

**PERBANDINGAN PERFORMA MIDDLEWARE
JAVA DAN PHP PADA ARSITEKTUR
TIGA LAYER BERBASIS REST**



Disusun Oleh:

N a m a : I Putu Arya Sabha Pramana

NIM : 13523218

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2018

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**PERBANDINGAN PERFORMA MIDDLEWARE
JAVA DAN PHP PADA ARSITEKTUR
TIGA LAYER BERBASIS REST**



الجمعة الإسلامية الأندلسية

Yogyakarta, 30 April 2018

Pembimbing,

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc)

HALAMAN PENGESAHAN DOSEN PENGUJI

**PERBANDINGAN PERFORMA MIDDLEWARE
JAVA DAN PHP PADA ARSITEKTUR
TIGA LAYER BERBASIS REST
TUGAS AKHIR**

Telah dipertahankan di depan sidang pengujian sebagai salah satu syarat untuk
memperoleh gelar Sarjana Teknik Informatika
di Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 04 Juni 2018

Tim Penguji

Dr. Raden Teduh Dirgahayu, S.T., M.Sc.

Anggota 1

Andhika Giri Persada, S.Kom., M.Eng.

Anggota 2

Sri Mulyati, S.Kom., M.Kom.

Mengetahui,

Ketua Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia



(Andhika, S.T., M.Eng.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : I Putu Arya Sabha Pramana

NIM : 13523218

Tugas akhir dengan judul:

PERBANDINGAN PERFORMA MIDDLEWARE JAVA DAN PHP PADA ARSITEKTUR TIGA LAYER BERBASIS REST

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 30 April 2018



(I Putu Arya Sabha Pramana)

HALAMAN PERSEMBAHAN***Orang tua***

Yang sudah memberikan dukungan serta kekuatan untuk motivasi saya sendiri

Komputer saya

Yang sudah bersedia tidak pernah error saat pengerjaan laporan terakhir

HALAMAN MOTO

“Learning never exhaust the mind”

KATA PENGANTAR

Alhamdulillah, penulis panjatkan kehadiran Allah SWT yang telah memberikan rahmat, hidayah, dan karunia-Nya, sehingga laporan Tugas Akhir dapat terselesaikan. Shalawat dan salam kami haturkan kepada junjungan kita Nabi Muhammad SAW, yang telah membawa kita dari zaman jahiliyah menuju jaman terang benderang.

Tugas akhir ini dibuat sebagai salah satu syarat yang harus dipenuhi untuk memperoleh gelar sarjana di Jurusan Teknik Informatika Universitas Islam Indonesia. Adapun Tugas Akhir saya Perbandingan Performa Middleware JAVA dan PHP Pada Arsitektur Tiga Layer Berbasis Rest

Pelaksanaan Tugas Akhir ini merupakan salah satu mata kuliah wajib dari jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia. Tugas Akhir juga merupakan sarana dalam menerapkan keilmuan yang sudah diambil selama berkuliah ini.

Oleh karena itu, penulis ingin menyampaikan rasa terima kasih dan dukungan yang telah diberikan kepada :

1. Orang tua dan keluarga penulis atas segala dukungan dan doa selama penulis mengerjakan Tugas Akhir.
2. Bapak Hendrik, S.T., M.Eng., SAP IHL, OCA., selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Dosen Pembimbing Tugas Akhir Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Segenap keluarga besar teman-teman di Fakultas Teknologi Industri khususnya Jurusan Teknik Informatika Universitas Islam Indonesia yang sudah membantu memberikan saran dan dukungan.
5. Forum serta komunitas dunia maya khususnya Stack Overflow dan Quora yang sudah memberikan banyak sekali referensi dalam pengerjaan dan implementasi.
6. Semua pihak yang telah membantu dan mendukung pelaksanaan Tugas Akhir yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, karena keterbatasan pengalaman yang dimiliki. Oleh karena itu, penulis mengharapkan kritik dan saran yang

membangun untuk kesempurnaan Laporan Tugas Akhir ini. Akhir kata, penulis berharap agar laporan ini dapat menjadi referensi dan rujukan untuk semua pihak.

Wassalamu'alaikum Warahmatulahi Wabarakatuh

Yogyakarta, 30 April 2018

(I Putu Arya Sabha Pramana)

SARI

Performa pada sebuah aplikasi berukuran enterprise merupakan faktor penting. Aplikasi berskala enterprise memiliki tingkat transaksi data yang sangat tinggi, contohnya adalah aplikasi *e-commerce*. Dalam trennya saat ini, pengguna belanja *online* semakin lama semakin mempercayai membeli barang melalui retail *online*. Retail *online* tersebut merupakan solusi kemudahan belanja yang nyaman dengan memanfaatkan teknologi saat ini. Untuk itu, aplikasi tersebut disarankan untuk memiliki performa yang tinggi. Pentingnya sebuah performa pada era transaksi online perlu didukung dengan adanya kualitas pengembangan. Terdapat berbagai macam pilihan untuk mengembangkan perangkat lunak berbasis *middleware*. Sebuah *middleware* merupakan solusi untuk menyelesaikan permasalahan terhadap kebutuhan transaksi yang semakin meningkat. Kebutuhan transaksi yang meningkat harus dibarengi dengan kenyamanan dan kecepatan transaksi. Dari sekian banyak itu, tentu saja perlu di kaji dari sisi performa untuk menentukan mana yang lebih cocok dari sisi performa untuk di kembangkan sesuai kebutuhan. Kebutuhan yang tepat akan menghasilkan produk yang efisien dan memiliki performa tinggi.

Pengujian performa *middleware* pada JAVA dan PHP ini menggunakan salah satu *microframework* yang dirancang khusus untuk pengembangan layanan berbasis web tersebut. Pada JAVA menggunakan *Spring Boot* dan PHP menggunakan *Slim*. Untuk melakukan pengujian, diperlukan sebuah *server* yang secara khusus dibangun untuk memasang layanan tersebut. Penggunaan *server* itu dapat memberikan independensi dalam pengujian sehingga tidak dipengaruhi oleh faktor atau proses lainnya yang tidak berhubungan dengan pengujian. Hal itu dapat terjadi apabila pengujian dikerjakan bersamaan di dalam komputer yang sama. digunakan sebuah aplikasi bernama Apache JMeter. Aplikasi tersebut akan memberikan sebuah informasi terkait dengan performa yang diujikan. Dengan menggunakan fitur yang ada, pengujian dalam dilakukan secara sistematis menggunakan *test plan* yang dirancang untuk kebutuhan pengujian performa.

Kata kunci: performa, middleware, transaksi, microframework, pengujian,.

GLOSARIUM

Client	sebuah komputer yang mengirimkan dan menerima permintaan
Record	kumpulan data.
Database	sebuah serangkaian data yang diolah oleh komputer.
Listener	mekanisme pada pengujian untuk merepresentasikan respon.
Middleware	sebuah teknologi yang memberikan jembatan komunikasi.
Request	permintaan yang berasal dari klien
Server	komputer yang berfungsi menerima permintaan.
Thread	serangkaian perintah dalam komputer.
Platform	lingkungan pendukung dalam perangkat lunak untuk berjalan.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING.....	ii
HALAMAN PENGESAHAN DOSEN PENGUJI.....	Error! Bookmark not defined.
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR	Error! Bookmark not defined.
HALAMAN PERSEMBAHAN	iv
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI.....	ix
GLOSARIUM.....	x
DAFTAR ISI.....	xi
DAFTAR TABEL.....	xiv
DAFTAR GAMBAR	xv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	4
1.5 Usulan Solusi	4
1.6 Langkah Penyelesaian.....	4
1.7 Sistematika Penulisan	4
BAB II LANDASAN TEORI.....	6
2.1 Konsep Dasar Arsitektur N-Tier.....	6
2.1.1 Definisi Arsitektur N-Tier.....	6
2.1.2 Komponen Arsitektur N-Tier.....	6
2.2 Konsep Dasar API.....	7
2.2.1 Definisi API	7
2.2.2 Fungsi API	7
2.2.3 Contoh API	8
2.3 Konsep Middleware	8
2.3.1 Definisi Middleware	8
2.3.2 Jenis Middleware	9
2.3.3 Spring Boot	10

2.3.4	Slim.....	10
2.4	Konsep Dasar Performance Test	11
2.4.1	Definisi Performance Test	11
2.4.2	Manfaat Performance Test.....	11
2.4.3	Jenis Performance Test	11
BAB III	METODOLOGI PENELITIAN	13
3.1	Peninjauan Lingkungan Pengujian.....	13
3.1.1	Definisi Ruang Lingkup Pengujian Sistem	13
3.1.2	Penentuan Lingkup Pengujian	13
3.1.3	Kekangan	14
3.2	Pengembangan Sistem Untuk Pengujian.....	15
3.2.1	Klasifikasi Masalah.....	15
3.2.2	Batasan Sistem	15
3.3	Analisis Pengujian Sistem	15
3.3.1	Indikator Pengujian	15
3.3.2	Arsitektur Pengujian Sistem.....	16
3.3.3	Use Case Diagram.....	17
3.4	Pengujian Kinerja	18
3.4.1	Pemilihan Perangkat Uji.....	18
3.4.2	Test Plan.....	18
BAB IV	PEMBAHASAN.....	21
4.1	Implementasi Server.....	21
4.1.1	Konfigurasi Server	21
4.1.2	Uji Coba Server.....	22
4.2	Implementasi Client.....	24
4.2.1	Konfigurasi Client.....	24
4.2.2	Konfigurasi JMeter	25
4.3	Pengujian Indikator	27
4.3.1	Pemilihan Beban Thread.....	27
4.4	Pengujian Performa PHP	29
4.4.1	Test Plan PHP	30
4.4.2	Service PHP	32
4.5	Pengujian Performa JAVA	34
4.5.1	Test Plan JAVA	34

4.5.2 Service JAVA	37
4.6 Perbandingan	39
4.6.1 Tabel Perbandingan.....	39
4.6.2 Grafik Perbandingan	40
4.6.3 Kendala	42
BAB V KESIMPULAN DAN SARAN	44
5.1 Kesimpulan.....	44
5.2 Saran.....	44
DAFTAR PUSTAKA	45

DAFTAR TABEL

Tabel 4.1 Tabel ujicoba interval 50	27
Tabel 4.2 Tabel ujicoba interval 200	28
Tabel 4.3 Tabel ujicoba interval 500	29
Tabel 4.4 Tabel komparasi.....	39

DAFTAR GAMBAR

Gambar 3.1 Arsitektur pengujian.....	16
Gambar 3.2 Diagram Use Case.....	17
Gambar 3.3 Contoh <i>thread group</i>	19
Gambar 3.4 Contoh HTTP Request.....	19
Gambar 3.5 Contoh grafik Request.....	20
Gambar 3.6 Contoh tabel Request.....	20
Gambar 4.1 Konfigurasi Server.....	21
Gambar 4.2 Koneksi untuk client.....	22
Gambar 4.3 Pengecekan kelas IP.....	23
Gambar 4.4 Pengecekan kelas IP server.....	23
Gambar 4.5 Memanggil koneksi ke server.....	24
Gambar 4.6 Virtual Machine client.....	25
Gambar 4.7 Halaman awal JMeter.....	26
Gambar 4.8 Thread Group.....	26
Gambar 4.9 Susunan Thread Group PHP.....	30
Gambar 4.10 Request PHP.....	31
Gambar 4.11 Tabel data PHP.....	32
Gambar 4.12 HTTP Request PHP.....	33
Gambar 4.13 Konfigurasi URL dan tipe request.....	33
Gambar 4.14 Respon data JSON dari PHP.....	34
Gambar 4.15 Susunan Thread Group JAVA.....	35
Gambar 4.16 Header data JSON dari JAVA.....	35
Gambar 4.17 Request JAVA.....	36
Gambar 4.18 Tabel data JAVA.....	37
Gambar 4.19 Layanan JAVA.....	38
Gambar 4.20 Respon data JAVA.....	38
Gambar 4.21 Grafik komparasi (1).....	41
Gambar 4.22 Grafik komparasi (2).....	42

BAB I

PENDAHULUAN

1.1 Latar Belakang

Arsitektur perangkat lunak merupakan sebuah pemilihan yang penting dalam mengembangkan perangkat lunak. Saat ini, ada dua jenis pemilihan arsitektur aplikasi yang sering digunakan yaitu arsitektur client-server dan n-tier application. Aplikasi client-server merupakan bentuk paling sederhana dari sebuah perangkat lunak. Arsitektur tersebut memungkinkan sebuah aplikasi dapat berinteraksi langsung kepada database. Hal tersebut juga dapat memberikan akses langsung kepada bagian database untuk membuat perubahan. Kelebihan dari arsitektur ini adalah tiap client dapat mengatur metodenya sendiri untuk mengakses database sesuai kebutuhannya masing masing. Namun, seiring dengan kebutuhan akses informasi yang cepat dan luas, arsitektur tersebut dianggap sudah usang karena memiliki kekurangan yaitu pengembangannya yang memakan waktu dan interoperabilitas yang buruk. Untuk menjawab permasalahan tersebut, pengembangan arsitektur saat ini berpindah menjadi n-tier application yang memudahkan dalam pengembangan aplikasi secara masif dan memiliki tingkat kehandalan yang tinggi. Arsitektur tersebut memberikan solusi yaitu dibuatnya sebuah perantara atau disebut middleware antara client dan server. Perantara atau middleware tersebut kemudian memungkinkan sebuah aplikasi dapat dikembangkan secara masif karena tidak perlu membuat secara berulang untuk menjadikan aplikasi yang berbeda. Programmer hanya perlu mendesain ulang tampilannya sesuai kebutuhan dan berinteraksi dengan data menurut aturan yang sudah ditetapkan di bagian perantara. Teknologi tersebut kemudian disebut API atau Application Programming Interface.

