BAB III
METODE PENELITIAN

3.1 Metode Penelitian

Tahapan penelitian yang di ajukan dalam proses penelitian mencangkup analisa kebutuhan *hardware* dan *software* yang digunakan, hingga pengujian keseluruhan sistem. Penelitian diawali dengan melakukan analisa terhadap kebutuhan akan *hardware*, *software*, dan komponen pendukung yang dibutuhkan. Kemudian studi literatur, yaitu melakukan pengumpulan data refrensi untuk dipelajari dari jurnal penelitian, baik jurnal penelitian nasional maupun internasional, serta buku artikel dan juga *Web*site untuk digunakan menjadi refrensi dalam penelitian ini. Lalu identifikasi masalah, mengidentifikasi masalah yang sebelumnya belum terpenuhi didalam suatu perancangan penelitian.

Kemudian perancangan *Database* RESTful API, JWT (JSON *Web Token*) dan *Indexing*, merancang *database* menggunakan MySQL kemudian merancang sekuriti menggunakan JWT sebagai autentikasi dan otorisasi dan penambahan metode *Indexing*. Perancangan metode *Round robin*, merancang sistem *Load balancing* menggunakan metode *Round robin* dengan Nginx. Kemudian pengujian sistem, menguji sistem secara keseluruhan agar sistem dapat berjalan dengan yang diharapkan, dan jika pengujian sistem ini tidak sesuai yang di rencanakan, maka dilakukan perbaikan pada perancangan sistem. Terakhir penulisan laporan akhir, menulis hasil penelitian dan kesimpulan dalam bentuk laporan akhir.

3.2 Identifikasi Masalah

Salah satu teknologi yang belum termanfaatkan adalah penggunaan sistem database yang dapat memastikan integritas data yang dikirim dan dapat digunakan untuk otentikasi/otorisasi dua aplikasi yang berbeda. Menggunakan otentikasi dapat membantu meningkatkan sekuritas dari data tersebut. Kemudian mengoptimasi *server virtual* yang sebelumnya menggunakan *single server* lalu di optimalkan menggunakan *multi-server*. Menggunakan *multi-server* berguna membantu *server-server* tersebut mentransfer data secara efisien, dan mengoptimalkan penggunaan aplikasi pengiriman sumber daya sehingga terhindar dari *server* yang *overload*.

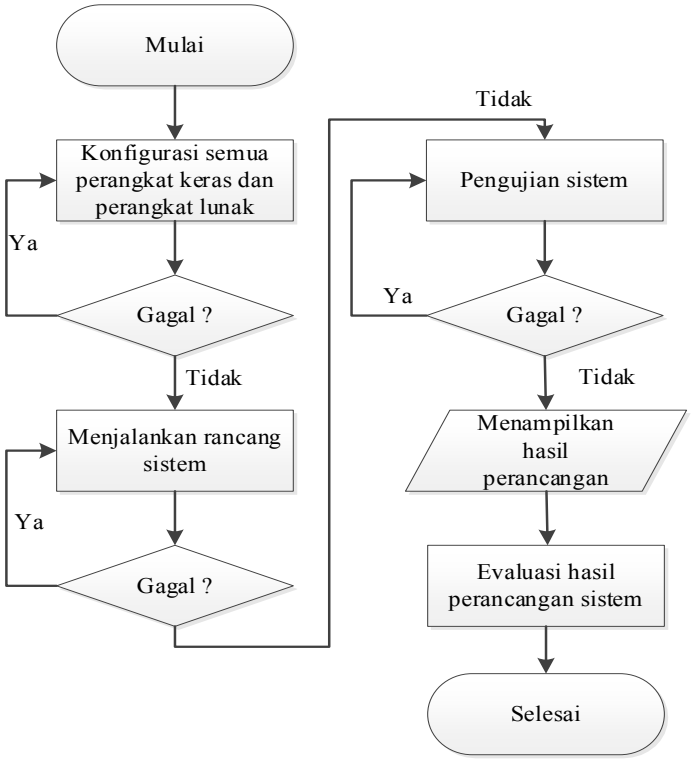
Algoritma yang dipakai dalam penelitian ini menggunakan Algoritma *Round robin*. Metode ini dapat merotasi *server* dengan mengarahkan *traffic* ke *server* yang pertama tersedia, lalu memindahkan *server* tersebut ke bawah dari antrean. Paling berguna saat *server-server* dengan spesifikasi serupa dan tidak banyak terdapat koneksi yang persisten.

