 1021016.801 | 1152369 |
| DELETE | linear | o(n) | 10 | 10 | 946110.2 | 2208551.43 | 1012035 | 1077325 | 925374.2313 | 1325640 |
| DELETE | linear | o(n) | 100 | 1 | 1097260 | 1458486.87 | 1512383 | 1418472 | 1077945.254 | 1340791 |
| DELETE | linear | o(n) | 100 | 2 | 982474.9 | 1357793.78 | 1019265 | 1816354 | 970054.6851 | 1206867 |
| DELETE | linear | o(n) | 100 | 3 | 980501.9 | 1315479.17 | 973833.5 | 1272421 | 1341152.665 | 1335394 |
| DELETE | linear | o(n) | 100 | 4 | 998336.3 | 1395271.4 | 990640.5 | 1106907 | 850737.6344 | 1409445 |
| DELETE | linear | o(n) | 100 | 5 | 988411.2 | 1748269.72 | 975094.1 | 1600283 | 872873.6111 | 1350473 |
| DELETE | linear | o(n) | 100 | 6 | 962525.2 | 1640253.26 | 1867769 | 1279641 | 1980489.457 | 1012536 |
| DELETE | linear | o(n) | 100 | 7 | 1120593 | 1803100.57 | 1004867 | 1117409 | 1376295.842 | 1517648 |
| DELETE | linear | o(n) | 100 | 8 | 1354366 | 1660912.37 | 931758.5 | 1349933 | 1247230.435 | 1022329 |
| DELETE | linear | o(n) | 100 | 9 | 961370.5 | 1872003.63 | 1005196 | 1133803 | 912969.8251 | 903832.5 |
| DELETE | linear | o(n) | 100 | 10 | 1059441 | 2498131.75 | 1161914 | 1599551 | 926890.8148 | 1169731 |
| DELETE | linear | o(n) | 500 | 1 | 1223962 | 1398904.89 | 1387244 | 1466092 | 1263069.721 | 1316618 |
| DELETE | linear | o(n) | 500 | 2 | 1009475 | 1397875.38 | 992259.2 | 1280057 | 1420398.658 | 1059521 |
| DELETE | linear | o(n) | 500 | 3 | 905179.4 | 1336566.67 | 977956.4 | 1380517 | 950671.6642 | 1119317 |
| DELETE | linear | o(n) | 500 | 4 | 1443579 | 1613643.91 | 934130.2 | 1173543 | 881505.9474 | 909189.9 |
| DELETE | linear | o(n) | 500 | 5 | 1002810 | 1801082.32 | 979693.2 | 1364799 | 842119.5739 | 1017167 |
| DELETE | linear | o(n) | 500 | 6 | 1027579 | 2329798.9 | 1004624 | 1169531 | 1043872.414 | 1205765 |
| DELETE | linear | o(n) | 500 | 7 | 1109090 | 1559289.92 | 1134456 | 1066980 | 930972.9013 | 1858496 |
| DELETE | linear | o(n) | 500 | 8 | 1046597 | 1513664.72 | 931274.9 | 1118819 | 1315427.708 | 1158946 |
| DELETE | linear | o(n) | 500 | 9 | 975228.4 | 1988720.5 | 868790.1 | 1118267 | 880883.8483 | 1002075 |
| DELETE | linear | o(n) | 500 | 10 | 1027731 | 4050700 | 1021792 | 1345004 | 802249.4282 | 987737.4 |
| DELETE | linear | o(n) | 1000 | 1 | 1129455 | 1423076.08 | 1096578 | 2216100 | 1151645.173 | 1162831 |
| DELETE | linear | o(n) | 1000 | 2 | 1033085 | 1439100.68 | 971280 | 1299632 | 1931093.458 | 1098308 |
| DELETE | linear | o(n) | 1000 | 3 | 1015444 | 1854511.83 | 998227.3 | 1107006 | 1143350.633 | 1362395 |
| DELETE | linear | o(n) | 1000 | 4 | 975504.4 | 1420158.26 | 994715.7 | 1192457 | 1326888.518 | 920643.8 |
| DELETE | linear | o(n) | 1000 | 5 | 1135570 | 1843709.04 | 982715 | 1071552 | 822522.0339 | 972770.7 |
| DELETE | linear | o(n) | 1000 | 6 | 1036062 | 1547361.4 | 1062689 | 1431643 | 838968.8742 | 1505633 |
| DELETE | linear | o(n) | 1000 | 7 | 1093476 | 2057438.44 | 970784.5 | 1198069 | 787360.9877 | 1354950 |
| DELETE | linear | o(n) | 1000 | 8 | 1008703 | 1575511.9 | 896702.8 | 1230779 | 1043182.787 | 1335930 |
| DELETE | linear | o(n) | 1000 | 9 | 1015566 | 1487449.64 | 1524701 | 1099411 | 1095104.131 | 974624.4 |
| DELETE | linear | o(n) | 1000 | 10 | 926875.5 | 2487617.78 | 1015597 | 1297920 | 1140792.652 | 1017232 |