BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Pada bab ini akan menyajikan tinjauan dari penelitian sebelumnya yang membahas analisis dan perbandingan pada *framework*, bahasa pemrograman, ataupun web *server* dari aspek performa dengan berbagai parameter yang relevan pada penelitian ini. Selain itu pada bab ini juga akan membahas teori-teori dari berbagai sumber terkait penelitian.

Tabel 2.1 Perbandingan Penelitian Sebelumnya.

No	Judul Penelitian	Objek	Parameter	Alat
1	Analisis Kinerja Komputasi Terdistribusi Dengan *Platform* Web *Service* Menggunakan Metode REST (Yogiswara, 2014)	REST *server* atau *web service*	*Latency.*	Tidak memakai alat.
2	Analisis Performasi *Framework* Codeigniter Dan Laravel Menggunakan Web *Server* Apache (Erinton et al., 2017)	Objek pada penelitan ini tidak disebutkan.	*Time, speed,* s*ize,* dan *error.*	Web *Stress Tool.*

Tabel 2.2 Perbandingan Penelitian Sebelumnya.

3	*Comparative Analysis Of* Codeigniter *And* Laravel *In Relation To Object-Relational Mapping, Load Testing And Stress Testing* (Ibrahim et al., 2018)	*E-Commerce.*	*Response time,* dan *throughput.*	J-Meter.
4	Perbandingan Performa Kinerja Node.Js, PHP, Dan Python Dalam Aplikasi REST (Rompis dan Aji, 2018)	Web *service /* REST *server* informasi jasa transportasi.	Penggunaan CPU, RAM, dan *response time.*	J-Meter.

Tabel 2.3 Perbandingan Penelitian Sebelumnya.

5	Analisis Perbandingan Performa Web *Service* REST Menggunakan *Framework* Laravel, Django, Dan Ruby *On Rails* Untuk Akses Data Dengan Aplikasi *Mobile* (Saputra dan Aji, 2018)	*Web service /* REST *Service* Portal E-Kampus.	Penggunaan CPU dan RAM pada *server,* serta *response time client* atau kecepatan eksekusi perintah pada web *service.*	Alat pada penelitian ini tidak disebutkan.
6	*A Comparative Study Of* PHP *Frameworks Performance* (Laaziri et al., 2019)	Objek pada penelitan ini tidak disebutkan.	*QSOS, request per second, memory usage,* dan *response time.*	Apache *Benchmark* dan QSOS *method.*

Tabel 2.4 Perbandingan Penelitian Sebelumnya.

7	Analisis Perbandingan *Framework* Larvel Dan *Framework* Codeigniter : Studi Kasus Inventaris HMJ TI STMIK AKAKOM Yogyakarta (Hamid, 2019)	*Website* inventaris.	*Response time, throughput, request per second,* cara akses *database,* dan implementasi fitur AJAX.	site24x7.com, dan Apache *Benchmark.*
8	Studi Komparasi Pengembangan *Website* Dengan *Framework* Codeigniter *D*an Laravel (Prasena dan Sama, 2020)	Objek pada penelitan ini tidak disebutkan.	*Time, speed,* dan s*ize*.	Web *Stress Tool.*

Pada penelitian tugas akhir ini mempunyai beberapa perbedaan dan relevansi dengan penelitian sebelumnya, berikut ini adalah penjelasan tentang perbedaan dan relevansi yang ada:

1. Pada penelitian dengan judul Analisis Kinerja Komputasi Terdistribusi Dengan *Platform* Web *Service* Menggunakan Metode REST (Yogiswara, 2014) melakukan analisis performa pada web *service* yang menggunakan *server* apache, nginx, dan iis dengan metode REST pada sisi *server* dengan *latency* sebagai parameter. Sedangkan pada penelitian ini dilakukan pada *framework* Laravel dan Codeigniter dengan menghitung waktu eksekusi transaksi data menggunakan arsitektur REST pada sisi *client*.

2. Pada penelitian dengan judul Analisis Performasi *Framework* Codeigniter Dan Laravel Menggunakan Web *Server* Apache (Erinton et al,, 2017) menggunakan alat web *stress tool* pada pengujian performa, dengan parameter *Time, speed,* s*ize,* dan *error*. Secara harafiah parameter *time* dan *speed* pada alat web *stress tool* memiliki kemiripan dengan parameter *response time* dan *throughput* yang akan diuji pada penelitian ini menggunakan *website monitoring* site4x7.com. Perbedaan pada penelitian ini ada pada penambahan parameter yaitu *page load time, request per second,* dan waktu eksekusi transaksi data menggunakan arsitektur REST, serta objek dan alat pengujian yang digunakan pada penelitian ini berbeda.

3. Pada penelitian dengan judul *Comparative Analysis Of* Codeigniter *And* Laravel *In Relation To Object-Relational Mapping, Load Testing And Stress Testing* (Ibrahim et al., 2018) menggunakan parameter *Response time,* dan *throughput* yang menggunakan J-Meter sebagai alat pengujian, serta menggunakan objek yang sama yaitu *e-commerce*. Perbedaan pada penelitian ini terdapat pada penambahan parameter yaitu *page load time, request per second,* dan waktu eksekusi transaksi data menggunakan arsitektur REST, serta penelitian ini juga terdapat perbedaan pada alat pengujian.

4. Pada penelitian dengan judul Perbandingan Performa Kinerja Node.Js, PHP, Dan Python Dalam Aplikasi REST (Rompis dan Aji, 2018) melakukan pengujian pada performa web *service* dengan penggunaan CPU, RAM, dan *response time* sebagai parameter, penelitian ini menggunakan J-Meter sebagai alat pengujian, dan objek yang digunakan pada penelitian ini adalah web *service* informasi jasa transportasi. Relevansi pada penelitian ini adalah *response time* yang digunakan sebagai parameter pada pengujian performa.

5. Pada penelitian dengan judul Analisis Perbandingan Performa Web *Service* REST Menggunakan *Framework* Laravel, Django, Dan Ruby *On Rails* Untuk Akses Data Dengan Aplikasi *Mobile* (Saputra dan Aji, 2018) melakukan pengujian pada performa web *service* dengan penggunaan CPU, RAM, dan waktu eksekusi transaksi data pada sisi *client* (Android) sebagai parameter.

