PERFORMA MICROFRAMEWORK PHP PADA REST API MENGGUNAKAN METODE *LOAD TESTING*

Indra Yatini[1)], F. Wiwiek Nurwiyati[2)], dan Khairul Anam[3)]
[1, 2, 3)]Teknik Informatika dan STMIK AKAKOM
Jl. Raya Janti No 143 Yogyakarta
e-mail : [1]indrayatini@akakom.ac.id, [2]wiwiek@akakom.ac.id, [3]iamkhoirulanam@gmail.com

Abstract

The need for flexibility in application development by minimizing the constraints of a server production environment makes n-tier architectures increasingly used. This architecture is implemented in the form of middleware or often called API. REST API is an API architecture that is currently the most widely used for middleware-based application development.

PHP is an easy-to-use programming language for developing REST APIs with its microframework. Each microframework has features such as database connection handling, URL routing, and performance is the top one. The large number of PHP microframeworks with different performance issues makes the selection of API-based application development cores very important. Especially if it is projected to handle data exchange or client requests in large numbers. As a basis for selection, performance testing needs to be carried out on each microframework to determine its suitability for API-based application development.

Performance testing uses the load testing method and develops the popular PHP microframework as a basis for creating test applications. The microframework includes FatFree, Lumen, Phalcon-micro, and Slim. Testing is carried out systematically using a test plan that has been designed for performance testing needs. The focus is to analyze the RPS (Request Per Seconds) and latency to the percentile that the wrk2 test tool generates on a predetermined test type.

Keywords: *Load Testing, PHP Microframework, Comparison, Performance, REST API.*

PENDAHULUAN

Semenjak diperkenalkan oleh Roy Fielding sekitar tahun 2000 pada disertasi PhD miliknya, teknologi pengembangan perangkat lunak berbasis REST (*Representational State Transfer*) adalah yang paling banyak digunakan untuk mengembangkan aplikasi berukuran personal maupun *enterprise*. REST merupakan gaya arsitektur untuk menyediakan standar antara sistem komputer di web, sehingga memudahkan sistem untuk berkomunikasi satu sama lain. Teknologi tersebut memanfaatkan sebuah protokol yang sudah disediakan oleh HTTP untuk berkomunikasi pada *database*[1]. Implementasi dari REST yang memiliki tingkat kehandalan tinggi membuat para pengembang aplikasi web mulai berpindah dan mengimplementasikan sistem menggunakan jenis arsitektur *n-tier application* yang semulanya bertipe *client-server*.

Arsitektur *n-tier application* merupakan perantara disebut juga sebagai *middleware* atau lebih populer dengan sebutan API *(Application Programming Interface)*. Melalui *middleware*, aplikasi dapat dikembangkan lebih masif karena tidak perlu membuat secara berulang untuk menjadikan aplikasi yang berbeda. Pengembang aplikasi hanya perlu mendesain ulang tampilan sesuai kebutuhan dan berinteraksi dengan data sesuai aturan yang sudah ditetapkan pada API.[2]

PHP merupakan salah satu bahasa pemograman *server-side* yang dapat digunakan untuk mengembangkan aplikasi REST API. Kemudahan dalam mengembangkan aplikasi menggunakan PHP disertai dengan komunitasnya yang sangat besar, menjadikan bahasa pemrograman ini pilihan bagi pengembang aplikasi pemula maupun yang sudah berpengalaman. [2]

Berbagai *framework* ditawarkan PHP dengan kompatibilitas dan kemampuan beragam. Jenis *framework* yang dirancang khusus untuk mengembangkan *middleware* disebut juga *microframework*. *Microframework* merupakan sebuah versi "kecil" dari *full-stack framework*. Fitur yang ditawarkan *microframework* pada kondisi *default* adalah protokol HTTP. Secara spesifik, *microframework* digunakan untuk membangun sebuah API. Beberapa *microframework* PHP yang populer saat ini diantaranya adalah FatFree, Lumen, Phalcon, dan Slim. Setiap *microframework* tersebut memiliki fitur yang diunggulkan seperti penanganan koneksi *database*, URL *routing*, kemudahan pengembangan, dan yang paling banyak diunggulkan adalah tentang performa. Banyaknya *microframework* PHP dengan metode pengembangan yang berbeda tentunya memiliki perbedaan pada segi performa.[3]