3.3 Perancangan Sistem

Perancangan sistem adalah proses perancangan untuk merancang sistem atau memperbaiki sistem yang telah ada sehingga sistem menjadi lebih baik serta dapat mengerjakan pekerjaan secara efektif dan efisien, proses rancangan bisa berupa rancangan masukan, rancangan keluaran, dan rancangan *file*.

3.3.1 Alur Perancangan Sistem

Di bagian sub-bab ini menjelaskan tentang alur dari perancangan sistem yang terdiri dari konfigurasi perangkat lunak, perancangan sistem, pengujian sistem, dan hasil perancangan. Alur perancangan sistem tersebut digambarkan melalui *flowchart* yang dapat dilihat pada Gambar 3.1.

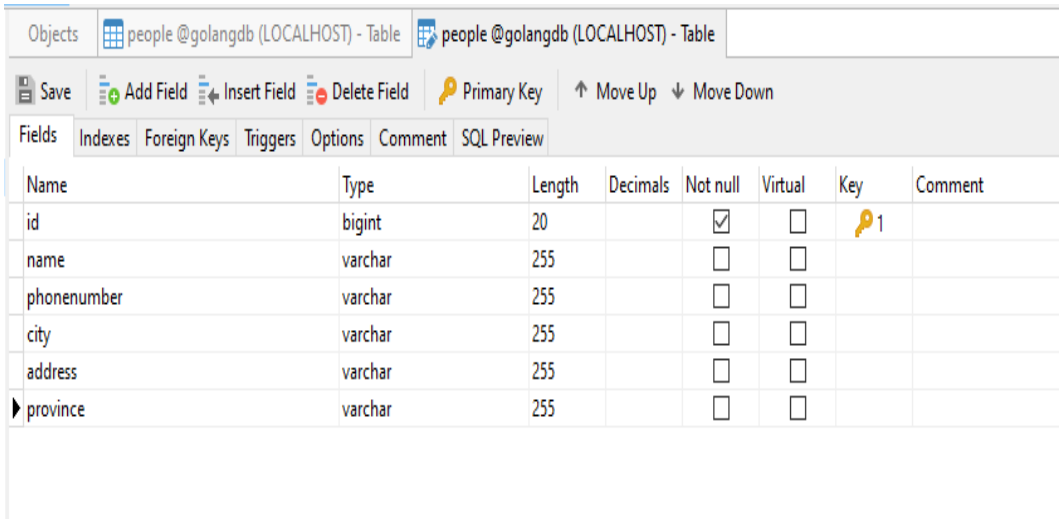


Gambar 3.1 Diagram Alur Perancangan Sistem

Memulai dari konfigurasi perangkat lunak yang digunakan yang disebutkan sebelumnya di sub-bab identifikasi masalah, lalu menjalankan rancangan sistem suatu infrastruktur, kemudian melakukan pengujian sistem untuk menguji sistem yang telah dirancang sebelumnya, dan yang terakhir menampilkan hasil perancangan sistem sekaligus melakukan evaluasi dari hasil perancangan tersebut.

3.3.2 Sistem Basis Data

Dalam perancangan sistem, dibutuhkan suatu tempat untuk menyimpan data-data yang dibutuhkan sistem, data-data tersebut dapat berupa *file* atau tabel. Perancangan basis data yang digunakan dalam penelitian ini dengan menggunakan *Database* MySQL terdiri dari satu tabel yang dilabeli *person,* memiliki 6 *field* di antaranya adalah *id*, *name*, *phonenumber*, *city*, *address*, dan *province*. Adapun struktur dari tabel *people* seperti yang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Struktur Tabel Basis Data *people*