API (Application Programming Interface) adalah sebuah protokol yang berfungsi sebagai perantara antara client dan server. Dalam dunia programming, API memiliki peranan penting dalam pengaksesan data supaya dapat diakses secara luas. Penggunaan teknologi API dalam pengembangan perangkat lunak, terutama yang berkaitan dengan transaksi jual-beli, menjadi sangat krusial. Contoh implementasi yang sering kita jumpai adalah pada sistem ticketing, third-party app, dan shopping assistant. Sistem tersebut mengambil data dari berbagai sumber untuk kemudian ditampilkan dalam suatu aplikasi. Aplikasi tersebut tentu memerlukan sebuah media untuk mengakses data dari sumbernya tersebut. Media yang

digunakan dalam “pengantaran” data tersebut adalah API. Saat ini, API menjadi bagian integral dari sebuah standar pengembangan aplikasi. Teknologi yang berkembang dalam pemilihan protokol API juga bermacam macam. Beberapa API yang pernah populer dan sampai saat ini sering digunakan adalah SOAP (Simple Object Access Protocol) dan REST (Representational State Transfer). SOAP menjadi solusi awal perkembangan API pada pengembangan perangkat lunak berbasis arsitektur n -tier. Namun, karena metodenya dianggap kurang efisien, sebab menggunakan XML yang notabene membutuhkan bandwidth tinggi, SOAP mulai ditinggalkan dan beralih kepada REST. REST memanfaatkan protokol yang sudah ada pada HTTP 1.1 untuk melakukan transaksi data.

Semenjak dikenalkan oleh Roy Fielding sekitar tahun 2000 pada disertasi PhD miliknya, teknologi pengembangan perangkat lunak berbasis REST (Representational State Transfer) adalah yang saat ini paling banyak digunakan untuk mengembangkan aplikasi berukuran enterprise maupun personal. Teknologi tersebut memanfaatkan sebuah protokol yang sudah disediakan oleh HTTP untuk berkomunikasi pada database. Implementasi dari REST yang memiliki tingkat kehandalan yang tinggi membuat para pengembang aplikasi web kemudian mulai berpindah dan mengimplementasikan sistem menggunakan jenis arsitektur n -tier application yang semulanya bertipe client-server.

Berbagai implementasi dalam merealisasikan *n-tier application* merupakan sebuah komoditi tersendiri. Berbagai framework ditawarkan untuk membangun sebuah *middleware* dengan kompatibilitas dan kemampuan yang beragam. Jenis dari framework terbagi menjadi dua yaitu full-stack framework dan microframework. Pertama, full-stack framework merupakan sebuah framework dalam kesatuan utuh. Kesatuan utuh yang dimaksud adalah sebuah framework tersebut sudah memiliki library untuk mengakses database, template untuk desain purwarupa, dan manajemen terhadap session. Sebuah full-stack framework sangat cocok untuk membangun sebuah aplikasi utuh, semisal berbasis web, karena sudah memiliki kemampuan yang lengkap. Framework seperti Android Studio, Laravel, CodeIgniter, dan Unity3D merupakan salah satu contoh dari full-stack framework. Kedua, microframework merupakan sebuah versi ‘kecil’ dari full-stack framework. Fitur yang ditawarkan pada microframework pada kondisi default adalah protokol HTTP. Secara spesifik, microframework digunakan untuk membangun sebuah API untuk services atau aplikasi tertentu. Beberapa *microframework* yang populer saat ini adalah Slim, Spring Boot, Lumen, Camping, dan Bootle. Oleh sebab itu, dalam berbagai situasi setiap performa dari masing-masing produk memiliki kemampuan yang berbeda.

Performa pada sebuah aplikasi berukuran enterprise merupakan faktor penting (Dirgahayu & Utama, 2004). Aplikasi berskala enterprise memiliki tingkat transaksi data yang sangat tinggi, contohnya adalah aplikasi e-commerce. Dalam trennya saat ini, pengguna belanja online semakin lama semakin mempercayai membeli barang melalui retail online. Retail online tersebut merupakan solusi kemudahan belanja yang nyaman dengan memanfaatkan teknologi saat ini. Untuk itu, aplikasi tersebut disarankan untuk memiliki performa yang tinggi. Pentingnya sebuah performa pada era transaksi online perlu didukung dengan adanya kualitas pengembangan. Terdapat berbagai macam pilihan untuk mengembangkan perangkat lunak berbasis middleware. Sebuah middleware merupakan solusi untuk menyelesaikan permasalahan terhadap kebutuhan transaksi yang semakin meningkat. Kebutuhan transaksi yang meningkat harus dibarengi dengan kenyamanan dan kecepatan transaksi. Dari sekian banyak itu, tentu saja perlu di kaji dari sisi performa untuk menentukan mana yang lebih cocok dari sisi performa untuk di kembangkan sesuai kebutuhan. Kebutuhan yang tepat akan menghasilkan produk yang efisien dan memiliki performa tinggi.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini memiliki fungsi untuk memberikan penjabaran masalah yang berkaitan dengan topik penelitian. Rumusan masalah penelitian adalah sebagai berikut:

- a. Bagaimana melakukan pengujian kinerja terhadap *middleware*?
- b. Apa hasil yang didapat dalam pengujian *middleware*?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini bertujuan untuk mengidentifikasi ruang lingkup pada suatu masalah serta menjadi memberikan penjelasan mengenai faktor yang mempengaruhi sebuah penelitian. Batasan masalah penelitian adalah sebagai berikut:

- a. Aplikasi yang dibuat menggunakan sebuah *micro framework* yang populer pada teknologi Java dan PHP (studi kasus layanan e-commerce).
- b. Aplikasi hanya memuat simulasi pada bagian kritis pada situs e-commerce.
- c. Tidak memiliki tampilan. Hanya sebuah *services* yang berjalan.

1.4 Tujuan Penelitian

Tujuan penelitian merupakan hasil yang didapatkan atas analisis dari tujuan masalah. Tujuan penelitian adalah sebagai berikut:

- a. Mengetahui pemilihan indikator perbandingan pada aplikasi *middleware*.
- b. Mendapatkan hasil perbandingan performa berupa tabel dan grafik pada tiap teknologi *middleware*.

1.5 Usulan Solusi

Usulan solusi merupakan representasi dalam mengerjakan penelitian dan penjabaran hasil melalui sebuah penjelasan. Usulan solusi penelitian adalah sebagai berikut:

- a. Menggunakan aplikasi Apache JMeter untuk menguji performa pada *middleware*
- b. Sebuah tabel dan grafik komparasi yang menyatakan kekuatan performa pada masing masing teknologi yang dikembangkan.

1.6 Langkah Penyelesaian

Langkah penyelesaian penelitian merupakan langkah-langkah yang diambil untuk menyelesaikan sebuah penelitian. Langkah penyelesaian penelitian adalah sebagai berikut:

- a. Studi Pustaka untuk menentukan metode pengujian performa.
- b. Membangun sebuah REST API menggunakan *micro framework* Spring Boot untuk Java dan Slim untuk PHP.
- c. Melakukan pengujian *service* menggunakan aplikasi Apache JMeter.
- d. Untuk melakukan *interface* pada REST API menggunakan Postman.

1.7 Sistematika Penulisan

Sistematika yang dibuat oleh penulis berfungsi untuk mempermudah pembaca dalam mengkaji laporan penelitian. Sistematika penulisan dibuat sesuai dengan kebutuhan dari jenis penelitian yang terkait. Berikut adalah sistematika yang digunakan :

- a. BAB I PENDAHULUAN
Menjelaskan latar belakang masalah serta tujuan secara umum.
- b. BAB II LANDASAN TEORI
Menjelaskan apa saja yang menjadi dasar pengetahuan pada penelitian.
- c. BAB III METODOLOGI PENELITIAN
Memberikan gambaran mengenai perancangan serta analisis dalam penelitian.

d. **BAB IV HASIL DAN PEMBAHASAN**

Menjelaskan implementasi dari perancangan dan analisis dari temuan saat penelitian.

e. **BAB V KESIMPULAN DAN SARAN**

Menjelaskan intisari dari temuan penelitian.

BAB II LANDASAN TEORI

2.1 Konsep Dasar Arsitektur N-Tier

Bagian ini akan menjelaskan definisi dari arsitektur N-Tier yang akan memberikan gambaran umum mengenai sebuah konsep dan komponennya.

2.1.1 Definisi Arsitektur N-Tier

Arsitektur N-Tier adalah perangkat lunak yang memiliki beberapa lapisan yang dapat membedakan peran pada masing masing lapisannya. Pada lapisan *presentation* akan memiliki perbedaan peran pada lapisan *logic* dan lapisan data (Martin & Amir, 2015).

N-Tier adalah sebuah sistem yang setidaknya memiliki tiga bagian *logic* yang terpisah. Setiap bagian tersebut bertanggung jawab atas fungsinya secara spesifik (BCM, 2018).

Dapat disimpulkan dari definisi diatas yaitu arsitektur *N-Tier* adalah seperangkat sistem yang memiliki lapisan-lapisan yang memiliki fungsi dan kerja secara spesifik.

2.1.2 Komponen Arsitektur N-Tier

Komponen dari arsitektur *N-Tier* yaitu sebagai berikut:

- a. Presentation Tier
- b. Logic Tier
- c. Data Tier

Lapisan paling atas, yaitu ***presentation tier***, dalam arsitektur *N-Tier* memiliki bagian yang bertanggung jawab atas penerjemahan dan interaksi sehingga maksud dari aplikasi dan data yang ditampilkan dapat dipahami oleh *user*.

Lapisan yang memiliki peran untuk mengeksekusi perintah, koordinasi antar lapisan, dan memberikan keputusan terhadap perintah yang diterima merupakan peran dari ***logic tier***. Lapisan itu berfungsi sebagai pengantar dan pemroses data antar dua lapisan yang berada di sekitarnya (Baxi, 2016).

Selanjutnya, untuk data-data yang diterima melalui *database* atau sistem dari bawaan itu sendiri memerlukan sebuah aplikasi. Lapisan yang memiliki peran tersebut adalah ***data tier***. Fungsi dari lapisan tersebut adalah melakukan penyimpanan secara permanen, atau bisa sementara, data yang diterima melalui antarmuka atau pun data yang sudah ada di dalam

sistem itu sendiri. Keberadaan data tersebut hanya dapat diakses melalui aturan yang sudah diterapkan pada lapisan sebelumnya.

2.2 Konsep Dasar API

Bagian ini akan menjelaskan konsep, fungsi, dan contoh dari API yang relevan untuk memberikan penjelasan secara mendasar bagaimana sebuah API bekerja.

2.2.1 Definisi API

API adalah sebuah perantara yang memiliki fungsi, aturan, dan standar yang digunakan untuk berkomunikasi antar perangkat lunak sebagai kriteria proses bisnis dalam sebuah sistem (CA Technologies, 2015).

API adalah pembawa pesan, yang membawa data antar aplikasi, *databases*, dan perangkat lainnya. Proses tersebut memiliki asset proses bisnis: memiliki nilai informasi, layanan atau produk. API memberikan filter, menciptakan *checkpoint* untuk data agar bisa sampai ke developer, aplikasi, dan *end user* (Fredrich, 2013).

Dapat disimpulkan, dari definisi diatas bahwa API adalah sebuah perantara yang memiliki fungsi mengatur laju data yang terikat pada sebuah aturan yang sudah diterapkan oleh API tersebut.

2.2.2 Fungsi API

Pada implementasinya terdapat beberapa fungsi dalam proses bisnis dari sebuah aplikasi dengan memanfaatkan API yaitu:

- a. Dapat mengintegrasikan konten dari mitra untuk melakukan lintas transaksi supaya memaksimalkan kesempatan.
- b. Menciptakan alur bisnis baru dan memperluas penjualan dan penawaran suatu produk dengan membagi data melalui cara baru.
- c. Menguatkan *brand* dengan memberikan pengalaman secara konsisten, seragam, dan secara pribadi lintas perangkat
- d. Memberikan keleluasaan dalam *reusability*, menjadikan integrasi pada mitra baru menjadi lebih cepat

2.2.3 Contoh API

Saat ini, sudah banyak *platform* yang menawarkan dalam mengembangkan aplikasi melalui API mereka sendiri. Menurut Nikko Bautista, aplikasi yang dibuat berdasarkan API menggunakan eksekusi langsung berdasarkan fungsi dari API itu sendiri. Beberapa contoh perusahaan yang menyediakan API secara *public* adalah sebagai berikut:

- a. **Twitter Developers**, menyediakan layanan untuk membuat aplikasi pihak ketiga. Beberapa hal yang bisa dilakukan aplikasi pihak ketiga itu adalah menerima, membalas, menghapus, dan mengedit sebuah *tweet*. Ada pula pengaturan profil, mengganti password dan sebagainya.
- b. **YouTube | Google Developers**, menyediakan layanan untuk membuat aplikasi pihak ketiga yang dapat digunakan untuk para *developer*. Beberapa layanan tersebut meliputi mengambil data video *trending*, berlangganan *channel*, mengunggah video dan sebagainya.
- c. **Camera Remote API by Sony**, menyediakan layanan pihak ketiga untuk dapat membuat sebuah aplikasi yang dapat mengatur *shutter speed*, tingkat kecerahan, diafragma, dan sebagainya.
- d. **LinkedIn Developer Network**, menyediakan layanan pihak ketiga. Aplikasi dapat menampilkan dan mengedit profil, mengundang teman, memberikan rekomendasi dan sebagainya.

2.3 Konsep Middleware

Pada bagian ini akan menjelaskan secara mendasar dari sebuah teknologi *middleware* tersebut untuk memberikan gambaran secara mendasar.

2.3.1 Definisi Middleware

Dalam definisinya, *middleware* adalah definisi umum dari perangkat lunak yang berperan untuk “mengelem bersama” sesuatu yang terpisah, yang sudah kompleks atau sudah ada, pada program. Beberapa komponen perangkat lunak memiliki keterikatan dengan *middleware* termasuk aplikasi berskala *enterprise* atau layanan web (TechTarget, 2015).

Teknologi *middleware* memiliki kesamaan dengan sistem operasi karena memberikan dukungan pada aplikasi, memberikan fasilitas interaksi secara komputasional antar komputer yang berbeda jaringan melalui layanan komunikasi (BCM, 2018).