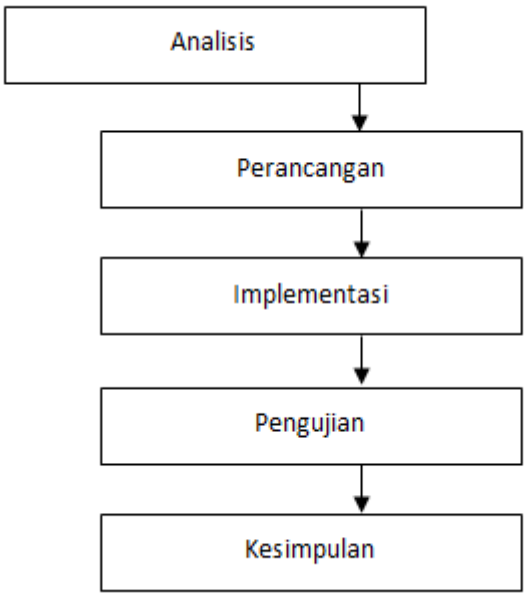
Hal itu menjadikan pemilihan *microframework* sebagai *core* pengembangan aplikasi berbasis API sangat penting. Terlebih jika sebuah API diproyeksikan akan menangani pertukaran data atau *request client* dalam jumlah besar. Sebagai dasar pemilihan, maka pengujian performa perlu dilakukan pada setiap *microframework* untuk menentukan kesesuaian pada pengembangan aplikasi berbasis API. Dari permasalah tersebut, pada penelitian ini akan dilakukan analisis perbandingan performa *microframework* PHP pada REST API. [4]

METODE PENELITIAN

Pada penelitian ini penulis mencoba untuk menganalisis baik dari sisi data maupun rancangan sampai kepada tahap implementasi yang di teruskan ke tahap pengujian agar di dapat suatu kesimpulan.

A. Gambaran Sistem

Berikut merupakan gambar prosedur penelitian yang dilakukan penulis :



Gambar 1 Prosedur penelitian

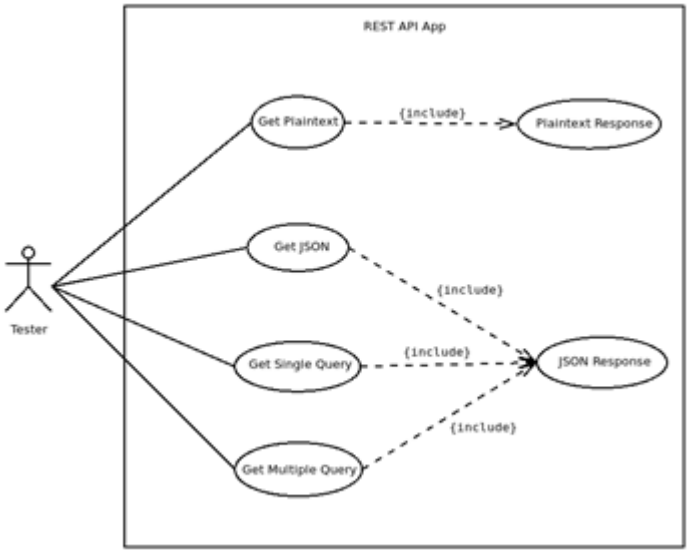
1. Analisis
   Pada tahapan ini mengumpulkan data-data dengan melakukan observasi untuk sistem yang akan dibangun
2. Perancangan

Pada tahap ini menggambarkan hasil dari analisis pengumpulan data-data yang ada pada selanjutnya juga dilakukan perancangan *workflow*dengan menggunakan UML Diagram dan desain untuk membuat atau membangun *web service*.