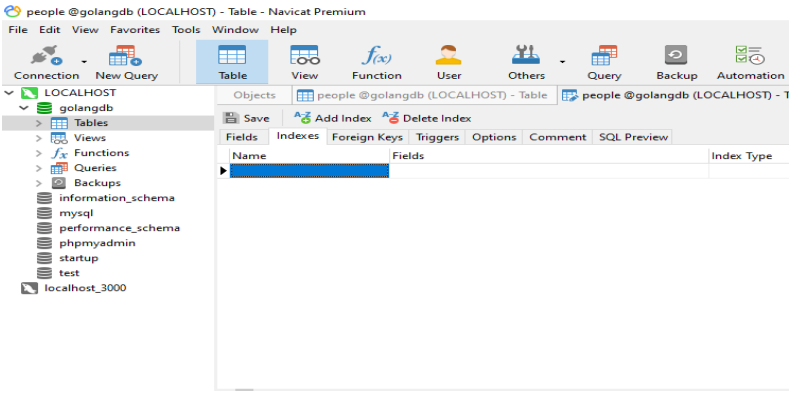
Pada Gambar 3.2 menunjukkan *Fields* dengan *row id* sebagai *primary keys* mempunyai format tipe *bigint* dengan *total length* 20. Kemudian untuk *name, phonenumber, city, address,* dan *province* mempunyai total *length* yang sama yaitu 255 dan mempunyai format tipe *varchar.*

Basis Data yang dirancang mempunyai *dummy* data yang diimpor dari *Microsoft Excel* berformat .csv berjumlah 59.283 yang bertujuan menjadi tolak

ukur perhitungan seberapa cepat *Response time, min. Response time, max. Response time,* dan *error.*

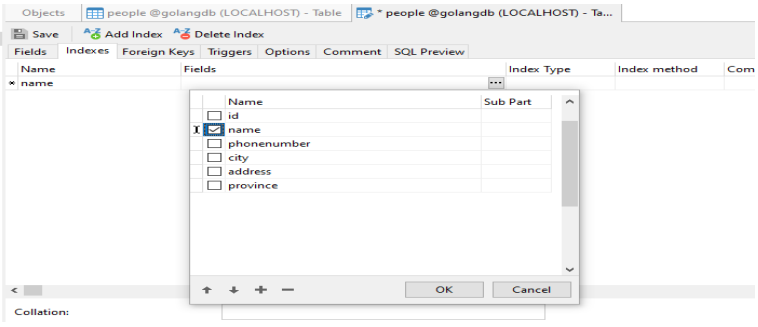
3.3.3 Sistem Penambahan *Indexes*

Index adalah objek pada MySQL yang berisi data yang terurut dari nilai-nilai pada satu atau lebih *field* dalam suatu *table*. Sama seperti daftar isi pada sebuah buku, *index* terutama digunakan untuk mempercepat pencarian terhadap suatu set data dengan kondisi tertentu yang melibatkan kombinasi *field* yang sudah didefinisikan dalam suatu *index*. Tanpa *index*, pencarian data biasanya akan memakan waktu lama, terutama jika data sudah dalam skala jumlah yang sangat besar. *Index* pada tabel dapat dikonfigurasi pada menu *Indexes* di *Software Navicat* yang dapat dilihat pada Gambar 3.3.



Gambar 3.3 Tab *Indexes* di *Navicat*

Pada Gambar 3.3 Tab *Indexes* digunakan untuk fitur *index* di *database* melalui *software Navicat* dapat mengkonfigurasikannya pada tab *Indexes*, dengan memilih *field* yang akan diimplementasikan suatu *index* dan mengubahnya hanya dengan mencentang *field* yang akan dipilih sebagai *index*, seperti tampak pada Gambar 3.4.