Relevansi pada penelitian ini adalah menggunakan parameter waktu eksekusi transaksi data pada sisi *client* yang terhubung dengan web *service.*

6. Pada penelitian dengan judul *A Comparative Study Of* PHP *Frameworks Performance* (Laaziri et al., 2019) melakukan pengujian pada performa *framework* Laravel, Symfony, dan Codeingiter dengan menggunakan parameter *request per second, memory usage,* dan *response time,* serta melakukan evaluasi *software open source* dengan QSOS method. Relevansi penelitian ini adalah melakukan pengujian pada *framework* Laravel dan Codeigniter dengan menggunakan *request per second* dan *response time* sebagai parameter, dan juga menggunakan alat yang sama dalam pengujian yaitu Apache *Benchmark.*

7. Pada penelitian dengan judul Analisis Perbandingan *Framework* Larvel Dan *Framework* Codeigniter : Studi Kasus Inventaris HMJ TI STMIK AKAKOM Yogyakarta (Hamid, 2019) melakukan penelitian menggunakan parameter *request time, throughput, dan request per second* dalam pengujian performa serta membandingkan penggunaan *database,* dan implementasi fitur AJAX. Pengujian performa menggunakan alat Apache *Benchmark,* dan site24x7.com. Sedangkan pada penelitian ini akan berfokus pada performa, dengan perubahan dan juga penambahan parameter yaitu *page load time*, dan waktu eksekusi transaksi data menggunakan arsitektur REST.

8. Pada penelitian dengan judul Studi Komparasi Pengembangan *Website* Dengan *Framework* Codeigniter *D*an Laravel (Prasena dan Sama, 2020) menggunakan alat web *stress tool* pada pengujian performa, dengan parameter *Time, speed,* dan s*ize*. Secara harafiah parameter *time* dan *speed* pada alat web *stress tool* memiliki kemiripan dengan parameter *response time* dan *throughput* yang akan diuji pada penelitian ini menggunakan *website monitoring* site4x7.com. Perbedaan ada pada penambahan parameter pada penelitian ini yaitu *page load time*, *request per second*, dan waktu eksekusi transaksi data menggunakan arsitektur REST, serta objek dan alat pengujian yang digunakan pada penelitian ini berbeda.

Secara umum perbedaan dan relevansi penelitian ini dibandingkan dengan penelitian-penelitian sebelumnya terletak pada objek yang digunakan, bahasa pemrograman dan *framework,* parameter, serta penggunaan *tool* pada saat pengujian performa.

2.2 Landasan Teori

Pada landasan teori akan membahas teori-teori dari berbagai sumber yang berkaitan dengan dilakukanya penelitian ini.

2.2.1 *E-Commerce*

Electronic commerce atau selanjutnya disebut *e-commerce* merupakan salah satu hasil dari perkembangan teknologi internet. Pengertian *e-commerce* itu sendiri adalah suatu proses berbisnis dengan menggunakan teknologi elektronik yang menghubungkan antara perusahaan, konsumen, dan masyarakat dalam bentuk transaksi elektonik. Dengan demikian pada prinsipnya bisnis dengan *e-commerce* adalah bisnis tanpa warkat *paperless trading* (Munir Fuady, 2002).

Sedangkan menurut pendapat lain, *e-commerce* adalah proses menjual dan membeli suatu produk secara elektronik oleh konsumen, dari suatu perusahaan ke perusahaan lainnya melalui transaksi bisnis yang terkomputerisasi (Laudon dan Laudon, 1998).

Berdasarkan pendapat di atas dapat dikatakan bahwa *e-commerce* merupakan sebuah transaksi jual-beli yang melibatkan perusahaan, konsumen, dan masyarakat dengan menggunakan media elektronik sebagai perantara.

2.2.2 PHP (*Hypertext Preprocessor*)

PHP adalah bahasa *server-side-scripting* yang menyatu dengan HTML untuk membuat halaman web yand dinamis. Karena PHP merupakan *server-side-scripting* maka *sintaks* dan perintah-perintah PHP akan dieksekusi di *server* kemudian hasilnya akan dikirimkan ke *browser* dengan format HTML (Arief, 2011c:43).

Sedangkan pendapat lain mengatakan PHP merupakan salah satu bahasa pemrograman yang berjalan pada sisi *server* atau sering disebut dengan *server-side.* PHP hanya bisa berjalan pada sisi *server* karena untuk menjalankan *file* dengan

ekstensi .php (dot php) haruslah menggunakan web *server* (Chastro dan Darmawan, 2020).

Dari beberapa pendapat yang telah diuraikan dapat disimpulkan bahwa PHP merupakan bahasa pemrograman yang hanya bisa berjalan pada sisi *server* yang dapat di eksekusi bersama HTML untuk membuat *website*. PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995 dan bersifat *open source* sehingga setiap orang dapat berkontribusi dalam mengembangkan bahasa pemrograman ini.

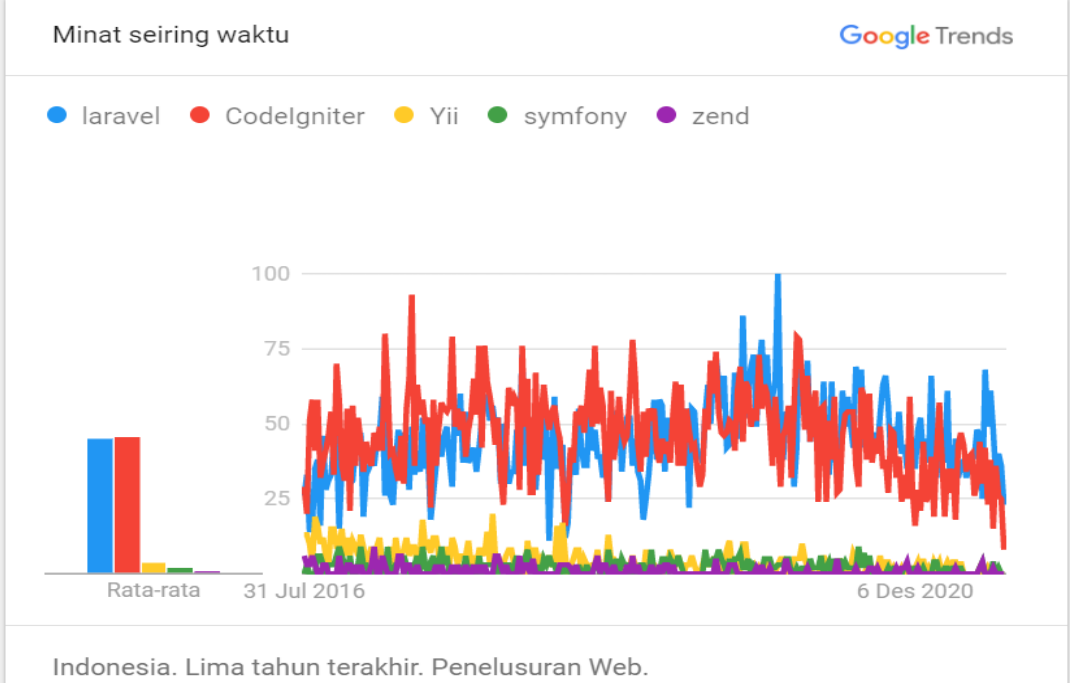
2.2.3 *Framework*

Menurut Siena (2009) *framework* adalah suatu *library* yang telah diorganisasikan pada sebuah rancangan arsitektur untuk memberikan kecepatan, ketepatan, kemudahan, dan konsisten dalam pengembangan suatu aplikasi.