3. Implementasi
   Pada tahap ini mentransfer hasil dari perancangan ke dalam coding atau bahasa pemrograman dengan menggunakan masing-masing *microframework* untuk membuat atau membangun *web service*.
4. Pengujian
   Pada tahap ini dilakukan pengujian terhadap *web service* yang telah dibuat dan pengukuran performa dengan cara mengakses *web service* yang telah dibuat atau disebut dengan *client*, *client* mengirimkan *request* ke *web service* dan menghitung *response time* dari *web service*. Pada sisi *web service* sendiri juga dilakukan analisis penggunaan CPU (*Central Processing Unit*) dan penggunaan RAM (*Random Access Memory*) pada saat proses request sedang berlangsung.
5. Kesimpulan
   Setelah mengukur masing-masing performa *web service* maka selanjutnya membuat perbandingan ataumengkomparasi hasil dari pengukuran untuk menghasilkan rekomendasi*microframework* apa yang paling cepat dan efektif.

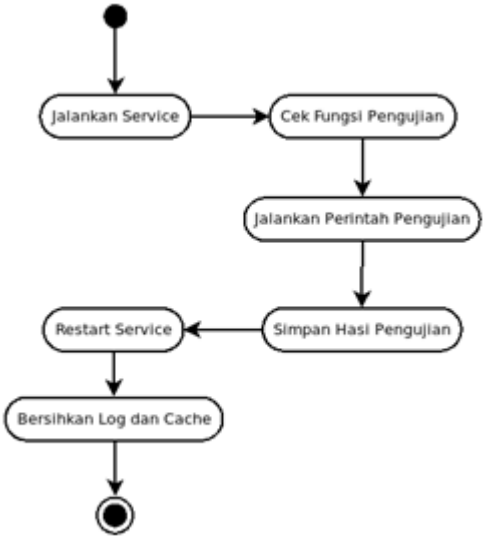
B. **Analisis Sistem**

Use case diagram merupakan sebuah rancangan relasi antara *user* dan sistem dengan memberikan spesifikasi terhadap kontruksi pada sebuah sistem yang akan dikerjakan. Penggunaan *use case* dapat mempermudah dalam menjelaskan lingkup sistem untuk pengujian performa. Implementasi *use case diagram* dapat dilihat pada gambar berikut.



Gambar 2. Analisis pengujian performa

Use case diagram tersebut menjelaskan bahwa aplikasi REST API mempunyai beberapa simulasi *activity* yaitu *Get* Plaintext, *Get* JSON, *Get Single Query*, *Get Multiple Query*. Setiap simulasi *activity* tersebut akan memberikan *response* berbeda. JSON *response* digunakan untuk menampilkan data dalam format *application/json*. Sedangkan Plaintext *response* akan menampilkan data dalam format *text/plain*.



Gambar 3. Perancangan Pengujian sistem

Pengujian performa diawali dengan menjalankan kebutuhan *service* aplikasi uji. Sebelum melakukan pengujian akan dilakukan pengecekan fungsi sekaligus *exercise* awal aplikasi uji dengan alat uji wrk2.

Aplikasi akan diuji sesuai skenario yang sudah ditetapkan sebelumnya. Hasil dari pengujian tersebut disimpan ke dalam *log file* sebagai bahan analisa. Setiap selesai pengujian pada tiap jenisnya, *service* akan dilakukan *restart*, menghapus *log* dan *cache* untuk mempersiapkan pengujian selanjutnya dengan lingkungan *service* yang netral. Parameter-parameter yang digunakan dalam pengujian dapat dilihat pada Tabel 1.

Tabel 1 Paramater Pengujian Sistem

No	Parameter	Satuan	Keterangan
1	*Response time (client)*	*Milisecond* (*ms*)	Waktu respon yang dibutuhkan oleh *microframework*
2	CPU *usage (server)*	Persentase (*%*)	Penggunaan CPU
3	RAM *usage (server)*	*Byte (B)*	Penggunaan RAM

Pengujian pada sisi client digunakan untuk menguji kecepatan eksekusi perintah dari masing-masing *microframework* yang telah diimplementasikan. Aplikasi client diimplementasikan pada sistem operasi *Windows* yang berfungsi untuk mencatat waktu pengiriman request dan penerimaan response, sehingga akan didapatkan total waktu proses yang dibutuhkan oleh *microframework*. Sedangkan pengujian pada sisi server digunakan untuk menguji penggunaan CPU dan penggunaan RAM saat proses eksekusi perintah pada masing-masing *microframework*. Pengujian penggunaan CPU dan penggunaan RAM pada server akan dilakukan dengan menggunakan *Resource Monitor* yang ada pada sistem operasi.