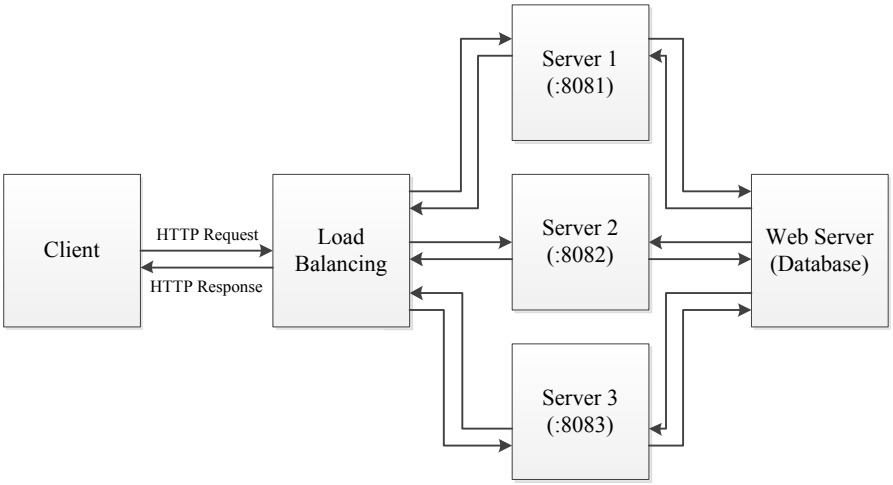


Gambar 3.4 Kotak edit *name* untuk mengatur indeks

Pada Gambar 3.4 Kotak edit *name* digunakan untuk mengatur *index* di *database* melalui *software Navicat* dapat mengkonfigurasikannya pada tab *Indexes*, dengan memilih *field* yang akan diimplementasikan suatu *index* dan mengubahnya hanya dengan mencentang *field* yang akan dipilih sebagai *index*.

3.3.4 Sistem *Load balancing*

Pada penelitian ini menggunakan perancangan sistem jaringan dengan menerapkan algoritma *Load balancing* sebagai mekanisme dalam pemilihan *server*. Perancangan sistem dapat dilihat dengan blok diagram sistem *Load balancing* pada Gambar 3.5.



Gambar 3.5 Blok diagram sistem *Load balancing*

Pada gambar 3.5 merupakan skenario percobaan dengan menggunakan 3 buah *virtual server* yang dihubungkan dengan *Load balancing*. Masing-masing *virtual server* di mulai melalui *Command Prompt*. Proses *Load balancing* secara umum, yaitu pembagian beban pada *server* dalam membagi *traffic* secara otomatis. Pada awal proses, *client* melakukan *request* kepada *server*. *Request* dari *client* akan masuk ke dalam *controller Load balancing* dan dilakukan pengecekan *server*. Kemudian *controller* akan memilih *server* sesuai algoritma *Load balancing*. *Request client* akan diteruskan ke dalam antrian *server*. *Server* akan merespon dan mengirimkan data kepada *client*. Sistem *Load balancing* akan terus berjalan hingga *request* dari *client* telah berhenti.

3.4 Konfigurasi Sistem

Sebelum menjalankan program ada beberapa hal yang diperhatikan, yaitu kebutuhan sistem dan konfigurasi dari sistem. Tujuan pokok dari sistem komputer adalah mengolah data untuk menghasilkan informasi. Dalam melaksanakan tujuan pokok tersebut diperlukan adanya elemen-elemen yang mendukung. Elemen-elemen dari sistem tersebut antara lain kebutuhan perangkat keras (*Hardware*) dan kebutuhan perangkat lunak (*Software*).

3.4.1 Konfigurasi Algoritma *Round Robin*

Sebelumnya telah dibahas mengenai perancangan sistem. Dimana dalam perancangan sistem tersebut terdapat langkah dalam konfigurasi perangkat. Konfigurasi perangkat merupakan langkah awal dalam melakukan analisa terhadap algoritma *Load balancing*. Pengaturan algoritma *Load balancing* dilakukan pada *server controller Load balancing*. Konfigurasi dilakukan pada *file default* pada direktori `etc/nginx/sites-available`. *Source code* konfigurasi dari *Load balancing Round robin* pada aplikasi *web server nginx* dapat dilihat pada Gambar 3.6.

localhost.conf

C: > Program Files > nginx-1.23.0 > conf > sites-available > localhost.conf

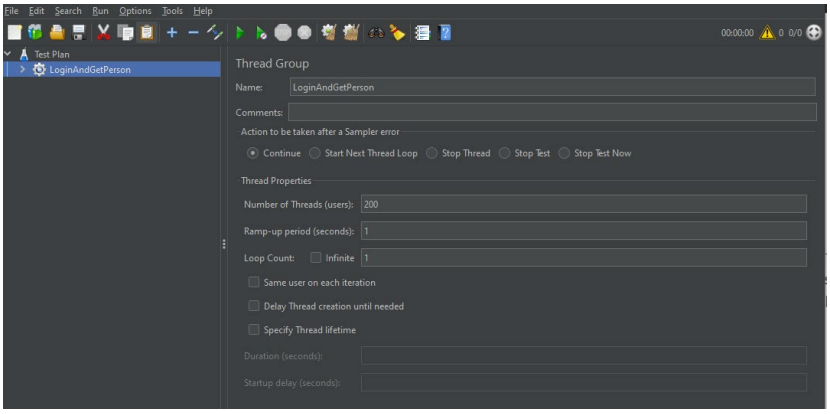
```
upstream localhost {
  server localhost:8081;
  server localhost:8082;
  server localhost:8083;
}

server {
  listen          8080;
  server_name     localhost;
  access_log      "C:/Program Files/nginx-1.23.0/logs/localhost.access.log" combined;

  location / {
    proxy_pass      http://localhost;
  }
}
```