Teknologi *middleware* merupakan sebuah definisi umum dari sebuah perangkat lunak, sedangkan API adalah sebuah definisi secara teknis pada implementasinya (Suominen, 2017).

Definisi dari *middleware* adalah sebuah perangkat lunak yang menghubungkan koneksi *end-to-end transaction*. Pada implementasinya di API, apa pun yang memenuhi jenis *request*, maka API tersebut akan disebut sebagai *middleware*.

Dapat disimpulkan dari definisi diatas, *middleware* merupakan sebuah perangkat lunak yang berfungsi sebagai jembatan penghubung antar lapisan sebuah sistem yang memberikan fasilitas untuk saling berkomunikasi melalui sebuah jalur komunikasi yang sudah diterapkan.

2.3.2 Jenis Middleware

Pengembangan aplikasi dengan memanfaatkan *middleware* sudah memiliki pilihan yang banyak. Penggunaan produk *middleware* disesuaikan dengan kebutuhan para pengembang aplikasi. Pembuatan *middleware* tersebut menggunakan sebuah *microframework*. Pada *microframework* terdapat fasilitas yang dapat digunakan untuk membangun sebuah *middleware*. Perangkat lunak tersebut, menurut Jeff Knupp, memiliki fungsi untuk menciptakan protokol menggunakan beberapa bahasa pemrograman sesuai dengan basis dari *microframework* tersebut. Beberapa contohnya adalah sebagai berikut:

- a. Spring Boot.
- b. Flask.
- c. Zend Expressive.
- d. Slim.
- e. Express.js.
- f. Lumen
- g. Nancy FX
- h. Yesod.
- i. Jooby.
- j. Grape.
- k. Camping.
- l. Bottle.
- m. Silex.
- n. Symfony.
- o. Kemal.

Pada penelitian kali ini, pengembangan *middleware* untuk melakukan pengujian terhadap performa PHP menggunakan *microframework* Slim dan untuk Java digunakan Spring Boot. Hal tersebut, menurut penulis, merupakan yang paling sering digunakan untuk pengerjaan skala besar maupun kecil, serta memiliki fitur yang ringkas namun esensial.

2.3.3 Spring Boot

Spring Boot merupakan cara mudah untuk membuat Aplikasi berbasis Spring yang dapat berdiri sendiri, yang bisa dijalankan melalui perintah independen. Spring mengambil pandangan tentang *platform* dan *library* pihak ketiga sehingga bisa memulai dengan konfigurasi yang minimal. Sebagian besar aplikasi Spring Boot memerlukan konfigurasi Spring yang sangat sedikit.

Spring memiliki fitur-fitur sebagai berikut:

- a. Membuat aplikasi *stand-alone*.
- b. Dapat dipasangkan dengan Tomcat, Jetty, atau Undertow secara langsung (tanpa perlu *men-deploy* file WAR).
- c. Memberikan opsi '*starter*' untuk konfigurasi Maven.
- d. Konfigurasi oleh Spring dapat dilakukan secara otomatis.
- e. Menyediakan fitur yang sudah siap untuk skala produksi.
- f. Tanpa *code generation* dan XML pada konfigurasi,

2.3.4 Slim

Slim adalah *microframework* PHP yang digunakan untuk menulis aplikasi web dan API sederhana namun canggih dengan mudah. Definisi dari *microframework* sendiri adalah sebuah *framework* minimalis yang hanya menawarkan fungsi 'esensial' yang biasanya didesain khusus untuk pembuatan API. Beberapa fitur unggulan dari Slim adalah sebagai berikut:

- a. HTTP Router yang memberikan HTTP *request* dan URI.
- b. Memiliki fokus pada pembuatan aplikasi *Middleware*.
- c. PSR-7 HTTP adalah implementasi dari pesan untuk dapat memeriksa dan mengubah HTTP *method*, status, URI, *headers*, *cookies*, dan *body*.
- d. Sudah mendukung *Dependency Injection*.

2.4 Konsep Dasar Performance Test

Bagian ini akan menjelaskan konsep serta pemahaman yang diperlukan mengenai uji coba performa pada sebuah aplikasi.

2.4.1 Definisi Performance Test

Definisi dari *performance test* adalah sebuah proses yang mengidentifikasi efektivitas dari sebuah komputer, jaringan, perangkat lunak atau perangkat. Proses yang dilakukan yaitu mengidentifikasi secara kualitatif di dalam laboratorium, yaitu mengukur waktu respon (Hass, 2008).

Dilansir dari Microsoft, adalah sebuah proses pengecekan secara bertahap atau untuk melakukan validasi terhadap kecepatan, *stability* dan *scalability* pada sebuah aplikasi berdasarkan pada kebutuhan produk.

Melalui definisi diatas, dapat disimpulkan bahwa *performance test* adalah seperangkat metode untuk mengetahui waktu respon, kekuatan *scalability*, dan *stability* dari sebuah produk aplikasi.

2.4.2 Manfaat Performance Test

Terdapat beberapa manfaat dari penggunaan metode tes tersebut untuk menguji perangkat lunak (Patton, 2001). Manfaat tersebut adalah sebagai berikut:

- a. Meningkatkan kualitas produk akhir.
- b. Memberikan kenyamanan dalam melakukan transaksi sesuai dengan permintaan produksi
- c. Memberikan sebuah gambaran standar produksi pada sebuah aplikasi yang akan dikembangkan.

2.4.3 Jenis Performance Test

Performance Test adalah untuk memberikan validasi atas kecepatan, kestabilan dan keleluasaan sebuah sistem supaya memberikan sebuah gambaran pada hasil akhir dari sebuah aplikasi yang diujikan berdasarkan ketentuan-ketentuan yang akan dilaksanakan selama proses pengujian (Murawski, Keck, & Schnaible, 2014). Dilansir dari situs Microsoft terdapat sejumlah tipe pengujian performa. Beberapa diantaranya adalah sebagai berikut:

- a. Load Test, untuk melakukan verifikasi terhadap aktivitas dari aplikasi tersebut apabila diberikan situasi yang normal untuk sebuah sistem yang diujikan.

- b. Stress Test, untuk mengidentifikasi perilaku aplikasi yang diuji melalui rekayasa kondisi diatas normal pada sebuah sistem yang diujikan.

BAB III

METODOLOGI PENELITIAN

3.1 Peninjauan Lingkungan Pengujian

Peninjauan pengujian sistem adalah untuk mendalami kelayakan sistem. Aktivitas untuk menguji kelayakan sistem bertujuan untuk memahami lingkungan yang akan diimplementasikan. Dengan adanya studi kelayakan, sistem akan berjalan dapat memiliki manfaat yang lebih efisien.

3.1.1 Definisi Ruang Lingkup Pengujian Sistem

Pengujian yang dilakukan untuk mengetahui kecepatan dari middleware dilakukan menggunakan studi kasus *e-commerce*. Pada kasus *e-commerce*, modul yang dilakukan pengujian merupakan modul transaksi jual dan beli yang akan menampilkan data pelanggan secara keseluruhan. Transaksi jual dan beli tersebut merupakan bagian paling kritis dalam sebuah aplikasi *e-commerce*. Banyak pengguna aplikasi *e-commerce* datang untuk membeli barang yang sudah tersedia. Pada modul transaksi menampilkan harga barang, jumlah barang, pemilihan metode pembayaran, pemilihan metode pengiriman, dan sebagainya bergantung pada jenis barang yang dipilih. Oleh sebab itu, pengujian dapat dilakukan pada modul tersebut mengingat transaksi yang dilakukan pada *e-commerce* cenderung masif.

3.1.2 Penentuan Lingkup Pengujian

Untuk keperluan pengujian, penulis menggunakan jaringan dari VirtualBox sebagai kanal untuk melakukan pengujian. Faktor luar yang akan di kontrol dengan penggunaan koneksi dari VirtualBox adalah sebagai berikut:

- a. Menggunakan mesin yang berbeda supaya tidak mengganggu proses pengujian.
- b. Melakukan isolasi terhadap faktor seperti spesifikasi komputer yang di bebaskan.
- c. Tanpa menggunakan *database* karena waktu *request* dapat berbeda dan memiliki interval sendiri.

Perancangan ini berfungsi untuk menentukan faktor-faktor yang menjadi ruang lingkup pada pengujian yang akan dilakukan. Beberapa hal yang perlu diperhatikan adalah sebagai berikut:

- a. Platform Coverage Client:
 - 1. OS: Windows 10
 - 2. RAM: 4096 MB
 - 3. System Manufacturer: Lenovo
 - 4. Model: 80U2
 - 5. Processor: Intel(R) Celeron(R) CPU N3350 @1.10 GHz (2 CPU), ~1.1GHz
 - 6. DirectX: Version 12
 - 7. Browser: Firefox Quantum 59.0.2 (64 bit)
- b. Network Connection:
 - 1. Type: Host-Only Adapter
 - 2. Adapter: Intel PRO/1000 MT Desktop (82540EM)
- c. Platform Coverage Server (PHP):
 - 1. OS: Ubuntu Server 16.04 LTS
 - 2. RAM: 1024 MB
 - 3. System Manufacturer: Lenovo
 - 4. Processor: Intel(R) Celeron (R) CPU N3350 @1.1GHz
 - 5. PHP Version: 7.0
 - 6. HDD: 20 GB
- d. Platform Coverage Server (Java):
 - 1. OS: Ubuntu Server 16.04 LTS
 - 2. RAM: 1024 MB
 - 3. System Manufacturer: Lenovo
 - 4. Processor: Intel(R) Celeron (R) CPU N3350 @1.1GHz
 - 5. Java Version: 10.0
 - 6. HDD: 20 GB

3.1.3 Kekangan

Menjelaskan kompatibilitas pada pengujian sistem serta beberapa prasyarat sebelum dilakukan sebuah pengujian. Beberapa faktor yang menjadi batasan tersebut adalah:

- a. Sistem berjalan pada jaringan lokal.
- b. Pengujian dilakukan tanpa perantara antarmuka aplikasi.
- c. Hanya pengujian untuk *load testing*.

3.2 Pengembangan Sistem Untuk Pengujian

Pengembangan sistem adalah tahap untuk melakukan pengembangan pada lingkungan sistem dengan mencari sumber referensi pengujian sistem untuk mendapatkan gambaran pada kompatibilitas pengujian sistem yang akan dikerjakan.

3.2.1 Klasifikasi Masalah

Batasan sistem berguna untuk memberikan ruang lingkup/konteks untuk sistem supaya dapat bekerja secara optimal. Beberapa permasalahan pada modul transaksi jual dan beli adalah sebagai berikut:

- a. Kecepatan transaksi dapat melambat apabila mendapat *request* yang melebihi kapasitas.
- b. Apabila dalam menampilkan daftar barang beserta parameternya terasa lambat, maka akan mengurangi kenyamanan pengguna.

3.2.2 Batasan Sistem

Batasan sistem berguna untuk memberikan ruang lingkup/konteks untuk sistem supaya dapat bekerja secara optimal. Sistem dapat bekerja secara fokus dan efisien dengan adanya beberapa batasan berikut :

- a. Sistem bekerja dengan media komputer.
- b. Sistem memiliki fitur output data berformat JSON.
- c. Sistem tidak menggunakan antarmuka GUI (Graphical User Interface).

3.3 Analisis Pengujian Sistem

Analisis sistem dilakukan untuk mendapatkan gambaran secara teknis mengenai sistem yang akan diuji. Analisis sistem yang diterapkan yaitu menampilkan gambaran berupa pemilihan indikator pengujian , arsitektur pengujian, dan Use Case Diagram.

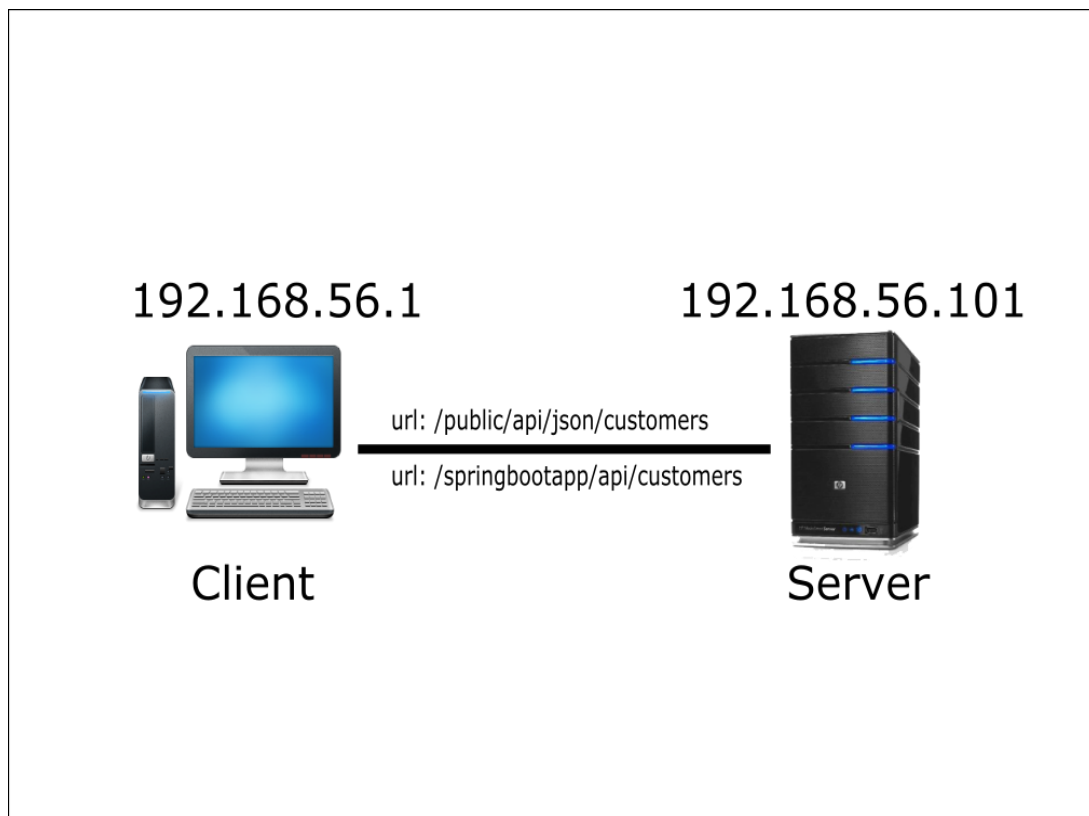
3.3.1 Indikator Pengujian

Untuk keperluan pengujian yang akan dikerjakan, pemilihan indikator pengujian akan menjadi faktor dalam menentukan hasil dari pengujian sistem. Beberapa hal yang menjadi indikator dalam pengujian aplikasi adalah sebagai berikut:

- a. Pemilihan *hardware*.
- b. Pemilihan penanganan jeda interval tiap iterasi uji.
- c. Pemilihan jumlah *thread* pada pengujian.