HASIL DAN PEMBAHASAN

Analisis performa dilakukan untuk membaca hasil dari pembobotan skor aplikasi uji berdasarkan jenis pengujian yang dimasukan dalam tabel dan visualisasi variable *latency* terhadap *percentile* berbentuk grafik.

A. Tabel Perbandingan

Tabel 2 berikut merupakan hasil dari beberapa jenis pengujian sesuai dengan rencana pengujian yang telah dikerjakan. RPS tertinggi dari tiap jenis pengujian nilainya disederhanakan dengan melakukan pembagian angka 100. Hasil akhir atau total skor pembobotan merupakan penjumlahan skor tiap jenis pengujian pada aplikasi uji.

Tabel 2 Pembobotan Skor Aplikasi Uji Berdasarkan Jenis Pengujian

Jenis Pengujian		**FatFree**	**Lumen**	**Phalcon-micro**	**Slim**
Plaintext	RPS	45,76	25,23	46,17	25,57
	Bobot	1	1	1	2
	Skor	45,76	25,23	46,17	51,14
JSON Serialization	RPS	46,45	24,85	45,1	25,82
	Bobot	3	1	4	4
	Skor	139,35	24,85	180,4	103,28
Single Query	RPS	27,74	13,38	22,59	17,25
	Bobot	1	2	5	4
	Skor	27,74	26,76	112,95	69
Multiple Query	RPS	26,59	11,6	20,43	17,72
	Bobot	1	1	1	1
	Skor	26,59	11,6	20,43	17,72
Total Skor		**239,44**	**88,44**	**359,95**	**241,14**

Total skor tertinggi diperoleh Phalcon-micro (359,95). Disusul oleh Slim (241,14), selanjutnya FatFree (239,44), kemudian Lumen (88,4). Phalcon-micro memperoleh skor tertinggi pada jenis pengujian JSON *serialization* dan *single query*. Khususnya pada pengujian *single query* Phalcon-micro jauh lebih unggul dari yang lain dengan skor 112,95. Kombinasi antara PHP dan C membuat Phalcon-micro mempunyai kecepatan diatas rata-rata. Fungsi yang ada pada *microframework* ini berjalan pada Phalcon PHP *extention,* sehingga bisa meminimalkan pengambilan *resouce* file dan memperpendek proses eksekusi sebuah perintah.

Angka RPS yang mendekati Phalcon-micro pada penelitian ini adalah FatFree. Bahkan pada pengujian *multiple query* FatFree mampu melampaui RPS Phalcon-micro. Salah satu fakor kunci kecepatan FatFree adalah mempunyai *resouce file* yang hanya berukuran 0,58 MB, lebih kecil diantara Lumen dan Slim. Disamping itu, arsitektur yang meminimalkan komponen struktural dan menghindari kompleksitas berjalan baik di *microframework* ini.

Analisa lain berkaitan antara RPS dan pembobotan, hasil yang cukup bagus justru ditunjukan oleh Slim. Angka RPS pada Slim tergolong standar tapi bobot yang dihasilkan lebih baik dibanding dengan FatFree. RPS dari FatFree menunjukan angka yang cukup tinggi tetapi rata-rata diperoleh pada jumlah koneksi kecil.

Disetiap jenis pengujian Lumen mendapatkan RPS terkecil. Banyaknya jumlah *resource file* yang digunakan *microframework* ini sangat berpengaruh pada performa. Berdasarkan RPS yang dihasilkan Slim dan Lumen tidak terlalu menonjol, akan tetapi pada setiap jenis pengujian mereka cenderung mempunyai angka RPS relatif stabil terhadap jumlah koneksi maupun iterasi yang diujikan.