Dan menurut Renaldo dan Sama (2020) *framework* merupakan sebuah kerangka kerja yang dibuat untuk memudahkan pembuatan *website*.

Dari pendapat yang telah diuraikan dapat disimpulkan bahwa *framework* pada umumnya terdapat fungsi-fungsi yang sudah dibangun untuk memudahkan penggunaan *script* dan memiliki struktur yang memudahkan *developer* dalam *maintenance* ataupun *debugging*. Jadi dengan adanya *framework* para *developer* dapat membangun sebuah aplikasi dengan waktu yang lebih singkat.

Pada bahasa pemrograman PHP terdapat berbagai *framework*, seperti Laravel, Codeigniter, Symfony, CakePHP, Zend, Yii, dan masih banyak lagi. Namun menurut Google *Trends* selama lima tahun terakhir yang paling populer dan banyak diminati para web *developer* di Indonesia adalah *framework* Laravel dan *framework* Codeigniter. Dibawah ini adalah gambar grafik yang menunjukan kepopuleran *framework* tersebut berdasarkan Google *Trends* dari hasil penulusuran web.



Gambar 2.1 Data Peminat *Framework* PHP Lima Tahun Terakhir.

Dapat dilihat dari gambar 2.1 *framework* Laravel dan Codeigniter sangat populer dibandingkan *framework* lainnya selama 5 tahun terakhir. Oleh karena itu penulis ingin membangun *website e-commerce* dengan performa yang baik menggunakan salah satu dari kedua *framework* tersebut untuk digunakan oleh PT Sinar Agung Prasadikindo. Dibawah ini merupakan penjelasan mengenai *framework* Laravel dan Codeigniter.

2.2.3.1 *Framework* Laravel

Laravel merupakan *framework* PHP yang menekankan pada kesederhanaan dan fleksibilatas pada desainnya. Laravel dirilis dibawah lisensi MIT dengan sumber kode yang disediakan di Github. Sama seperti *framework* PHP lainnya, Laravel dibangun dengan basic MVC (*Model View Controller*). Laravel dilengkapi *command line tool* yang bernama "Artisan" yang bisa digunakan untuk *packaging bundle* dan instalasi *bundle*. Menurut survey yang dilakukan oleh Sitepoint.com pada Desember 2013 dalam popularitas *framework* PHP, Laravel menduduki urutan teratas (Erinton et al., 2017).

Sedangkan pendapat penulis, Laravel merupakan salah satu *framework* PHP yang digunakan untuk membuat sebuah *website*. Laravel bersifat *open source* yang dapat digunakan oleh siapapun dan setiap orang dapat berkontribusi dalam perkembangan Laravel. Laravel pertama kali dibuat oleh Taylor Otwell pada tahun 2011, *framework* ini dibangun menggunakan konsep MVC (*Model View Controller*) dan yang membuat Laravel ini sangat populer adalah karena *framework* ini kaya akan fitur yang dimilikinya, sehingga waktu pengembangan *website* menjadi lebih singkat.

2.2.3.2 *Framework* Codeigniter

Codeigniter adalah sebuah web *application framework* yang digunakan untuk membangun aplikasi PHP dinamis yang dibangun menggunakan konsep *Model View Controller development pattern.* Codeigniter menyediakan berbagai macam *library* yang dapat mempermudah dalam pengembangan dan termasuk *framework* tercepat dibandingkan dengan *framework* lainnya (Erinton et al., 2017).

Sedangkan pendapat penulis, Codeigniter merupakan salah satu *framework* alternatif yang dimiliki PHP selain Laravel. Sama seperti Laravel, Codeigniter bersifat *open source* dan menggunakan konsep MVC (*Model View Controller*). Codeigniter pertamakali dikembangkan oleh Rick Ellis pada tahun 2006, dan yang membuat *framework* ini populer adalah karena ukurannya yang sangat kecil, dokumentasi yang ramah untuk pemula, dan diklaim sangat cepat oleh pencipta PHP yaitu Rasmus Lerdorf pada konverensi FrOSCon di tahun 2008.

2.2.4 www.site24x7.com

Dari situs resminya mengatakan bahwa site24x7.com dilahirkan ketika keahlian IT dari ManageEngine, yaitu sederetan *software* manajement IT kelas dunia dan Zoho, sebagai sebuah SaaS terkemuka untuk aplikasi bisnis dan produktivitas, bersama membangun sebuah solusi untuk IT dan DevOps. Sebuah solusi pemantauan yang lengkap untuk aplikasi skala *cloud* yang dapat memantau situs web, *server*, aplikasi, jaringan, *cloud*, dan masih banyak lagi. Site24x7 melakukan

pemantauan berdasarkan metrik, jejak, dan log di bawah satu konsol untuk berbagai lapisan arsitektur *cloud*. Salah satu fitur yang ada pada *platform* ini adalah *website monitoring*. Oleh karena itu penulis akan melakukan pengujian menggunakan *platform* ini sebagai *tool* untuk melakukan pengujian performa menggunakan parameter *response time, throughput,* dan *page load time*. Dibawah ini akan menjelaskan apa itu *response time, throughput,* dan *page load time*.

2.2.4.1 *Response Time*

Waktu tanggap (*Response Time*) adalah waktu yang dibutuhkan oleh suatu proses dari minta dilayani hingga ada respon pertama yang menanggapi permintaan tersebut (Excalanta et al., 2012).

Sedangkan menurut situs resmi site24x7, *response time* merupakan waktu yang diperoleh dari hasil penjumlahan waktu pencarian DNS, waktu koneksi, waktu *handshake* SSL, dan waktu yang dibutuhkan untuk mengunduh seluruh konten HTML. Namun itu tidak termasuk mendapatkan gambar dan sumber daya lainnya yang dimuat dalam halaman HTML.

Berdasarkan penjelasan di atas dapat dikatakan *response time* atau waktu tanggap merupakan waktu yang dibutuhkan sebuah proses untuk mendapatkan *response* dari *server* berupa seluruh konten HTML, akan tetapi tidak termasuk gambar dan sumber lainnya yang berada pada halaman HTML tersebut.