3.3.2 Arsitektur Pengujian Sistem

Arsitektur sistem merupakan sebuah gambaran teknis pada lingkup aplikasi yang akan dijalankan. Arsitektur sistem berfungsi sebagai landasan sebuah aplikasi yang akan dibangun sehingga dapat dimengerti oleh para pengembang aplikasi. Pada pengujian sistem, arsitektur dapat menjelaskan konteks aplikasi yang akan diujikan. Berikut adalah arsitektur dalam pengujian aplikasi *middleware* dapat dilihat pada gambar 3.1 berikut.

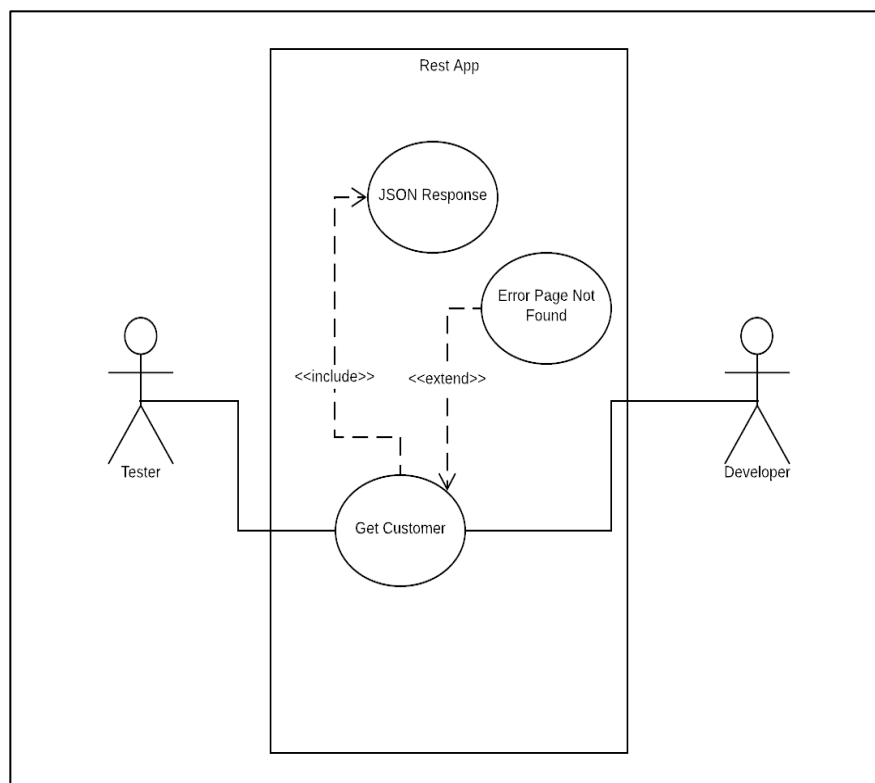


Gambar 3.1 Arsitektur pengujian

Pada arsitektur pengujian tersebut menggunakan sistem jaringan lokal yang di desain menurut kebutuhan dari pengujian. Komputer klien akan bertugas sebagai pengirim *request* untuk diteruskan menuju *server*. Klien juga akan menerima jawaban *request* yang diambil dari *server* itu sendiri. Bagian *server* kemudian akan memberikan respon berdasarkan permintaan dari komputer klien itu. Data yang di dapat dari *service* tersebut merupakan bagian dari aplikasi. Penggunaan *database* tidak dianjurkan karena bagian tersebut merupakan faktor lain dari pengujian sehingga akan berpengaruh pada hasil akhir dari uji performa ini.

3.3.3 Use Case Diagram

Use Case Diagram merupakan sebuah rancangan relasi antar *user* dan sistem dengan memberikan spesifikasi terhadap konstruksi pada sebuah sistem yang akan dikerjakan. Penggunaan use case dapat mempermudah dalam menjelaskan lingkup sistem yang bekerja untuk pengujian performa yang akan dikerjakan. Berikut adalah implementasi dari Use Case Diagram dapat dilihat pada gambar 3.2 berikut.



Gambar 3.2 Diagram Use Case

Pada diagram use case tersebut dijelaskan bahwa sebuah aplikasi REST tersebut memiliki sebuah simulasi *activity* yaitu mendapatkan data pelanggan (Get Customer). Simulasi *activity* itu akan memberikan sebuah respon data berbentuk JSON (JSON Response) saat tester mengirimkan sebuah *request* dengan alamat yang tepat. Developer tersebut akan memberikan sebuah data yang sudah disajikan kepada tester untuk ditampilkan. Sebagai penanganan kesalahan, aplikasi menyediakan halaman *error* sebagai petunjuk untuk tester dalam memahami sebuah jawaban dari *request* tersebut.

3.4 Pengujian Kinerja

Pengujian kinerja memerlukan sebuah rekayasa lingkungan supaya mendapatkan hasil yang serupa dengan kondisi yang nyata. Kondisi tersebut akan memberikan gambaran yang sedikit banyak memberikan informasi umum mengenai apa saja yang menjadi variabel yang diujikan. Perencanaan pengujian meliputi pemilihan *tools*, *test plan*, dan grafik hasil uji.

3.4.1 Pemilihan Perangkat Uji

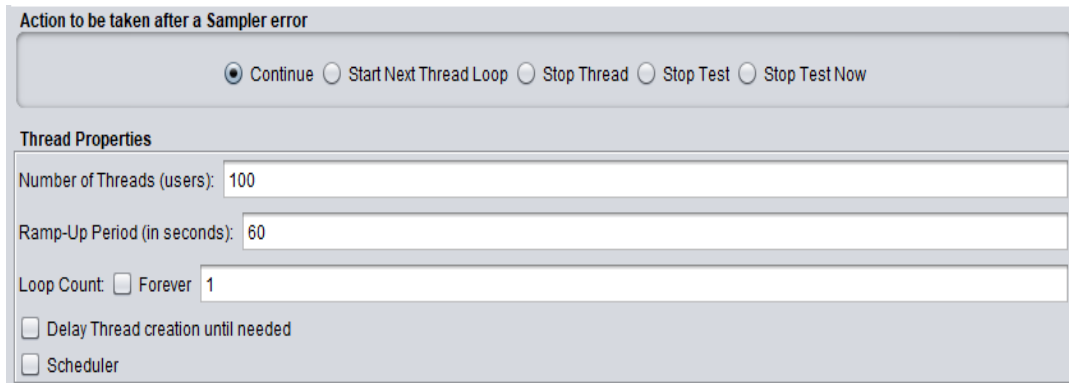
Pengujian kali ini, penggunaan alat yang digunakan untuk melakukan uji aplikasi adalah dengan menggunakan Apache JMeter. Aplikasi tersebut berjalan pada Java sehingga tidak terikat pada suatu sistem operasi sendiri. Berikut adalah beberapa fitur dari JMeter dilansir dari situsnya:

- a. Dapat melakukan load dan performance testing pada aplikasi dengan berbagai tipe yang sudah umum (Web HTTP, SOAP/REST, FTP, dsb).
- b. Memiliki *record* dari aplikasi *native* sendiri.
- c. Laporan dapat dijadikan HTML.
- d. Format respon dapat dari berbagai format seperti HTML, JSON, XML, dan berbagai bentuk teks lainnya.

3.4.2 Test Plan

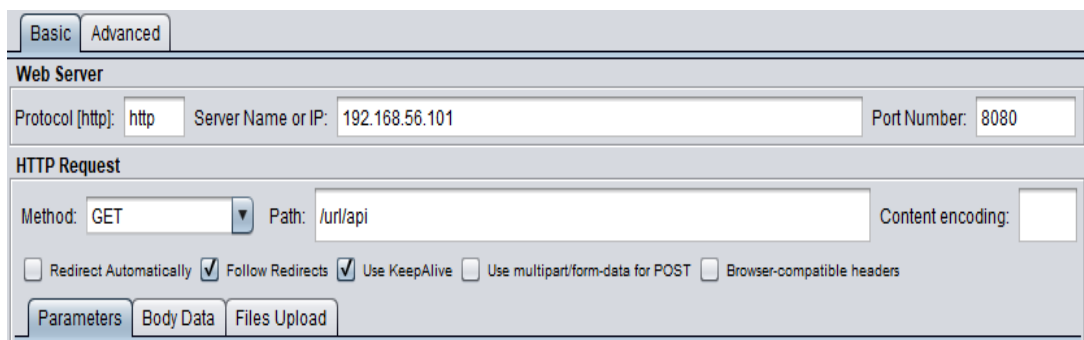
Test Plan berfungsi untuk memberikan *roadmap* pada sebuah aktivitas pengujian. Dalam pengujian tersebut dapat dilakukan dengan beberapa tahapan berikut pada aplikasi JMeter:

- a. Pada tahapan ini, penguji dapat melakukan penambahan *thread* baru. *Thread* baru merupakan sebuah test plan baru untuk sebuah aplikasi. Fase ini berfungsi untuk menentukan beban kerja yang akan dilakukan pada tiap iterasi pengujian. Pada contoh *thread* berikut akan diberikan sebuah contoh pengujian pada pemasangan *thread* berjumlah 100 *users* dengan waktu periode pengujian selama 60 detik dengan jumlah loop sebanyak satu kali. Contoh penambahan *thread* dapat dilihat pada gambar 3.3 berikut.



Gambar 3.3 Contoh *thread group*

- b. Pada tahap berikutnya, melakukan konfigurasi pada HTTP untuk dapat membuat sebuah koneksi pada *request* yang sudah dibuat. Koneksi HTTP berfungsi untuk mengambil data yang sudah ditetapkan pada tahap perancangan. Konfigurasi HTTP berisi jenis protokol, nama *server* atau IP, nomor *port*, jenis metode HTTP, dan jalur atau url. Untuk keperluan contoh pengujian, konfigurasi protokol yang akan dijalankan memiliki protokol http, alamat IP 192.168.56.101 dengan nomor *port* 8080, akan dikirim sebuah *request* GET dengan alamat url /url/api. Implementasi contoh dapat dilihat pada gambar 3.4 berikut.



Gambar 3.4 Contoh HTTP Request

- c. Membuat grafik *listener* merupakan tahap untuk melihat parameter respon pengujian pada aplikasi yang akan diuji. Representasi data menggunakan grafik dapat memberikan kemudahan dalam melihat secara keseluruhan hasil yang sudah didapat saat melakukan pengujian. Penggunaan grafik juga dapat merepresentasikan sebuah performa dari masing-masing teknologi menjadi lebih terlihat secara kualitatif. Contoh grafik dapat dilihat pada gambar 3.5 berikut.



Gambar 3.5 Contoh grafik Request

- d. Membuat tabel perbandingan performa untuk memberikan gambaran dari hasil pengujian yang akan dikerjakan. Tabel dalam pengujian ini akan berfungsi sebagai monitor dalam pelaksanaan pengujian sehingga hasil respon dan nilai rata-rata pada tiap teknologi dapat diamati. Contoh tabel dapat dilihat pada gambar 3.6 berikut.

Sample #	Start Time	Thread Na...	Label	Sample Ti...	Status	Bytes	Sent Bytes	Latency	Connect Ti...
36	18:06:53.384	Users(100)...	GET_CUS...	484	✓	11666	194	465	10
37	18:06:53.986	Users(100)...	GET_CUS...	368	✓	11666	194	259	68
38	18:06:54.587	Users(100)...	GET_CUS...	526	✓	11666	194	425	45
39	18:06:55.159	Users(100)...	GET_CUS...	445	✓	11664	194	197	23
40	18:06:55.782	Users(100)...	GET_CUS...	924	✓	11658	194	875	370
41	18:06:56.385	Users(100)...	GET_CUS...	1045	✓	11656	194	769	216
42	18:06:56.984	Users(100)...	GET_CUS...	1014	✓	11656	194	554	19
43	18:06:57.588	Users(100)...	GET_CUS...	783	✓	11666	194	762	431
44	18:06:59.386	Users(100)...	GET_CUS...	238	✓	11662	194	151	14
45	18:06:59.984	Users(100)...	GET_CUS...	284	✓	11658	194	283	35
46	18:07:00.583	Users(100)...	GET_CUS...	650	✓	11658	194	518	45
47	18:06:58.186	Users(100)...	GET_CUS...	3425	✓	11654	194	3259	85
48	18:07:01.184	Users(100)...	GET_CUS...	489	✓	11654	194	459	166
49	18:06:58.786	Users(100)...	GET_CUS...	3657	✓	11654	194	3442	33
50	18:07:01.785	Users(100)...	GET_CUS...	1214	✓	11666	194	915	217
51	18:07:02.364	Users(100)...	GET_CUS...	2028	✓	11652	194	1277	307
52	18:07:03.559	Users(100)...	GET_CUS...	1307	✓	11666	194	1005	253
53	18:07:04.204	Users(100)...	GET_CUS...	1076	✓	11666	194	789	273
54	18:07:02.961	Users(100)...	GET_CUS...	2659	✓	11666	194	1589	211
55	18:07:05.403	Users(100)...	GET_CUS...	1684	✓	11646	194	1558	582
56	18:07:06.004	Users(100)...	GET_CUS...	1264	✓	11646	194	1116	276
57	18:07:07.210	Users(100)...	GET_CUS...	898	✓	11672	194	686	249
58	18:07:07.804	Users(100)...	GET_CUS...	979	✓	11652	194	704	167
59	18:07:06.604	Users(100)...	GET_CUS...	2194	✓	11672	194	1232	455
60	18:07:04.803	Users(100)...	GET_CUS...	4046	✓	11652	194	3818	305