Gambar 3.6 *Source Code Algoritma Round robin*

Round robin merupakan metode *default* apabila parameter algoritma tidak ditentukan. Maka otomatis metode yang digunakan menggunakan algoritma *Round robin* dan beban akan dibagi secara rata ke seluruh *server*.

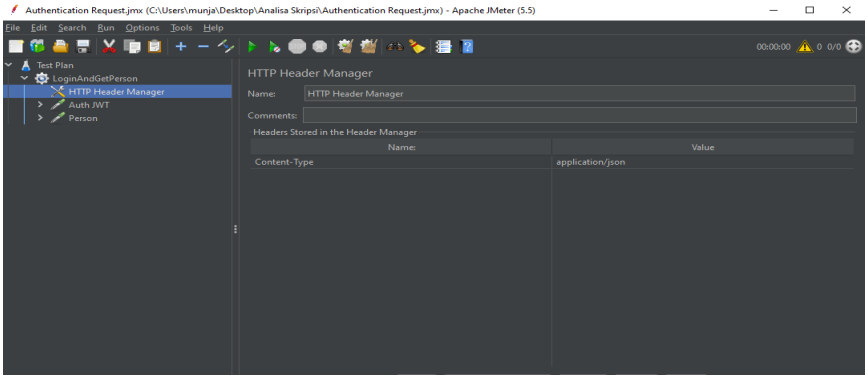
3.4.2 Konfigurasi Apache JMeter

Apache JMeter untuk pengujian kinerja membantu untuk menguji sumber daya statis dan dinamis, membantu menemukan pengguna bersamaan di situs *Web* dan berbagai analisis grafis untuk pengujian kinerja. Pengujian kinerja JMeter meliputi uji beban dan uji tegangan aplikasi *Web*. Ini adalah urutan konfigurasi dari perangkat lunak Apache JMeter pada Gambar 3.7.



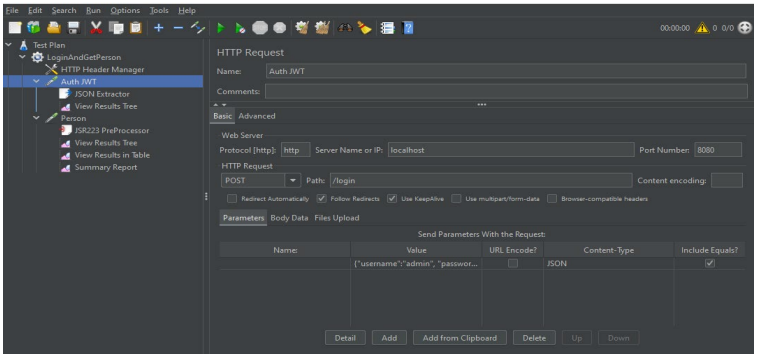
Gambar 3.7 Menambahkan *Thread Group*

Pada Gambar 3.7 menjelaskan tentang *Thread* Group yang digunakan untuk mengelompokan *service* yang akan di tes. Contoh mempunyai *service* Mahasiswa dan Dosen, kedua *service* ini sebaiknya dibuatkan *Thread Group* masing-masing pada Gambar 3.8.



Gambar 3.8 HTTP *Header Manager*

Pada Gambar 3.8 HTTP *Header Manager* digunakan untuk menambahkan *name* dan *value*, dengan kolom name diisi dengan *Content-Type* dan *Value application-json* yang di jelaskan pada Gambar 3.9.



Gambar 3.9 Konfigurasi HTTP *Request*

Pada Gambar 3.9 HTTP *Request* berfungsi untuk menambahkan node HTTP *Request*. Di node inilah peneliti akan menentukan *Web service* yang akan di tes. Misal *Web service* mempunyai layanan *Auth* JWT yaitu untuk menkonfigurasi otentikasi ke *database* menggunakan JSON *Extractor* dan *Person* untuk menerima otentikasi dari *Auth* JWT menggunakan *JSR223 PreProcessor*. Layanan ini akan ditambahkan sebagi bagian dari HTTP *Request*.