2.2.4.2 *Throughput*

Throughput adalah jumlah kerja yang dapat diselesaikan dalam satuan waktu tertentu. Lebih tinggi angka *throughput*, lebih banyak kerja yang dilakukan sistem (Excalanta et al., 2012).

Sedangkan menurut situs resmi site24x7, *throughput* merupakan hasil perhitungan dengan membagi *content length* (dalam satuan KB) dengan *response time* (dalam satuan detik) selama periode waktu tertentu. Data ini berguna untuk menganalisis kecepatan koneksi serta penggunaan *bandwith*.

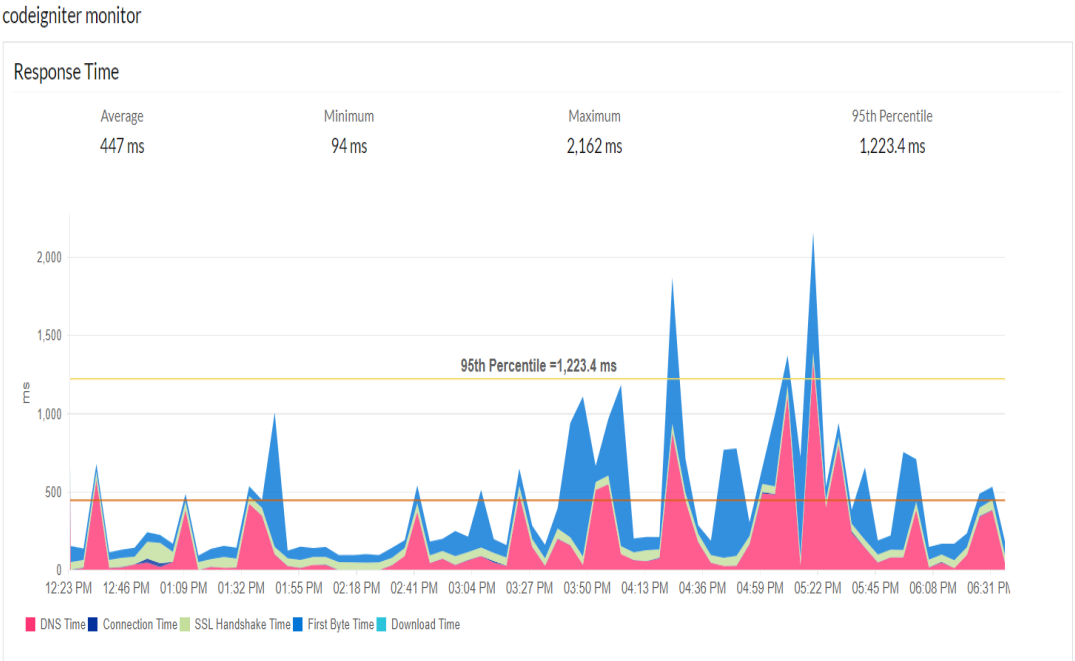
Dari penjelasan diatas dapat disimpulkan bahwa *throughput* merupakan kecepatan yang dibutuhkan untuk menyelesaikan suatu proses (dalam satuan KB/sec).

2.2.4.3 *Page Load Time*

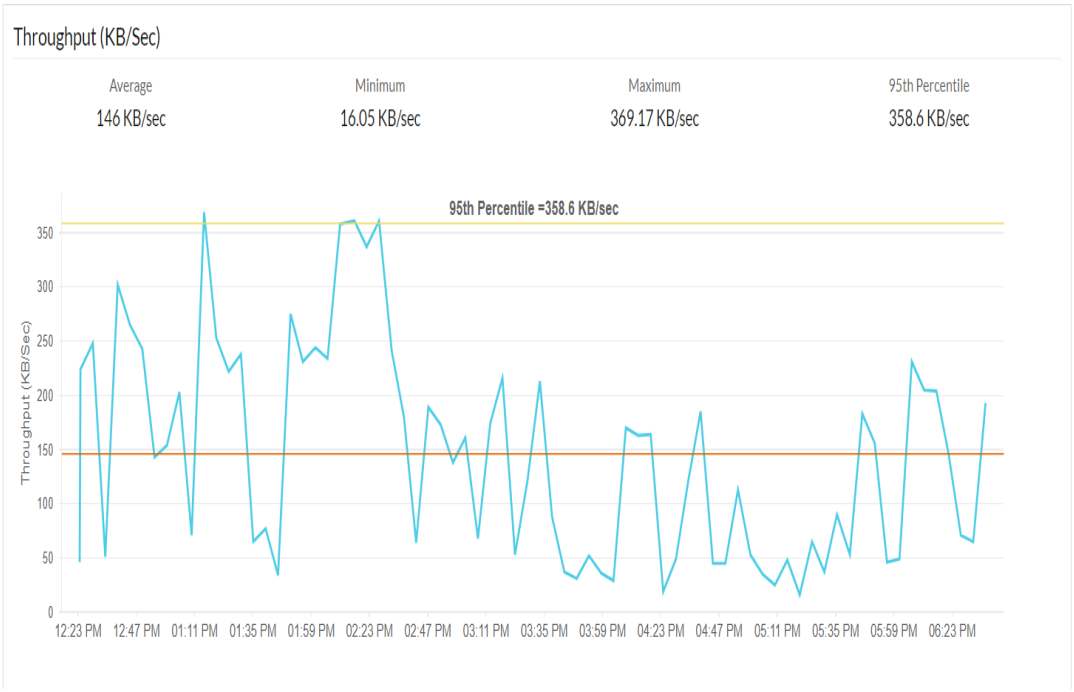
Menurut situs resmi site24x7 *page load time* adalah waktu untuk memuat seluruh konten dari respon. Seluruh konten yang dimaksud adalah HTML, JavaScript, CSS, gambar, dan lain-lain.

Jadi dapat dikatakan bahwa *page load time* merupakan waktu yang dibutuhkan sebuah *website* untuk menampilkan seluruh konten yang ada pada halaman, sampai halaman tersebut siap melakukan interaksi dengan pengguna.