Scroll automatically?
 Child samples?
 No of Samples 63
 Latest Sample 556
 Average 896
 Deviation 810

Gambar 3.6 Contoh tabel Request

BAB IV

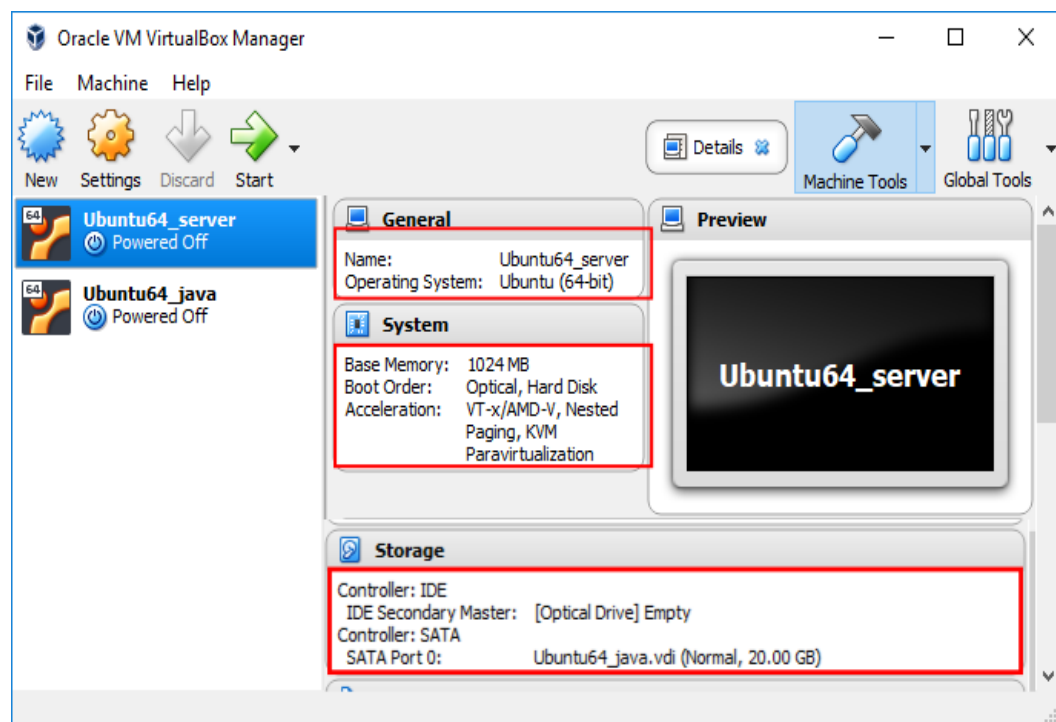
PEMBAHASAN

4.1 Implementasi Server

Server yang digunakan pada penelitian ini merupakan sebuah *server* independen yang beroperasi pada sebuah *virtual machine*. Penggunaan tersebut berguna untuk dapat memberikan simulasi yang riil terhadap pengujian performa pada masing-masing teknologi yang diujikan.

4.1.1 Konfigurasi Server

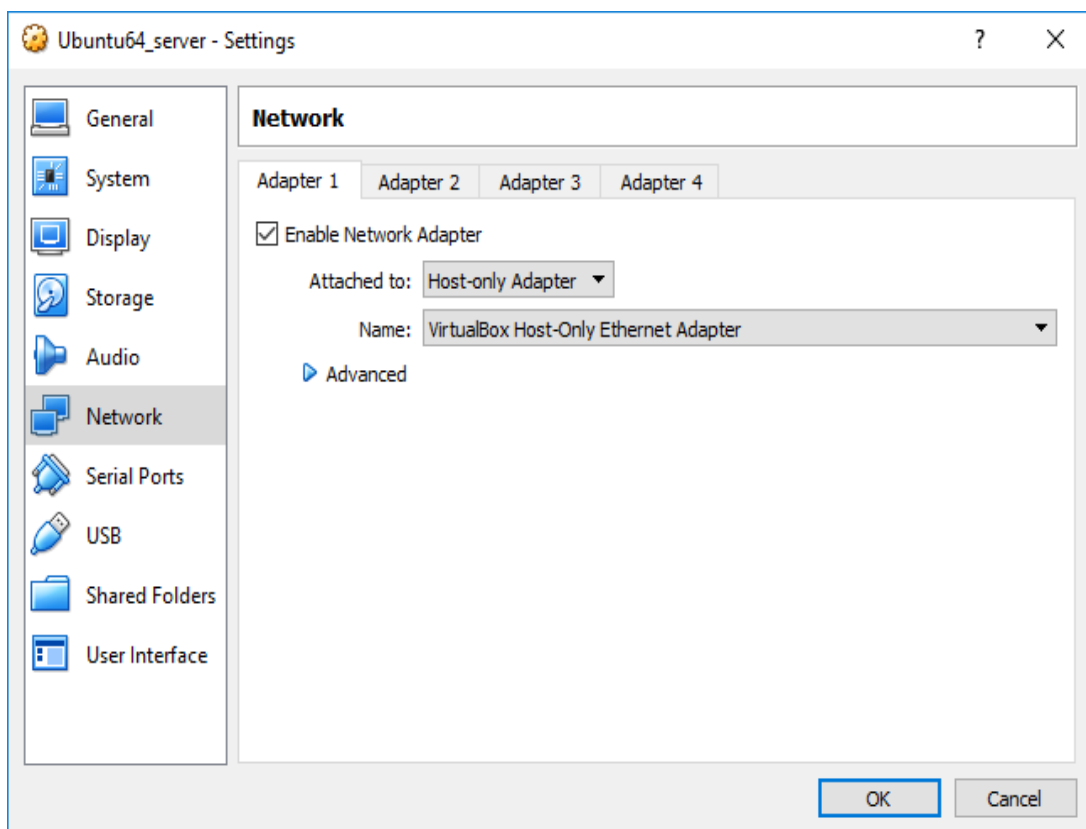
Konfigurasi server pada penelitian kali ini melibatkan sebuah perangkat lunak virtualisasi bernama VirtualBox. Aplikasi tersebut memberikan sebuah virtualisasi komputer yang akan digunakan untuk membagi *resource* untuk keperluan pengujian. Pembagian tersebut penting agar dapat mengisolasi kemampuan pada *service* masing-masing agar tidak terlibat gangguan dari komputer *client*. Konfigurasi dapat dilihat pada gambar 4.1 berikut.



Gambar 4.1 Konfigurasi Server

Pada konfigurasi tersebut dapat dilihat bahwa pembagian *resource* sudah dapat didefinisikan sendiri menurut kebutuhan dari pengujian. Mesin tersebut berjalan pada sistem operasi Ubuntu Server 16.04 LTS (with GUI), menggunakan prosesor Intel(R) Celeron (R) CPU N3350 @1.1GHz, RAM 1024 MB dan HDD 20 GB.

Sebagai keperluan untuk koneksi pada *client*, komputer virtual tersebut terhubung pada sebuah *adapter* yang sudah disediakan oleh VirtualBox sendiri. Konfigurasi koneksi tersebut dapat dilihat pada gambar 4.2 berikut.



Gambar 4.2 Koneksi untuk client

4.1.2 Uji Coba Server

Untuk melakukan uji coba, pada komputer *client* menggunakan sebuah perintah melalui cmd. Pertama kali dapat dilakukan adalah mengecek apakah IP dari *client* dan *server* sudah masuk dalam kelas yang sama. Perintah yang digunakan adalah perintah 'ipconfig'.

```

Command Prompt

Windows IP Configuration

Ethernet adapter Ethernet:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter VirtualBox Host-Only Network:

    Connection-specific DNS Suffix  . :
    Link-Local IPv6 Address . . . . . : fe80::c4fc:8223:4f8:bf1b%8
    IPv4 Address. . . . . : 192.168.56.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 1:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . :
    Link-Local IPv6 Address . . . . . : fe80::31a4:9cc5:611c:7011%14
    Autoconfiguration IPv4 Address. . : 169.254.112.17
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . :

```

Gambar 4.3 Pengecekan kelas IP

Sebagai komparasi terhadap bagian *server*, akan dilakukan juga sebuah pengecekan terhadap konfigurasi alamat IP yang disediakan. Alamat IP tersebut akan menunjukkan bahwa koneksi *client* dan *server* sudah terinisiasi. Inisiasi dari koneksi tersebut merupakan hasil dari pemilihan koneksi pada konfigurasi VirtualBox yang menggunakan host-only connection.

```

medium@ubuntuBox: ~
File Edit View Search Terminal Help
medium@ubuntuBox:~$ ifconfig
enp0s3  Link encap:Ethernet HWaddr 08:00:27:79:ca:b3
        inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0
        inet6 addr: fe80::a00:27ff:fe79:cab3/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:8 errors:0 dropped:0 overruns:0 frame:0
        TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:2832 (2.8 KB) TX bytes:10894 (10.8 KB)

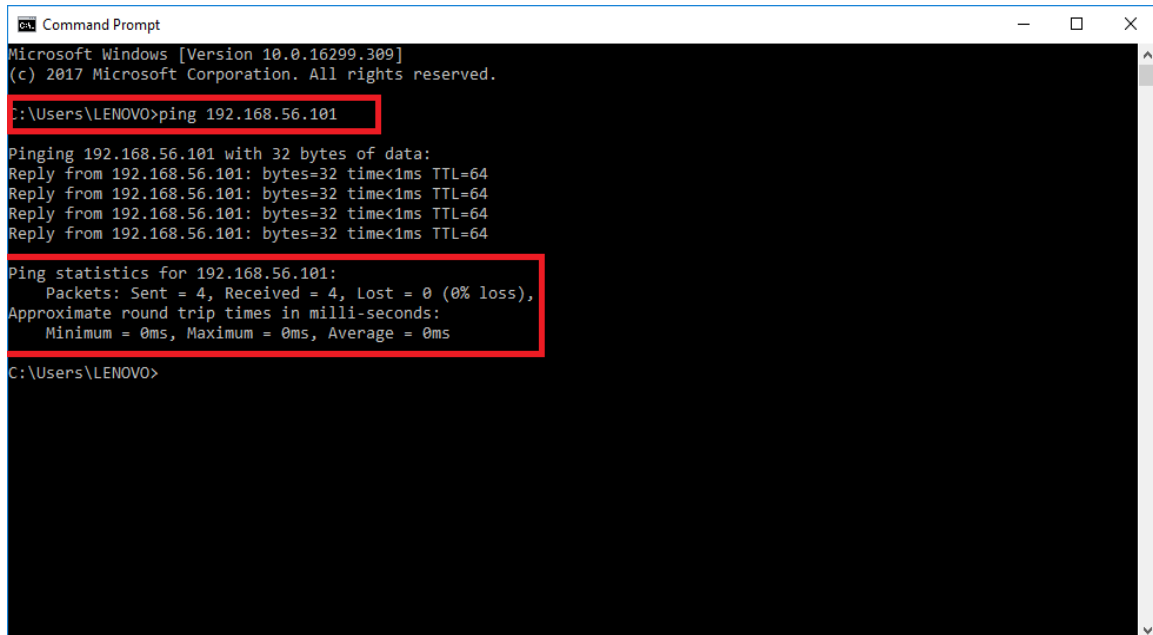
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING MTU:65536 Metric:1
        RX packets:4164 errors:0 dropped:0 overruns:0 frame:0
        TX packets:4164 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:252795 (252.7 KB) TX bytes:252795 (252.7 KB)

medium@ubuntuBox:~$ █

```

Gambar 4.4 Pengecekan kelas IP server

Pengecekan selanjutnya, setelah menggunakan perintah yang ada pada cmd. Perintah itu bertujuan untuk memanggil koneksi apakah sudah tersambung atau belum. Pada cmd sebuah perintah yang digunakan adalah 'ping'.



```
Command Prompt
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>ping 192.168.56.101

Pinging 192.168.56.101 with 32 bytes of data:
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64
Reply from 192.168.56.101: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.56.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\LENOVO>
```

Gambar 4.5 Memanggil koneksi ke server

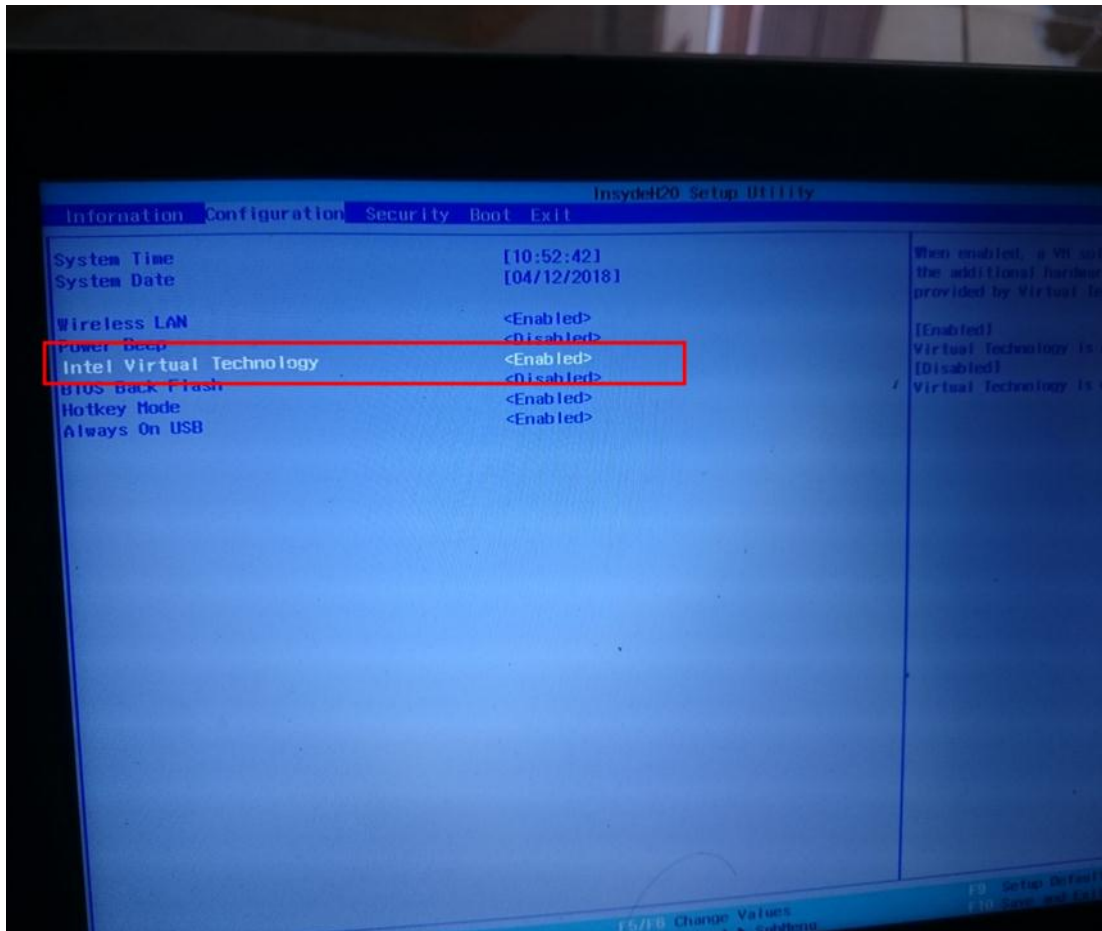
Dapat dilihat bahwa *client* dan *server* yang akan digunakan dalam pengujian sudah dalam satu jaringan yang telah ditetapkan.