B. **Grafik Perbandingan**

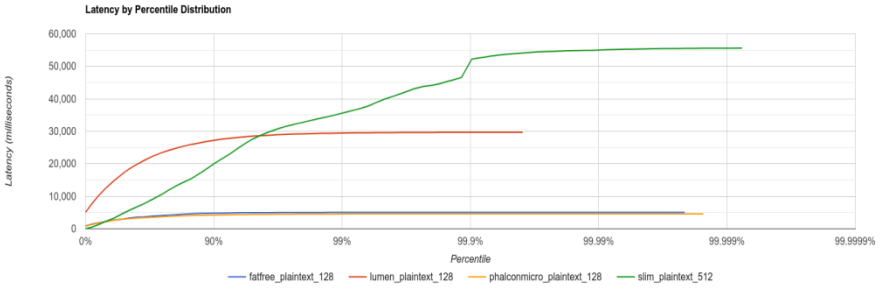
Melalui grafik perbandingan akan dijelaskan mengenai perbandingan *latency* terhadap *percentile* pada RPS tertinggi dari hasil pengujian performa. Hasil *latency* diperoleh dari aplikasi wrk2 pada tiap jenis pengujian. Gambar 4 adalah ringkasan hasil *latency* yang ditampilkan dari aplikasi uji Phalcon-micro. Data detil l*atency* dapat ditampilkan dengan menambahkan parameter --latency (-L) saat menjalankan perintah wrk2.

```
Running 1m test @ http://localhost:3333/single
8 threads and 16 connections
Thread calibration: mean lat.: 2975.101ms, rate sampling interval: 10526ms
Thread calibration: mean lat.: 2974.729ms, rate sampling interval: 10526ms
  Thread calibration: mean lat.: 2976.231ms, rate sampling interval: 10534ms
Thread Stats   Avg      Stdev     Max   +/- Stdev
Latency    19.73s     8.18s   34.21s    58.10%
Req/Sec   271.06      9.83   283.00     50.00%
Latency Distribution (HdrHistogram - Recorded Latency)
50.000%   19.55s
75.000%   26.48s
90.000%   31.56s
99.000%   33.95s
99.900%   34.21s
99.990%   34.21s
99.999%   34.21s
100.000%   34.24s

Detailed Percentile spectrum:
Value   Percentile   TotalCount 1/(1-Percentile)

5885.951     0.000000           9         1.00
8495.103     0.100000       10847         1.11
11272.191     0.200000       21664         1.25
14311.423     0.300000       32520         1.43
16449.535     0.400000       43356         1.67
34242.559     0.999991      108309    109226.67
34242.559     1.000000      108309          inf
#[Mean    =    19726.136, StdDeviation   =     8178.724]
#[Max     =    34209.792, Total count    =       108309]
#[Buckets =           27, SubBuckets     =         2048]
----------------------------------------------------------
128957 requests in 1.00m, 21.62MB read
```

```
Requests/sec:   2149.18
Transfer/sec:    369.03KB
```

Gambar 4. Contoh Hasil Latency Aplikasi Wrk2

Hasil pengujian direkam dalam bentuk *text file* untuk kemudian dikonversi dalam sebuah grafik. Data yang ditampilkan dalam bentuk grafik adalah data yang dihasilkan aplikasi uji dengan RPS tertinggi berdasarkan jenis uji. Garis dari aplikasi uji yang ketinggiannya paling mendekati angka 0 *latency* adalah aplikasi uji yang memiliki waktu *response* tercepat.