3.5 Perangkat Penelitian

Instrumen yang di gunakan pada penelitian kali ini meliputi perangkat keras (*Hardware*) dan perangkat lunak (*Software*). Berikut spesifikasi lebih rinci terkait *Hardware* dan *Software* yang digunakan penelitian dalam aplikasi sebagai berikut:

1. Perangkat keras

Satu unit laptop dengan *processor* intel core i3-7020. RAM dengan tipe DDR4 berkapasitas 4GB, penyimpanan *internal* MidasForce SSD berkapasitas 256GB, dan *Graphics card* menggunakan Intel UHD.

2. Perangkat lunak

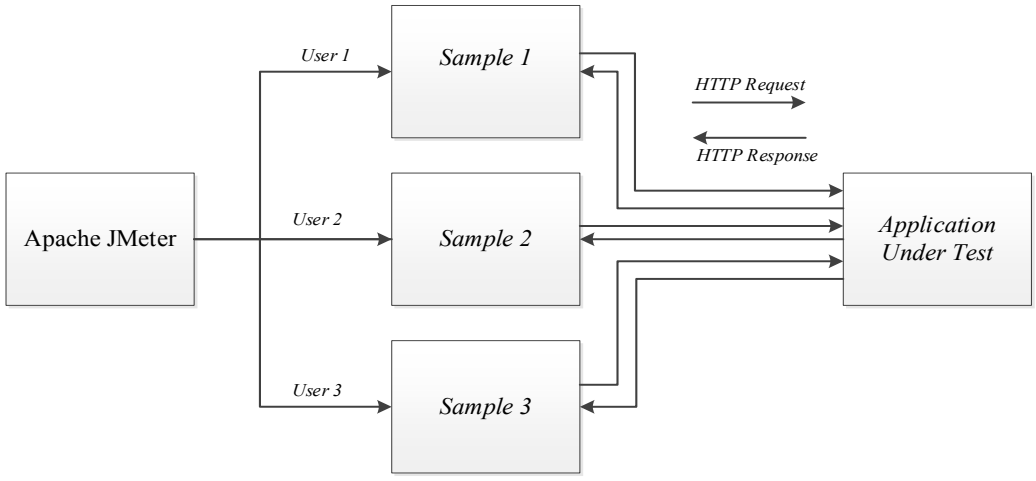
Software yang di gunakan adalah sebagai berikut:

a. Sistem Operasi yang di gunakan adalah *Windows 10 Home Single Language* 64bit (*build* 19043)
b. Perangkat Lunak yang digunakan sebagai *testing* API adalah *POSTMAN* (v9.25.2)
c. Bahasa Pemrograman yang digunakan adalah GOLANG (GO *Language*) (v1.18.4)

d. *Web server* yang digunkan adalah *Apache* (v2.5.53)
e. *Database Server* yang digunakan adalah MySQL dengan jenis *server* MariaDB(v10.4.24) dan perangkat lunak tambahan menggunakan *Navicat* (v15.0.22)
f. Perangkat lunak yang digunakan sebagai *code editor* adalah *Microsoft Visual Studio Code* (v1.69.2)
g. Perangkat lunak yang digunakan sebagai *load balancer* adalah *Nginx* (v.1.23.0)
h. Perangkat lunak yang digunakan sebagai *test performance* menggunakan *Apache JMeter* (v.5.5)

3.6 Desain Pengujian

Uji coba kali ini menggunakan perangkat lunak Apache JMeter, Aplikasi Apache JMeter adalah perangkat lunak *open source*, 100% aplikasi Java murni dirancang untuk memuat tes perilaku fungsional dan mengukur kinerja. Pada pengujian ini akan menggunakan dua *Confing Element* yaitu, *HTTP Cookie Manager* dan *HTTP Request Default*. *HTTP Request Default* merupakan salah satu fitur yang ada pada *Config Element* agar pengujian lebih optimal sehingga *Server Name* atau *IP* and *Port Number* tidak di jalankan berulang-ulang. *HTTP Cookie Manager* merupakan salah satu fitur yang ada pada *Config Element* agar dapat menyimpan *cookie* yang dapat digunakan oleh setiap *request* selanjutnya. Alur kinerja dari Apache JMeter pada Gambar 3.10.