4.2 Implementasi Client

Client yang digunakan pada penelitian ini merupakan sebuah *client* independen yang beroperasi pada sebuah *hardware* nyata. Penggunaan tersebut berguna untuk dapat memberikan simulasi yang riil terhadap pengujian performa pada masing-masing teknologi yang diujikan.

4.2.1 Konfigurasi Client

Konfigurasi yang digunakan pada pengujian kali ini menggunakan sebuah *hardware* nyata berupa laptop. Laptop tersebut akan menggunakan sebuah teknologi VMs (Intel Virtual Technology) yang terdapat pada konfigurasi BIOS laptop tersebut. VMs tersebut akan mengaktifkan laptop tersebut dapat menjalankan aplikasi VirtualBox dan dapat menggunakan koneksi Host-Only ke dalam komputer *virtual* yang berada pada mesin yang berbeda.

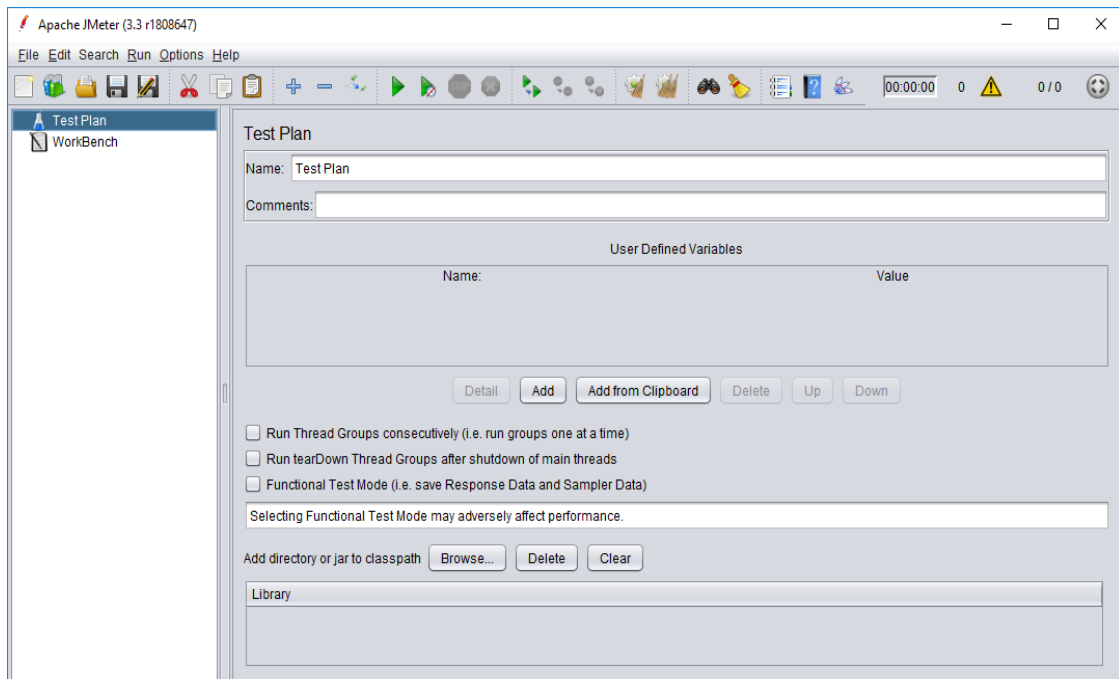


Gambar 4.6 Virtual Machine client

Dengan mengaktifkan Intel VT (Virtual Technology), maka sebuah komputer tersebut dapat melakukan pembagian *resource* secara independen. Pembagian tersebut akan berguna untuk menjalankan aplikasi VirtualBox supaya mesin dapat menjalankan sumber daya yang sudah didefinisikan sejak awal untuk keperluan *server*.

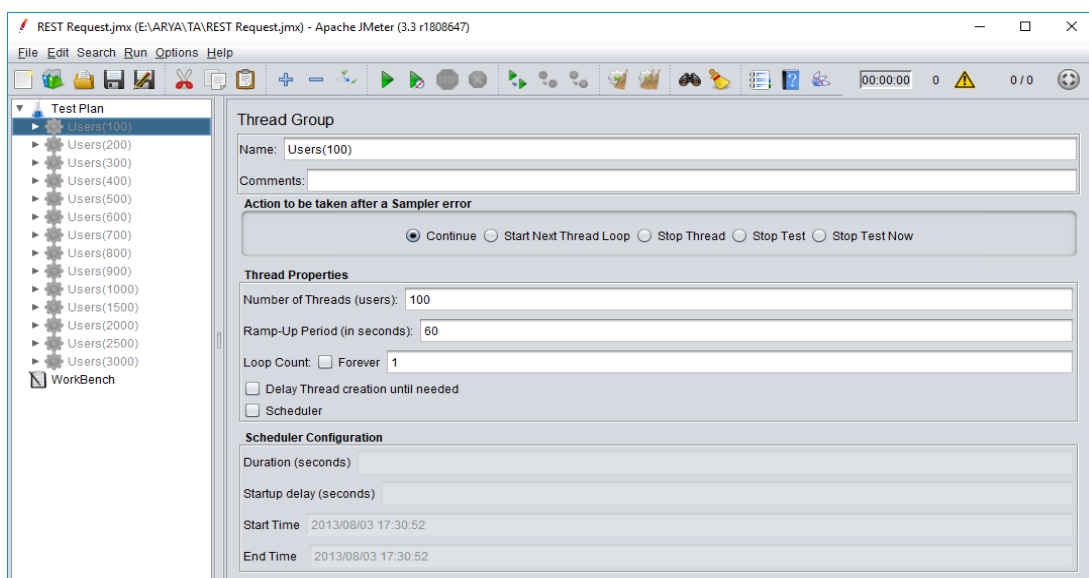
4.2.2 Konfigurasi JMeter

JMeter merupakan sebuah aplikasi berbasis JAVA sehingga termasuk aplikasi independen yang tidak terpengaruh pada sistem operasi sebuah komputer yang berjalan. Aplikasi tersebut akan memuat beberapa konfigurasi yang berfokus pada pembuatan *test plan* dan koneksi ke *service* yang akan diuji. Halaman awal JMeter dapat dilihat pada gambar berikut.



Gambar 4.7 Halaman awal JMeter

Pada inisiasi halaman muka JMeter, terdapat pilihan dalam membuat *test plan* yang dapat digunakan sesuai dengan kebutuhannya. Pada pengujian kali ini, penggunaannya akan berfokus pada jumlah *thread* yang diujikan. Jumlah itu akan menentukan seberapa waktu yang dibutuhkan pada sebuah *service* yang berjalan. Berikut adalah *thread* yang dapat dilihat dari gambar berikut.



Gambar 4.8 Thread Group

Bagian ini menjelaskan tentang bagaimana sebuah simulasi *user* yang akan dijalankan saat pengujian. Pada gambar diatas dijelaskan bahwa ada 100 *users* pada ukuran waktu tempuh 60 detik pada sekali *loop*.

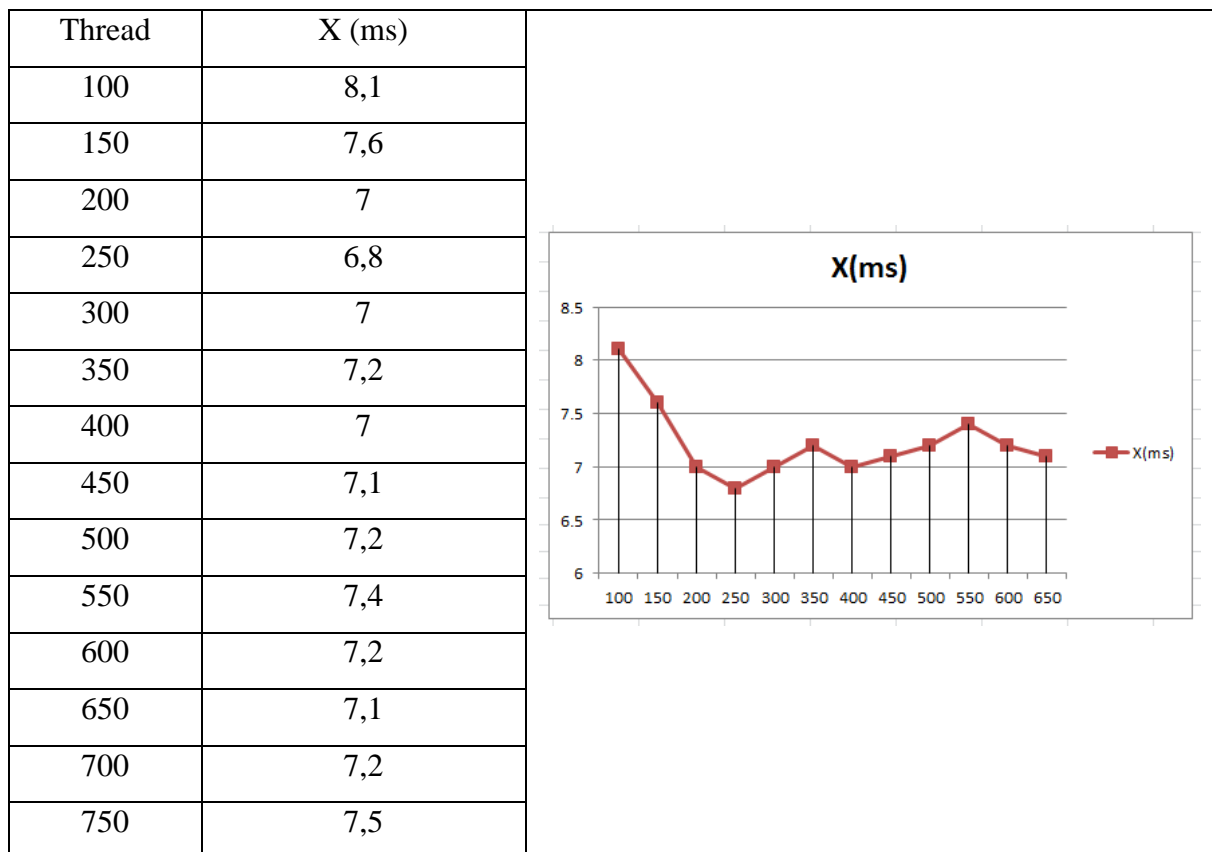
4.3 Pengujian Indikator

Pemilihan pada indikator yang terkait untuk keperluan pengujian merupakan tahapan penting karena dengan adanya pemilihan indikator tersebut maka pengujian dapat dilakukan dengan lebih valid dan terukur. Indikator pengujian yang akan di analisis adalah beban dari *thread*.