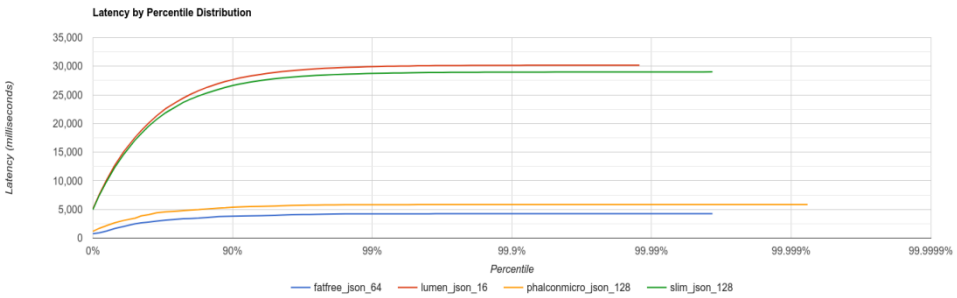


Gambar 5. Grafik Latency *Pada Jenis Pengujian Plaintext*

Gambar 5 aplikasi uji Slim mengalami peningkatan yang cukup signifikan terhadap *latency*. Peningkatan dimulai pada percentile 90% hingga pada puncaknya di percentile 99,9% dengan *latency* antara 20.000ms hingga 50.000ms lebih.

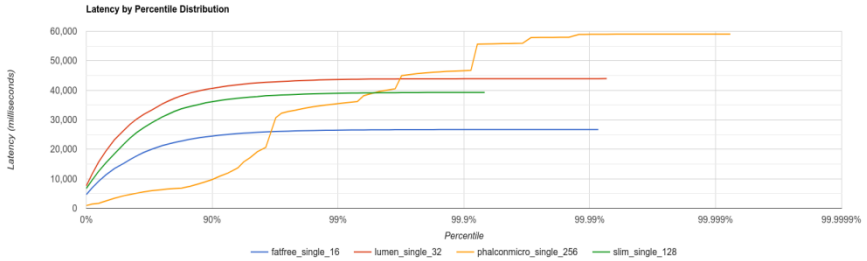
Hasil serupa juga ditunjukan pada Gambar 6, oleh Phalcon-micro. Aplikasi uji tersebut mengalami kenaikan *latency* yang ekstrim. Peningkatan besar terjadi pada *percentile* 90% hingga hampir menyentuh *percentile* 99,99% dengan selisih *latency* kurang lebih 50.000ms.

Peningkatan atau lonjakan besar yang terjadi pada pengujian tersebut menandakan aplikasi uji mempunyai kinerja kurang baik sekalipun mampu mencapai RPS tinggi dengan jumlah koneksi tertentu. Gambaran jika diimplementasikan secara nyata, pengguna aplikasi akan mengalami waktu tunggu untuk mendapatkan *response* dari proses yang terjadi.

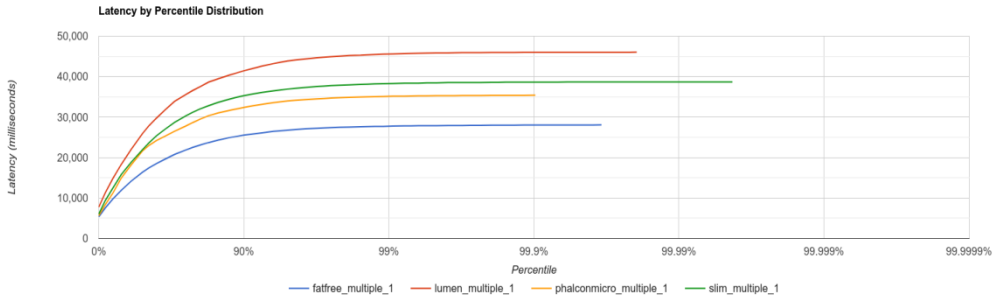


Gambar 6. Grafik Latency *Pada Jenis Pengujian JSON* Serialization

Hubungan *latency*, *percentile* dan akses *database* dengan eksekusi *query* dapat dilihat pada Gambar 7 dan Gambar 8. Kedua grafik tersebut menunjukan aplikasi uji membutuhkan waktu untuk menginisiasi koneksi ke *database* dan menjalanan proses pada program. Sehingga waktu mulai eksekusi mereka sedkit mengalami keterlambatan mengeluarkan *response*.