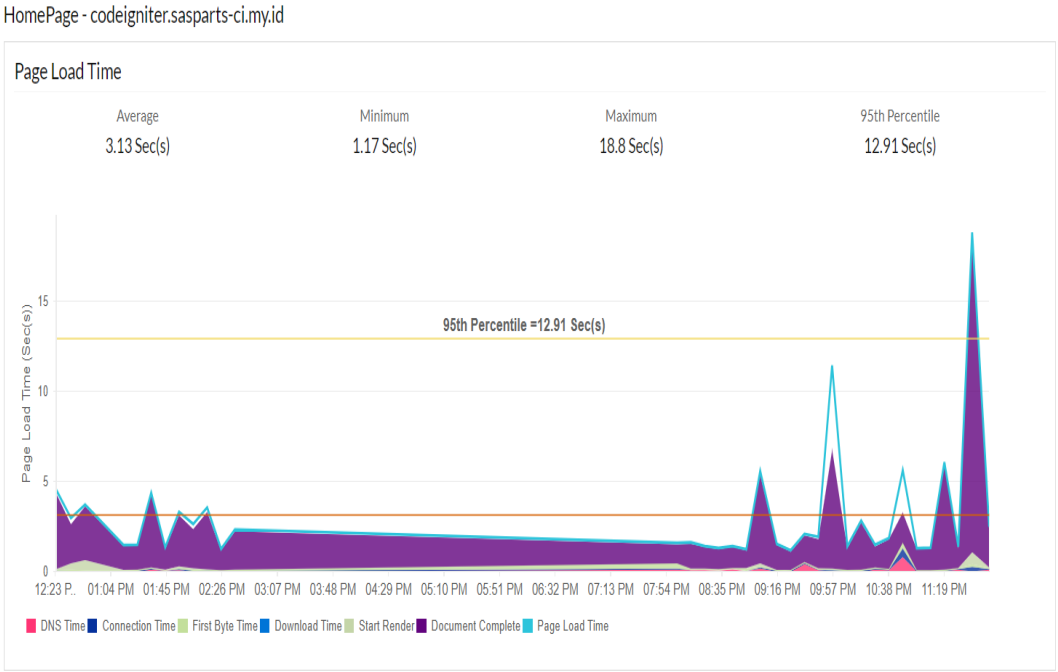
Untuk memberikan gambaran pada penggunaan site24x7, dibawah ini penulis menyertakan gambar contoh hasil pengujian *response time, throughput,* dan *page load time* pada *website e-commerce* yang menggunakan *framework* Codeigniter.



Gambar 2.2 Hasil *Response Time Website E-Commerce* menggunakan Codeigniter.



Gambar 2.3 Hasil *Throughput Website E-commerce* menggunakan Codeigniter.



Gambar 2.4 Hasil *Page Load Time Website E-Commerce* menggunakan Codeigniter.

Dari contoh hasil pengujian pada *website e-commerce* yang dibangun menggunakan *framework* Codeigniter pada gambar 2.2, 2.3, dan 2.4 menunjukan bahwa *website* tersebut memiliki *response time* dengan rata-rata 447 ms, *throughput* dengan nilai rata-rata 146 KB/s, dan *page load time* dengan nilai rata-rata 3,13 s.

2.2.5 Apache *Benchmark*

Apache *Benchmark* atau disingkat ab adalah sebuah *tool* berbasis *command line* untuk melakukan *load tests* sederhana pada *server* HTTP, baik itu situs web ataupun API (Diamantidis, 2020).

Pendapat lain mengatakan Apache *Benchmark* merupakan sebuah *tool* yang melakukan *benchmark* dengan mengirimkan *request* ke *server* HTTP setiap detik dengan tujuan untuk mengetahui kemampuan *server* dalam menangani *request*, terutama dalam jumlah data yang banyak dan konkuren (Gunawan dan Halim, 2011).

Dari pendapat di atas dapat disimpulkan bahwa, Apache *Benchmark* adalah *tool* yang digunakan untuk menguji performa *website* dengan mengirimkan sejumlah *request* ke *server*. Berikut ini adalah perintah yang digunakan untuk menjalankan Apache *Benchmark*:

ab –n x –c y [URL]

Diketahui:

ab = Perintah untuk menjalankan Apache *Benchmark*
-n = Perintah untuk mengirimkan *request*
x = Jumlah *request* yang ingin dikirim
-c = Perintah melakukan *request* dalam waktu bersamaan (*concurrency)*
y = Jumlah *request* dalam waktu bersamaan yang ingin dikirim
URL = Alamat *website* yang akan dikirimkan *request* (*website* yang akan diuji)

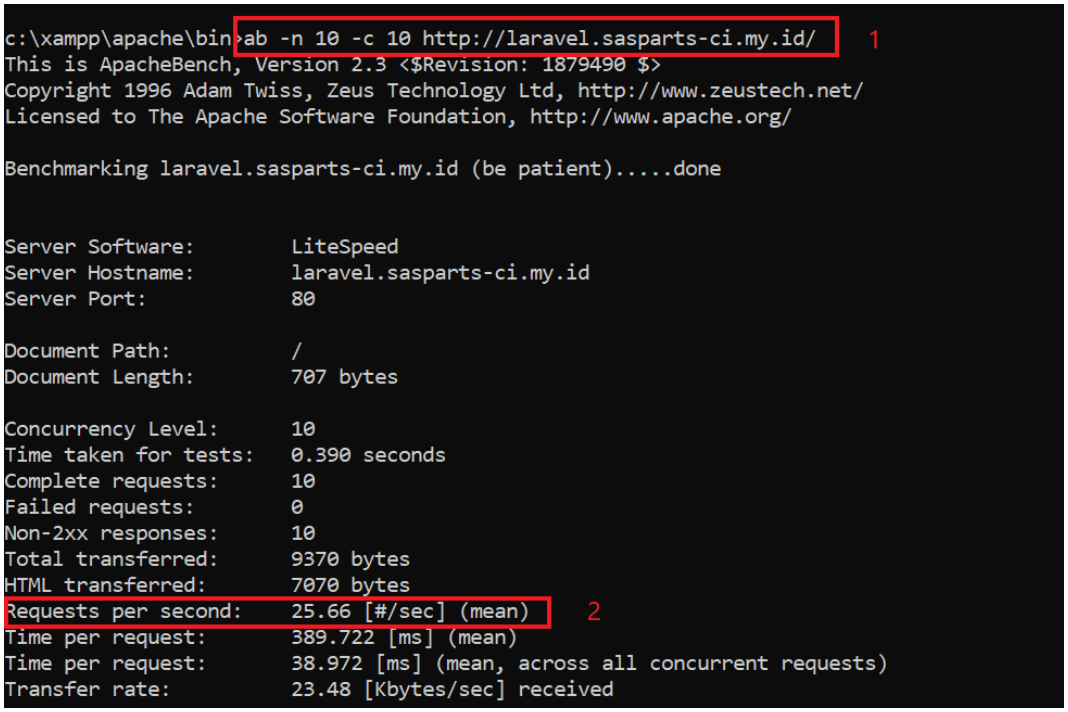
Pada penelitian ini parameter yang akan digunakan pada pengujian performa menggunakan *tool* Apache *Benchmark* adalah *request per second*. Berikut ini adalah penjelasan tentang *request per second*:

2.2.5.1 *Request Per Second*

RPS (*Request per second*) berarti berapa banyak *request* yang dapat diproses per detik. Semakin tinggi angkanya maka semakin efisien *framework* tersebut (Shiddieq dan Aqmarina, 2013).