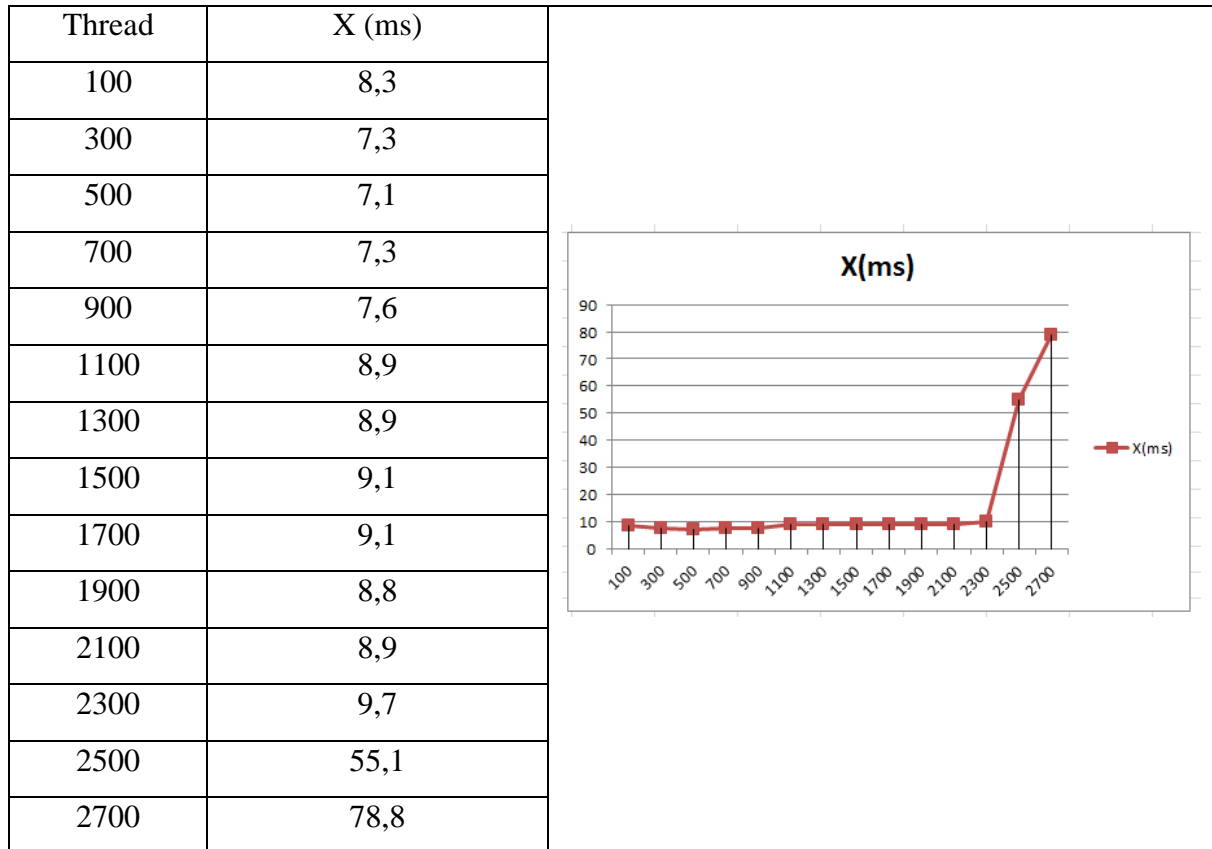
4.3.1 Pemilihan Beban Thread

Pemilihan beban *thread* menjadi penting untuk menguji kinerja suatu sistem. Dalam menentukan beban uji tiap iterasi untuk mendapatkan representasi grafik dengan perubahan yang konsisten. Pada konteks kali ini akan ditentukan menggunakan *thread* dengan jumlah awal 100 dengan kelipatan 50 dan interval waktu sebanyak 60 detik. Tabel pemilihan beban *thread* dapat dilihat pada tabel 4.1 berikut.

Tabel 4.1 Tabel ujicoba interval 50

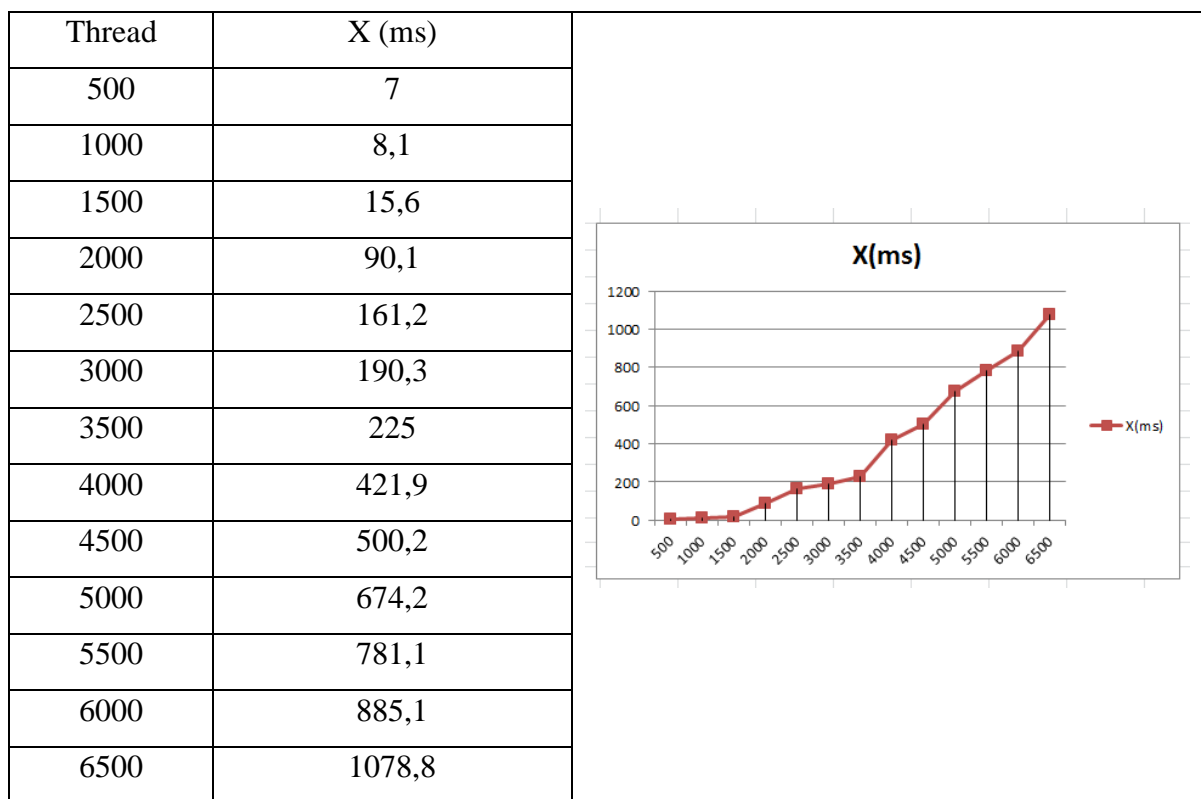


Tabel 4.2 Tabel ujicoba interval 200



Pada pemilihan tersebut, yang merupakan sebuah asumsi yang dibuat oleh penulis, penggunaan interval yang tepat memperlihatkan dari kestabilan grafik yang dapat terlihat pada hasil gambaran tersebut. Pada konteks kali ini akan ditentukan menggunakan *thread* dengan jumlah awal 100 dengan kelipatan 200 dan interval waktu sebanyak 60 detik. Oleh sebab itu, pemilihan beban *thread* akan diberikan lebih tinggi supaya hasilnya dapat terlihat.

Tabel 4.3 Tabel ujicoba interval 500



Pada pemilihan tersebut, yang merupakan sebuah asumsi yang dibuat oleh penulis, penggunaan interval yang tepat memperlihatkan dari kestabilan grafik yang sudah akan dibuat. Pada konteks kali ini akan ditentukan menggunakan *thread* dengan jumlah awal 100 dengan kelipatan 200 dan interval waktu sebanyak 60 detik.

Penggunaan grafik pada interval antara 200 – 500 merupakan pilihan yang tepat karena dalam implementasinya dengan citra grafik akan terlihat lebih konsisten.

4.4 Pengujian Performa PHP

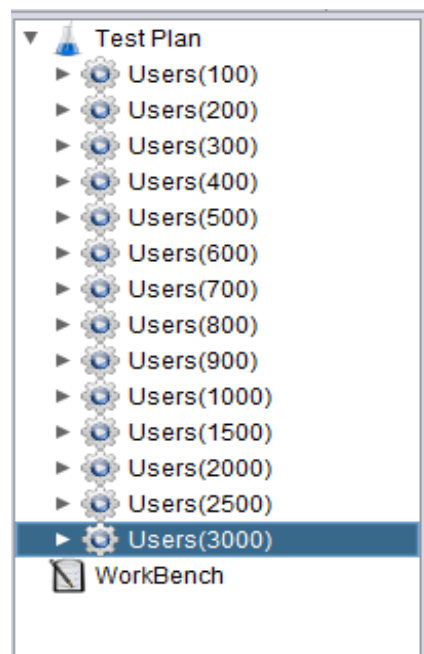
Pengujian performa pada PHP dikerjakan menggunakan sebuah skema yang ditentukan. Skema tersebut adalah *test plan*. *Test plan* berperan untuk memberikan alur pengujian supaya data yang akan didapat menjadi lebih konsisten. Pada tahapan kali ini, skenario pengujian akan dibuat supaya dapat memberikan uniformitas pada faktor pengujian tiap subyek uji yang ada. Pengujian pada PHP akan menyesuaikan *environment* pada teknologi itu.

4.4.1 Test Plan PHP

Pembuatan *test plan* pada PHP berfungsi untuk membuat sebuah skenario yang akan diujikan. Skenario pengujian berfungsi untuk memberikan gambaran data serta dapat membantu penelitian untuk mengumpulkan data. Pembuatan *test plan* meliputi beberapa konfigurasi yaitu:

- a. Konfigurasi jumlah *thread*.
- b. Konfigurasi HTTP Request.
- c. Konfigurasi *listener*

Pada pemilihan jumlah *thread*, akan dibagi menjadi dua kenaikan. Kenaikan 100 dan kenaikan 500. Kenaikan 100 berfungsi untuk melihat seberapa kecepatan maksimal dari sebuah *service* PHP yang berjalan. Sedangkan *thread* pada kenaikan 500 akan menguji tingkat kejenuhan dari *service* tersebut. Setiap pengujian pada perpindahan *thread* akan dilakukan berdasarkan interval waktu. Setiap *loop* akan dikenakan durasi sebanyak 660 detik untuk melakukan eksekusi terhadap jumlah *thread* yang sudah ditentukan. Interval waktu yang digunakan untuk setiap perpindahan *thread* dilakukan menggunakan *timer* secara *manual* untuk menambah konsistensi pada pengujian. Jumlah *thread* dapat dilihat dari gambar berikut.



Gambar 4.9 Susunan Thread Group PHP

Pada konfigurasi selanjutnya akan dibuat sebuah skema dalam menerima *request* yang terdapat pada *service* dari PHP itu sendiri. Hal yang perlu diperhatikan dalam membuat sebuah skema *request* adalah HTTP Header, Web Server, dan HTTP Request. Konfigurasi dapat dilihat pada gambar berikut.

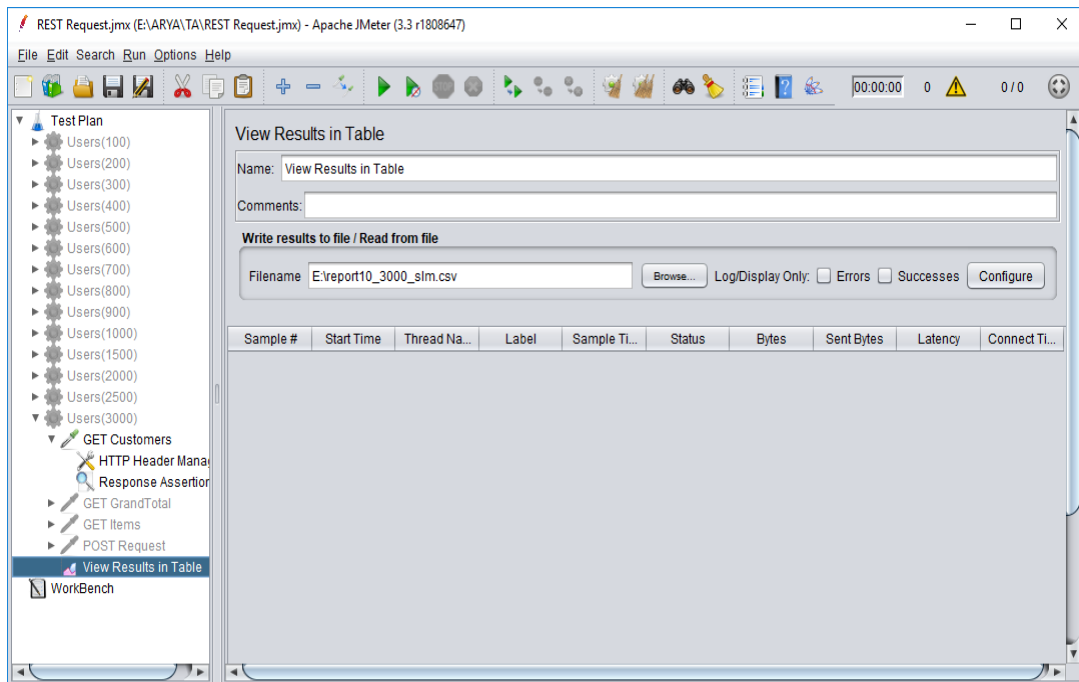
The image shows the 'HTTP Request' configuration window in JMeter. It is divided into several sections:

- Name:** GET Customers
- Comments:** (empty text area)
- Web Server:** Protocol [http]: http, Server Name or IP: 192.168.56.101, Port Number: (empty)
- HTTP Request:** Method: GET, Path: /public/api/json/customers, Content encoding: (empty)
- Options:** Redirect Automatically, Follow Redirects, Use KeepAlive, Use multipart/form-data for POST, Browser-compatible headers
- Parameters:** A table with columns 'Name', 'Value', 'Encode?', and 'Include Equals?'. It is currently empty.
- Buttons:** Detail, Add, Add from Clipboard, Delete, Up, Down

Gambar 4.10 Request PHP

Konfigurasi tersebut akan berjalan pada sisi *client* dari laptop. JMeter akan menerima sebuah request berdasarkan HTTP Header yaitu “application/json”. Penggunaan *header* JSON adalah sebagai uniformitas dalam menerima *sampling* data dan juga JSON sebagai kumpulan data yang sering digunakan dalam pertukaran data pada arsitektur N-Tier.

Alur pengujian terakhir adalah memberikan sebuah penerima data atau *listener*. Pemilihan *listener* untuk pengujian kali ini menggunakan tabel. Pemilihan *listener* tabel berfungsi untuk memberikan hasil evaluasi data yang lengkap dan praktis serta tidak membutuhkan banyak *resource* dalam pengujian.