Gambar 7 Grafik Latency *Pada Jenis Pengujian* Single Query



Gambar 8 Grafik Latency Pada Jenis Pengujian Multiple Query

KESIMPULAN

Berdasarkan hasil pengujian yang sudah dikerjakan, terdapat beberapa kesimpulan yang dapat diambil diantaranya adalah:

1. Performa tertinggi berdasarkan RPS diperoleh *microframework* Phalcon-micro. Disusul Slim, kemudian FatFree dan terakhir Lumen.
2. Hasil *latency* terbaik dilihat dari kestabilan dan waktu response terkecil diperoleh oleh *microframework* FatFree.
3. RPS hasil dari pengujian *microframework* aplikasi uji Lumen dan Slim kurang begitu besar, akan tetapi keduanya mampu mendapatkan angka yang relatif stabil.
4. *Microframework* PHP dengan *resource file* berukuran kecil cenderung mempunyai performa yang lebih baik.
5. *Latency* yang dihasilkan *microframework* Slim pada pengujian plaintext di RPS tertinggi hasilnya kurang begitu baik, hal serupa dialami oleh Phalcon-micro pada pengujian *single-query*.
6. Angka terbaik pada aplikasi uji rata-rata didapat pada jumlah koneksi dibawah 128 dengan rate antara 2000 - 3000 RPS.

SARAN

Penelitian ini masih terdapat beberapa kekurangan yang bisa diperbaiki untuk kedepannya. Pertama, sebaiknya pengujian dilakukan menggunakan server (*dedicated* maupun *cloud*) yang terkoneksi internet dengan perangkat lunak server pada pengaturan

grade production. Hal tersebut diperlukan untuk menghasilkan ukuran performa yang sesuai dengan kondisi riil di lapangan.

Pembobotan akhir hasil pengujian bisa diganti dengan metode yang lebih tepat. Harapannya dengan metode pembobotan yang lebih tepat, pengukuran skor dengan parameter, jenis, atau variable lain pada pengujian bisa mendapatkan angka yang lebih sesuai dan akurat.

Mengenai alur pengujian, terutama persiapan sebelum pengujian. Sebaiknya didokumentasikan, baik itu berhubungan dengan teknis maupun non-teknis. Hal tersebut mungkin akan bermanfaat bagi peneliti selanjutnya. Beberapa contoh persiapan pengujian yang bisa didokumentasikan meliputi: cara penentuan indikator pengujian, cara penentuan parameter uji, rumus dan lainnya.

DAFTAR PUSTAKA

[1] Agung Rahmat Saleh. 2010. *Analisis Dan Implementasi Perbandingan Framework Yii Dan Zend. Skripsi.* Universitas Telkom. Bandung.

[2] Angga Ibnu Saputra. 2015. *Benchmarking Teknologi Framework Yii dengan Pemrograman PHP Konvensional. Skripsi.* STMIK AKAKOM. Yogyakarta.

[3] Delipetrev, Mile Janev and Ristova Suzana, 2015, *Performance Benchmark of PHP Frameworks with Database Select Method,* Proceeding, IX Internasional Conference for Young Researchers. Burgas.

[4] Jose Sandoval, 2009., RESTful Java Web Services, Pack Publishing

[5] Muhammad Joddy Gorby Cendraraja. 2013. *Analisis Perbandingan Framework PHP Antara Framework Codeigniter dengan Framework Yii Menggunakan Metode Penelitian Kualitatif. Skripsi.* Universitas Esa Unggul. Jakarta.

[6] Muhammad Nur Hamid. 2020. *Analisis Perbandingan Framework Codeigniter Dan Framework Laravel (Studi Kasus Inventaris Hmj Ti Stmik Akakom Yogyakarta)*, *Skripsi*, STMIK AKAKOM, Yogyakarta.

[7] Wahyu Rifa'i Dwi Septian. 2010. *Analisis Perbandingan Framework PHP Berdasarkan Moose CK dan Properti Kualitas Disain Menggunakan Metode Analytic Hierarchy Process (AHP). Skripsi.* UIN Syarif Hidayatullah. Jakarta.