Menurut situs resmi Apache *request per second* adalah jumlah *request* per detik, dimana nilai ini merupakan hasil pembagian jumlah permintaan dengan total waktu yang dibutuhkan.

Jadi dapat disimpulkan bahwa *request per second* merupakan kemampuan sebuah *website* dalam menyelesaikan sejumlah *request* dalam waktu satu detik.



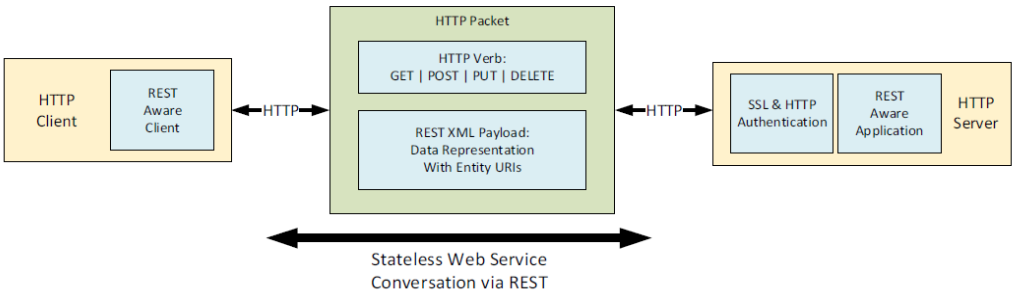
Gambar 2.5 Pengujian *Request Per Second.*

Dari contoh hasil pengujian pada *website e-commerce* yang dibangun menggunakan *framework* Laravel pada gambar 2.5, dimana pada baris yang ditandai kotak warna merah dengan angka 1 merupakan perintah untuk melakukan pengujian dengan mengirimkan 10 *request* pada saat waktu yang sama ke situs laravel.sasparts-ci.my.id (*website e-commerce*), lalu pada kotak merah dengan angka 2 menunjukan bahwa *website* tersebut memiliki *request per second* sebanyak 25,66.

2.2.6 REST (*Representational State Transfer*)

Representational State Transfer (REST) merupakan pendekatan yang menjadikan fungsi-fungsi aplikasi menjadi *service*, yang dipaket sebagai bagian yang dapat digunakan ulang. Pada teknologi ini setiap sumber daya akan diberikan sebuah ID yang unik (misalnya, URL dokumen) yang memungkinkan sumber daya dapat berhubungan menggunakan standar (HTTP, HTML, XML). REST terletak pada kemampuanya mengakses sumber (*source*) melalui alamat unik yang dapat mengidentifikasi data yang dibutuhkan. Hanya dengan mengkontruksi URL (*Uniform Resource Locator*) maupun URI (*Uniform Resource Identifier*) pada suatu layanan web (*web service*) (Yogiswara, 2014).

Sedangkan menurut (Saputra dan Aji, 2018) REST (*Representational State Transfer*) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web atau sistem terdistribusi. Istilah REST diperkenalkan oleh Roy Fielding pada tahun 2000. Arsitektur gaya REST adalah arsitektur *client server* di mana *client* mengirimkan permintaan ke *server*, *server* kemudian memproses permintaan dan mengembalikan tanggapan. Umumnya menggunakan HTTP sebagai protokol untuk komunikasi data.



Gambar 2.6 Arsitektur REST.

a) HTTP *client*, merupakan *client* yang akan mengakses web *service*.

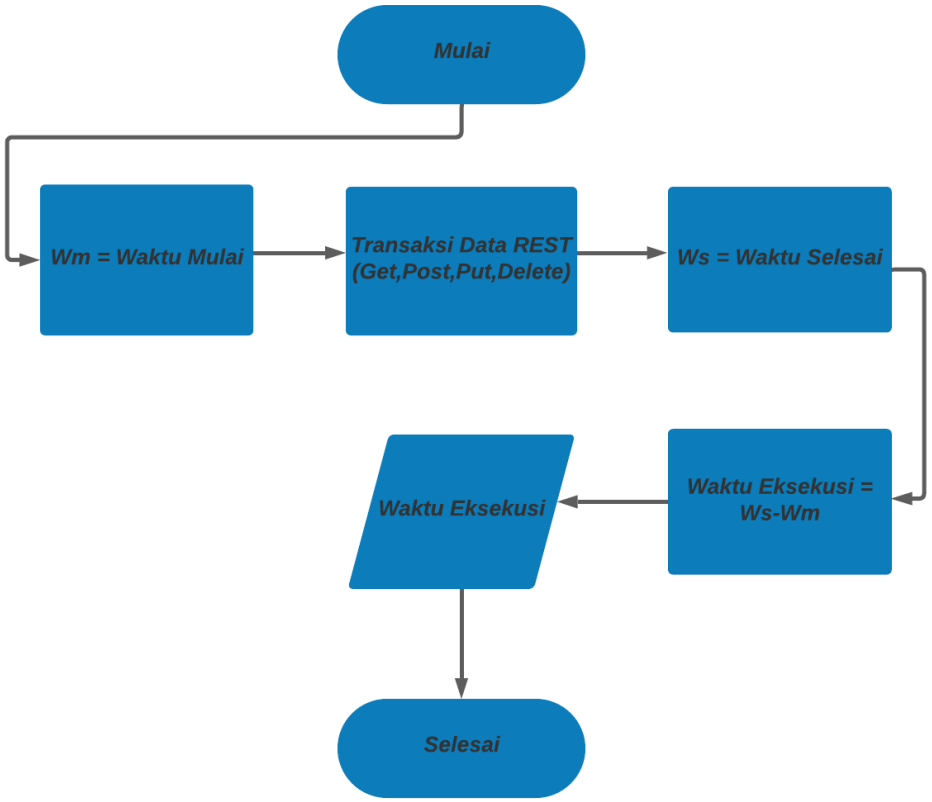
b) HTTP *packet*. HTTP *packet request* dapat berupa *GET, POST, PUT, DELETE*. Sedangkan HTTP *packet response* dapat berupa XML maupun JSON *payload*. Pada umumnya menggunakan JSON. JSON (*Javascript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca

dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer.

c) HTTP *server*, merupakan *server* tempat web *service* diimplementasikan.