Gambar 3.10 Alur Kinerja *Apache JMeter*

Pada Gambar 3.10 terlihat bahwa tujuan dari pengujian yaitu untuk mengetahui perbandingan performa dari algoritma *Round robin* pada *Web server*. Pengujian juga dilakukan untuk mengetahui nilai rata-rata pada setiap parameter yang diberikan. Pemberian *request* digunakan untuk memberi beban kepada *server* dalam satu waktu. Hasil dari setiap percobaan akan dibandingkan dan dilakukan analisa. Sehingga akan didapat perbandingan perfoma antara algoritma *Round robin* dan tidak memakai algoritma *Round robin*, serta memakai *Index*es atau tidak memakai *Index*es yang dapat disimpukan pada akhir penelitian. Berikut tahapan uji coba menggunakan perangkat lunak Apache JMeter pada Tabel 3.1.

Tabel 3.1 Tahapan pengujian penelitian

Urutan percobaan	**Deskripsi Percobaan**
1	Mengaktifkan *server* tanpa *Load balancing*, *client* melakukan *Request* GET *Person* all.
2	Mengaktifkan 3 *server* dengan *Load balancing*, *client* melakukan *Request* GET *Person* all.
3	Mengaktifkan *server* tanpa *Load balancing* menggunakan metode *Indexes*, *client* melakukan *Request* GET *Person* all
4	Mengaktifkan *server* dengan *Load balancing* menggunakan metode *Indexes*, *client* melakukan *Request* GET *Person* all.
5	Mengaktifkan *server* tanpa *Load balancing* dan tidak menggunakan metode *Indexes*, *client* melakukan *Request* GET *Person* all.
6	Mengaktifkan *server* dengan *Load balancing* tetapi tidak menggunakan metode *Indexes*, *client* melakukan *Request* GET *Person* all.

Pengujian pada parameter *Response time* di atas yang dilakukan sebanyak 6 kali, mempunyai 5 kategori sebagai berikut.

a. 40 req/30s (*rate*) dengan banyak koneksi 40.
b. 60 req/30s (*rate*) dengan banyak koneksi 60.
c. 80 req/30s (*rate*) dengan banyak koneksi 80.
d. 90 req/30s (*rate*) dengan banyak koneksi 90.

e. 100 req/30s (*rate*) dengan banyak koneksi 100.

Hasil dari pengujian dengan mengirimkan *request* akan didapat nilai dari *Response time* yang kemudian diambil nilai rata-ratanya dari hasil 10 kali percobaan. Perekaman hasil pengujian pada Apache JMeter dilakukan secara otomatis menjadi tabel hasil pengukuran disetiap percobannya, dan disimpan hasilnya secara manual dengan format *.csv*. Hal tersebut dilakukan karena pada Apache JMeter tidak ada fitur untuk menyimpan hasil pengukuran secara otomatis.

Pengujian yang dilakukan menggunakan parameter *Samples* (*User*/s), *Response time* Average (ms), *Min Respons*e *time* (ms), *Max Response time* (ms), dan *Error*. Berikut penjelasan dari masing-masing parameter:

a. Samples

Samples menangkap jumlah total sampel yang dikirim ke *server*. Contoh menempatkan Pengontrol *Loop* untuk menjalankannya 5 kali permintaan khusus ini dan kemudian 2 iterasi (Disebut Hitungan *Loop* di Grup Utas) diatur dan uji beban dijalankan untuk 100 pengguna.

b. *Response time Average*

Ini adalah waktu *Response* rata-rata untuk permintaan *HTTP* tertentu. Waktu *Respons*e ini dalam milidetik, dan rata-rata untuk 5 *loop* dalam dua iterasi untuk 100 pengguna.

c. Min Response time

Waktu minimum yang dihabiskan oleh permintaan sampel yang dikirim untuk label ini. Total sama dengan waktu minimum di semua sampel.

d. Max Response time

Pengeluaran ikatan maksimum berdasarkan permintaan sampel yang dikirim untuk label ini. Totalnya sama dengan waktu maksimum di semua sampel.

e. Error

Persentase total kesalahan yang ditemukan untuk permintaan sampel tertentu. 0,0% menunjukkan bahwa semua permintaan berhasil diselesaikan. Total sama dengan persentase kesalahan sampel di semua sampel (Total Sampel).