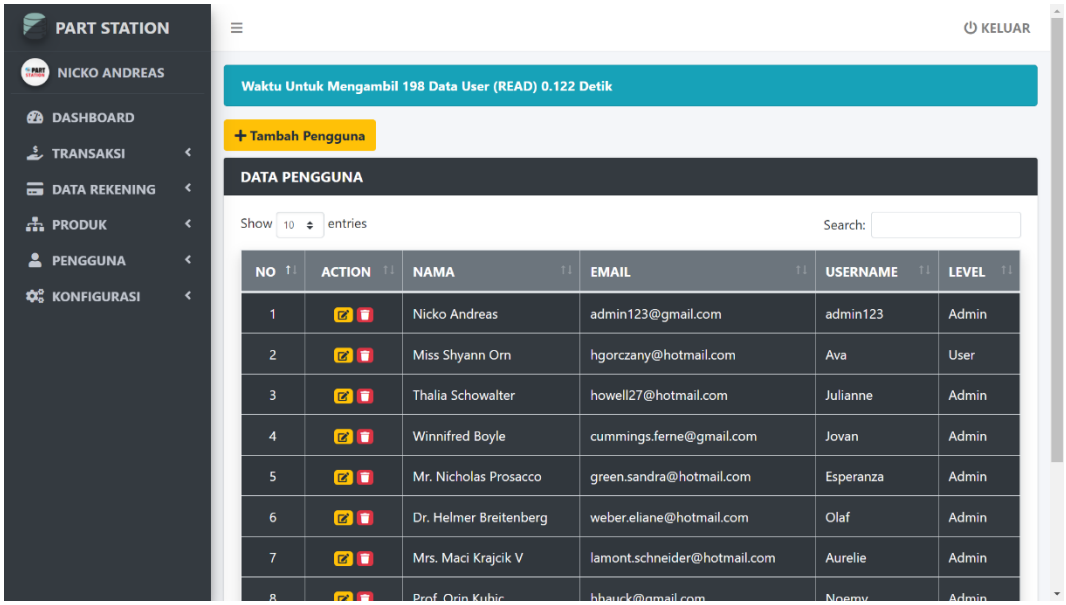
Jadi dapat disimpulkan bahwa REST merupakan gaya arsitektur untuk melakukan transaksi data antara dua sistem atau lebih. Transaksi data pada arsitektur REST biasanya dilakukan oleh *client* yang berperan mengirimkan *request* dan *server* yang berperan menerima *request* dan mengembalikan *response* berupa data yang diminta oleh *client*, data/*response* yang dikirimkan oleh *server* pada umumnya berformat JSON. Sebuah REST *server* atau web *service* biasanya memiliki *identifier* yang unik berupa URL atau juga disebut *endpoint* untuk digunakan oleh *client* dalam melakukan *request*. Sehingga pada saat *client* melakukan *request* harus menggunakan *endpoint* yang benar, supaya *server* mengenali *endpoint* tersebut dan memberikan *response* yang tepat.

Pada penelitian ini, salah satu parameter yang digunakan untuk menganalisis dan membandingkan *framework* Laravel dan *framework* Codeigniter dari aspek performa adalah waktu eksekusi transaksi data dengan arsitektur REST. Transaksi data akan dilakukan pada sebuah *web service/*REST *server* yang mengolah data *user*. Hal ini dilakukan untuk menguji performa masing-masing *framework* dalam melakukan transaksi data dengan sebuah web *service* atau REST *server*, karena sebuah *website e-commerce* pasti akan terlibat dengan web *service* seperti *payment gateway* dan penyedia informasi ongkos kirim (ongkir). Untuk memberi gambaran tentang perhitungan pada pengujian waktu eksekusi transaksi data dengan arsiterktur REST, penulis menyertakan *flowchart* sebagai berikut :



Gambar 2.7 *Flowchart* Pengujian Waktu Eksekusi Transaksi Data REST.

Dapat dilihat pada gambar 2.7, waktu eksekusi yang menjadi salah satu parameter pada penelitian ini, dihasilkan dari perkurangan antara waktu selesai dengan waktu mulai pada proses transaksi data. Pada saat proses transaksi data dilakukan, penulis menggunakan sebuah *library* PHP yaitu cURL untuk melakukan *request* pada *web service* yang mengolah data *user*. Dan pengujian waktu eksekusi transaksi data ini, akan dilakukan pada setiap jenis transaksi atau CRUD (*Create, Read, Update, Delete*). Berikut ini adalah contoh pengujian waktu eksekusi transaksi data dengan method *GET* (*Read*).

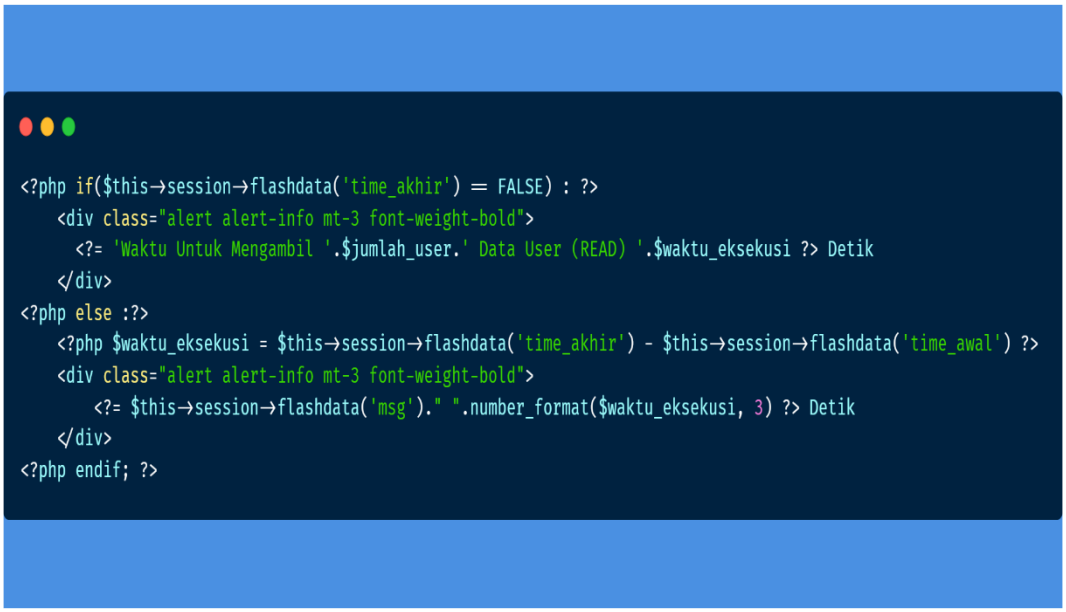


Gambar 2.8 Pengujian Transaksi Data Dengan *Method GET* (*Read*).

Pada gambar 2.8 kita dapat melihat contoh hasil pengujian, dimana ada sebuah *alert* berwarna biru yang memberitahukan bahwa waktu yang dibutuhkan untuk mengambil 198 *user* (semua data) adalah 0,122 detik. Pada pengujian waktu eksekusi transaksi data ini tidak ada antarmuka khusus, sehingga pengujian dilakukan secara langsung pada menu pengguna. Hal ini pun sama pada transaksi data *Create, Update, Delete*, dimana akan menampilkan *alert* yang menunjukan waktu eksekusi, ketika transaksi data selesai dilakukan. Berikut ini adalah *script* yang digunakan dalam pengujian berdasarkan *flowchart* pada gambar 2.7.

```
$waktu_mulai    = microtime(true);
$user           = $this->Userapi_model->listing();
$waktu_selesai  = microtime(true);
$waktu_eksekusi = $waktu_selesai - $waktu_mulai;
```

Gambar 2.9 *Script* Perhitungan Waktu Eksekusi Transaksi Data.



```php
<?php if($this->session->flashdata('time_akhir') == FALSE) : ?>
    <div class="alert alert-info mt-3 font-weight-bold">
      <?= 'Waktu Untuk Mengambil '.$jumlah_user.' Data User (READ) '.$waktu_eksekusi ?> Detik
    </div>
<?php else :?>
    <?php $waktu_eksekusi = $this->session->flashdata('time_akhir') - $this->session->flashdata('time_awal') ?>
    <div class="alert alert-info mt-3 font-weight-bold">
        <?= $this->session->flashdata('msg')." ".number_format($waktu_eksekusi, 3) ?> Detik
    </div>
<?php endif; ?>
```

Gambar 2.10 *Script* Untuk Menampilkan Waktu Eksekusi (*Alert*).

```php
#CREATE
public function tambah()
   {

      $nama = $this->input->post('nama', true);
      $email = $this->input->post('email', true);
      $username = $this->input->post('username', true);
      $password = password_hash($this->input->post('password', true), PASSWORD_BCRYPT);
      $akses_level = $this->input->post('akses_level', true);

      $curl = curl_init();

      curl_setopt_array($curl, array(
      CURLOPT_URL => "https://sasparts-ci.my.id/user",
      CURLOPT_SSL_VERIFYHOST => 0,
      CURLOPT_SSL_VERIFYPEER => 0,
      CURLOPT_RETURNTRANSFER => true,
      CURLOPT_ENCODING => "",
      CURLOPT_MAXREDIRS => 10,
      CURLOPT_TIMEOUT => 30,
      CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
      CURLOPT_CUSTOMREQUEST => "POST",
      CURLOPT_POSTFIELDS => "nama=".$nama."&email=".$email."&username=".$username.
                            "&password=".$password."&akses_level=".$akses_level,
      CURLOPT_HTTPHEADER => array(
         "content-type: application/x-www-form-urlencoded",
         "key: 8cb2237d0679ca88db6464eac60da96345513964"
      ),
      ));

      $response = curl_exec($curl);
      $err = curl_error($curl);

      curl_close($curl);

      if ($err) {
      echo "cURL Error #:" . $err;
      } else {
         return $response = json_decode($response, true);
      }


   }
```

Gambar 2.11 *Script* Transaksi Data Dengan *Method POST* (*Create*).

```php
#READ
public function listing()
   {
      $curl = curl_init();

      curl_setopt_array($curl, array(
      CURLOPT_URL => "https://sasparts-ci.my.id/user",
      CURLOPT_SSL_VERIFYHOST => 0,
      CURLOPT_SSL_VERIFYPEER => 0,
      CURLOPT_RETURNTRANSFER => true,
      CURLOPT_ENCODING => "",
      CURLOPT_MAXREDIRS => 10,
      CURLOPT_TIMEOUT => 30,
      CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
      CURLOPT_CUSTOMREQUEST => "GET",
      CURLOPT_HTTPHEADER => array(
         "key: 8cb2237d0679ca88db6464eac60da96345513964"
      ),
      ));

      $response = curl_exec($curl);
      $err = curl_error($curl);

      curl_close($curl);

      if ($err) {
      echo "cURL Error #:" . $err;
      }
      else {
         $response = json_decode($response, true);
         return $response['data'];
      }
   }
```

Gambar 2.12 *Script* Transaksi Data Dengan *Method GET* (*Read*).

```php
#UPDATE
public function ubah($id)
   {
      $nama = $this->input->post('nama', true);
      $email = $this->input->post('email', true);
      $akses_level = $this->input->post('akses_level', true);

      $curl = curl_init();

      curl_setopt_array($curl, array(
      CURLOPT_URL => "https://sasparts-ci.my.id/user",
      CURLOPT_SSL_VERIFYHOST => 0,
      CURLOPT_SSL_VERIFYPEER => 0,
      CURLOPT_RETURNTRANSFER => true,
      CURLOPT_ENCODING => "",
      CURLOPT_MAXREDIRS => 10,
      CURLOPT_TIMEOUT => 30,
      CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
      CURLOPT_CUSTOMREQUEST => "PUT",
      CURLOPT_POSTFIELDS => "id_user=".$id."&nama=".$nama."&email=".$email.
                            "&akses_level=".$akses_level,
      CURLOPT_HTTPHEADER => array(
         "content-type: application/x-www-form-urlencoded",
         "key: 8cb2237d0679ca88db6464eac60da96345513964"
      ),
      ));

      $response = curl_exec($curl);
      $err = curl_error($curl);

      curl_close($curl);

      if ($err) {
      echo "cURL Error #:" . $err;
      } else {
         return $response = json_decode($response, true);
      }
   }
```

Gambar 2.13 *Script* Transaksi Data Dengan *Method PUT* (*Update*).

```php
#DELETE
public function hapus($id)
   {
      $curl = curl_init();

      curl_setopt_array($curl, array(
      CURLOPT_URL => "https://sasparts-ci.my.id/user",
      CURLOPT_SSL_VERIFYHOST => 0,
      CURLOPT_SSL_VERIFYPEER => 0,
      CURLOPT_RETURNTRANSFER => true,
      CURLOPT_ENCODING => "",
      CURLOPT_MAXREDIRS => 10,
      CURLOPT_TIMEOUT => 30,
      CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
      CURLOPT_CUSTOMREQUEST => "DELETE",
      CURLOPT_POSTFIELDS => "id_user=".$id,
      CURLOPT_HTTPHEADER => array(
         "content-type: application/x-www-form-urlencoded",
         "key: 8cb2237d0679ca88db6464eac60da96345513964"
      ),
      ));

      $response = curl_exec($curl);
      $err = curl_error($curl);

      curl_close($curl);

      if ($err) {
      echo "cURL Error #:" . $err;
      } else {
         return $response = json_decode($response, true);
      }
   }
```

Gambar 2.14 Script Transaksi Data Dengan Method DELETE (Delete).