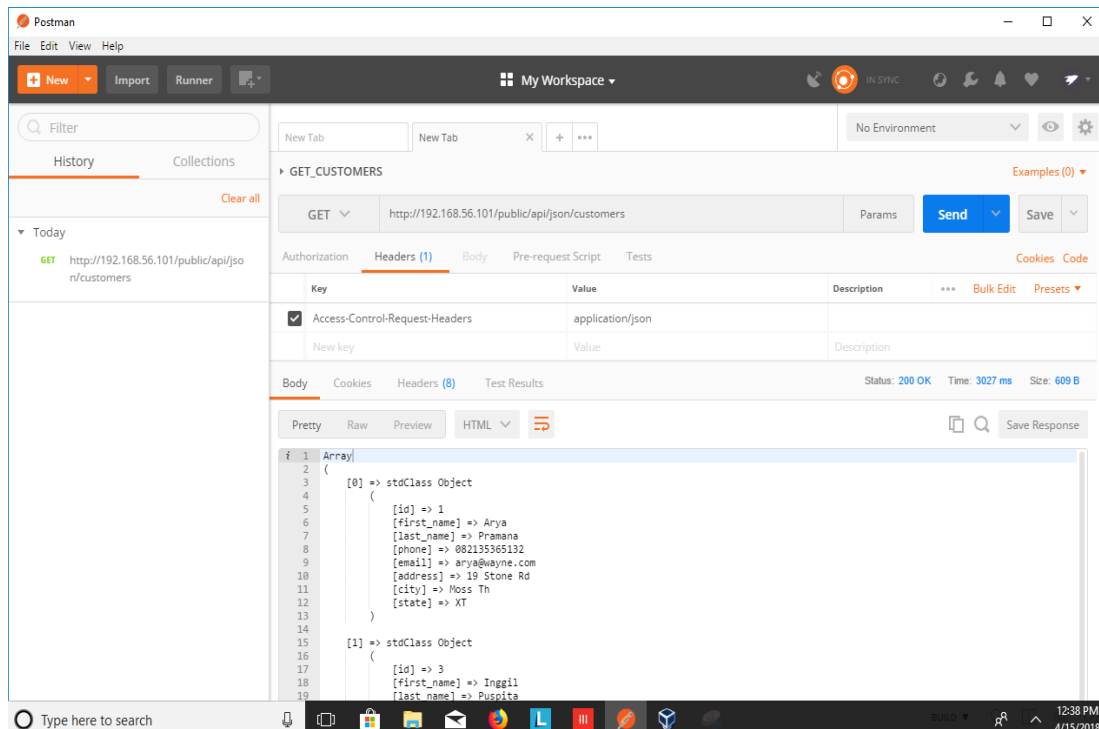


Gambar 4.11 Tabel data PHP

Penggunaan *listener* tabel tersebut juga memungkinkan untuk melihat parameter menjadi lebih lengkap dan mudah dibaca. Fasilitas lainnya adalah dapat menuliskan laporan pengujian pada *loop* tersebut secara independen.

4.4.2 Service PHP

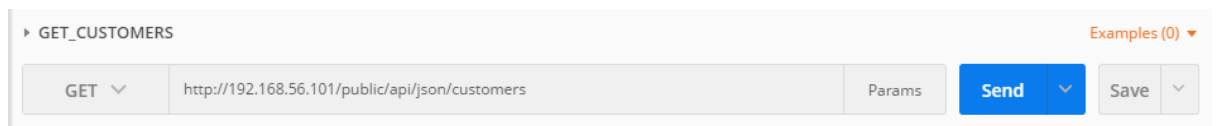
Layanan PHP yang sudah dibangun akan dilakukan sebuah pengujian *request* yang akan ditetapkan. Pengujian tersebut berfungsi untuk menentukan apakah data yang ada di dalam *service* tersebut memiliki format yang sama atau tidak serta url yang dapat berfungsi dengan baik.



Gambar 4.12 HTTP Request PHP

Aplikasi yang digunakan untuk mengecek jalannya *service* dari PHP menggunakan Postman. Postman merupakan sebuah aplikasi yang memiliki tujuan khusus untuk memberikan kemudahan *programmer* untuk mengembangkan layanan berbasis web. Postman memberikan fasilitas untuk dapat menangkap beberapa protokol *default* dari REST (GET, DELETE, PUT, POST).

Pada pembacaan *request* tersebut akan menggunakan respon GET dikarenakan *service* yang sudah dibuat. Respon GET tersebut akan menampilkan data yang sudah ada dalam layanan PHP tersebut.



Gambar 4.13 Konfigurasi URL dan tipe request

Alamat IP tersebut akan menyesuaikan dari alamat *service* yang sudah dibuat. IP akan diakses melewati koneksi host-only yang sudah dibangun pada arsitektur pengujian itu.


```

Array ( [0] => stdClass Object ( [id] => 1 [first_name] => Arya [last_name] => Pramana [phone] => 082135365132 [email] => arya@wayne.com [address] => 19 Stone Rd [city] => Moss Th [state] => XT ) [1]
=> stdClass Object ( [id] => 3 [first_name] => Inggil [last_name] => Puspita [phone] => 083243434767 [email] => pusпита@wayne.com [address] => 37 Abbey Rd [city] => Yogyakarta [state] => DIY ) [2] =>
stdClass Object ( [id] => 4 [first_name] => Alfred [last_name] => Pennyworth [phone] => 08754632251 [email] => alfred@wayne.com [address] => 12 Scott Kill Rd [city] => Gotham [state] => BT ) [3] =>
stdClass Object ( [id] => 5 [first_name] => Barbara [last_name] => Gordon [phone] => 08136523741 [email] => bgordon@wayne.com [address] => 14 Weirfold Kidrter Rd [city] => Gotham [state] => BT )

```

Gambar 4.14 Respon data JSON dari PHP

Data yang menjadi *output* dari PHP yaitu menggunakan JSON. Penggunaan respon dari JSON dikarenakan respon tersebut merupakan sebuah standar pertukaran data yang sering digunakan dalam pengembangan layanan berbasis web.

4.5 Pengujian Performa JAVA

Pengujian performa pada JAVA dikerjakan menggunakan sebuah skema yang ditentukan. Pada *test plan* yang sudah dibuat pada tahapan pengujian PHP, penggunaan *test plan* pada JAVA tidak akan jauh berbeda. Penggunaan beberapa protokol yang sama akan terlihat pada komponen yang menjadi parameter di pengujian yang akan dilakukan.

4.5.1 Test Plan JAVA

Pembuatan *test plan* pada JAVA berfungsi untuk membuat sebuah skenario yang akan diujikan. Pembuatan *test plan* meliputi beberapa konfigurasi yaitu:

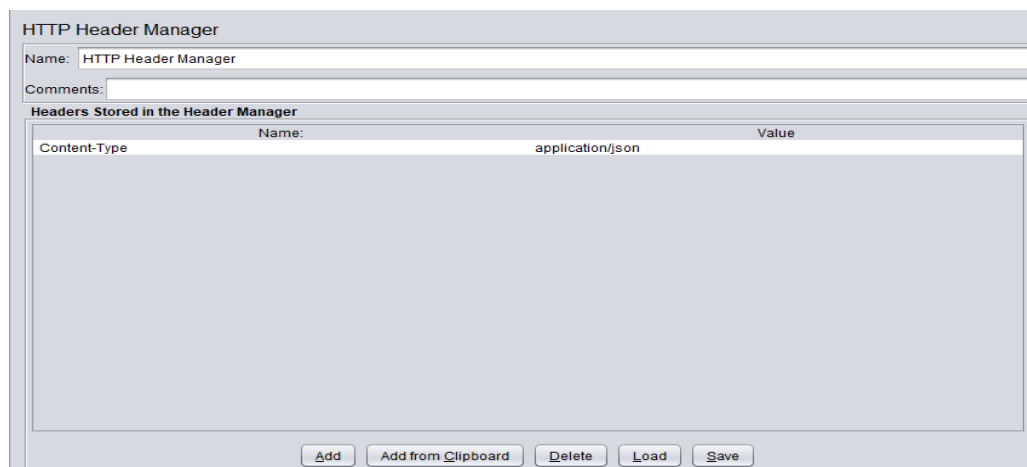
- a. Konfigurasi jumlah *thread*.
- b. Konfigurasi HTTP Request.
- c. Konfigurasi *listener*

Pada pemilihan jumlah *thread*, skema dalam penentuan jumlah kenaikan pada JAVA sendiri akan sama seperti yang sudah dilakukan oleh PHP sebelumnya sehingga tidak ada perbedaan yang terjadi dalam pembuatan *test plan*. Hal ini dilakukan supaya dapat menjaga independensi dalam melakukan pengujian performa. Perbedaan yang ada pada layanan JAVA sendiri adalah pada format aplikasi tersebut. Format *packaging* pada JAVA yaitu *.jar* sendiri memiliki sebuah *embedded server* yang berjalan independen. Pada *thread* kenaikan 500 akan menguji tingkat kejenuhan dari *service* tersebut. Penggunaan kenaikan jumlah *thread* yang tinggi akan berdampak pada hasil pengujian untuk memberikan perbedaan terhadap kapasitas dari *service* itu sendiri. Penggunaan interval waktu untuk pergantian waktu *loop* juga tetap sama dan telah dibantu oleh *timer* yang telah disediakan untuk menambah konsistensi dari hasil pengujian. Jumlah *thread* dapat dilihat dari gambar 4.15 berikut.



Gambar 4.15 Susunan Thread Group JAVA

Pada konfigurasi selanjutnya akan dibuat sebuah skema dalam menerima *request* yang terdapat pada *service* dari JAVA itu sendiri. Pada teknologi JAVA sendiri, penggunaan port sedikit berbeda dikarenakan penggunaan port pada *native* menggunakan port 8080. Pada PHP sendiri, port yang digunakan adalah 8000 yang merupakan komponen *default* dari konfigurasi yang telah ditetapkan pada *server* masing-masing teknologi. Pada respon data yang akan digunakan menggunakan ‘application/json’.



Gambar 4.16 Header data JSON dari JAVA

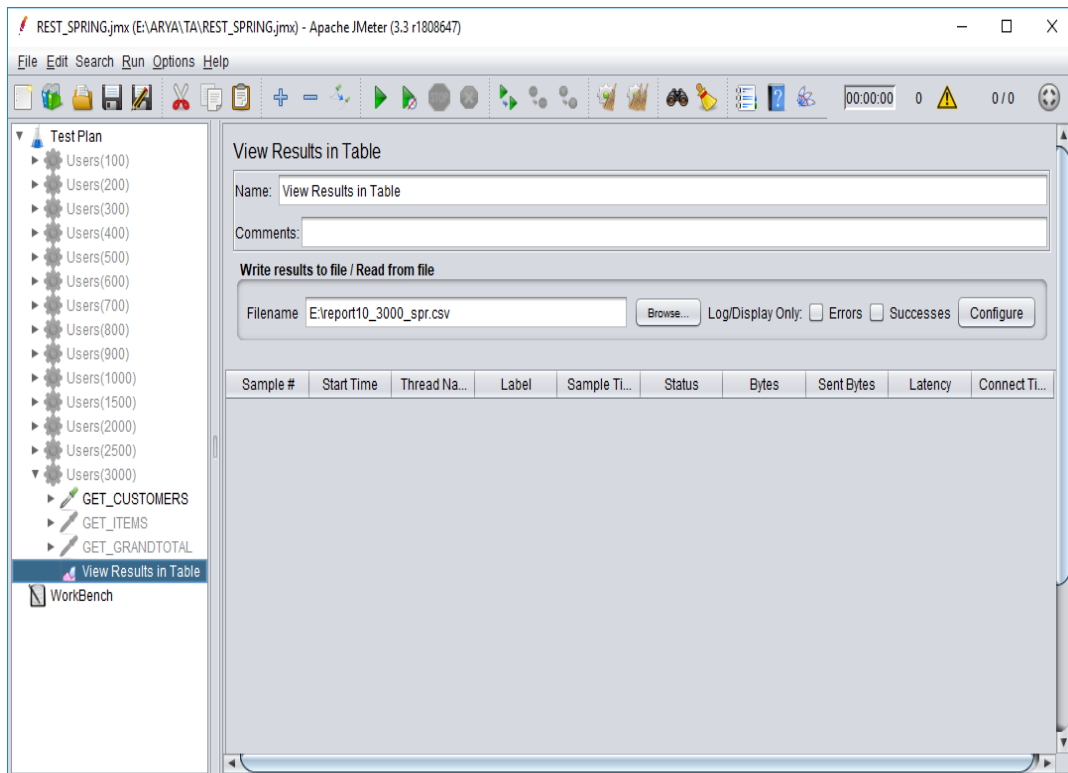
Pemilihan respon JSON pada bagian *request* memungkinkan untuk membaca respon dengan format JSON yang sudah ditetapkan. Penggunaan format JSON menjadi umum karena dalam pengembangan layanan berbasis web, format data JSON merupakan yang populer.

The screenshot shows the 'HTTP Request' tool interface. At the top, the 'Name' field is set to 'GET_CUSTOMERS'. Below it, there are tabs for 'Basic' and 'Advanced'. The 'Web Server' section is configured with 'Protocol [http]: http', 'Server Name or IP: 192.168.56.101', and 'Port Number: 8080'. The 'HTTP Request' section shows 'Method: GET' and 'Path: /springbootapp/api/customers'. There are several checkboxes: 'Redirect Automatically' (unchecked), 'Follow Redirects' (checked), 'Use KeepAlive' (checked), 'Use multipart/form-data for POST' (unchecked), and 'Browser-compatible headers' (unchecked). Below this, there are tabs for 'Parameters', 'Body Data', and 'Files Upload'. The 'Parameters' tab is active, showing a table with columns 'Name', 'Value', 'Encode?', and 'Include Equals?'. The table is currently empty. At the bottom, there are buttons for 'Detail', 'Add', 'Add from Clipboard', 'Delete', 'Up', and 'Down'.

Gambar 4.17 Request JAVA

Pemberian konfigurasi *request* akan berpengaruh pada hasil pengujian pada layanan tersebut. Penggunaan port 8080 yang merupakan *default* dari *server* JAVA sendiri. Metode pengiriman *request* berupa GET dikarenakan *request* tersebut adalah mengambil respon berupa data dengan format JSON.

Pembuatan *listener* pada JAVA akan menggunakan sebuah tabel. Penggunaan tabel sendiri itu merupakan pilihan untuk dapat menekan *resource* yang digunakan saat melakukan pengujian. Tabel pada pengujian tersebut juga akan memberikan data yang mudah diamati dan mudah diolah. Data yang akan diperlihatkan dalam *listener* itu berupa *sample time* yang merupakan parameter yang akan diujikan dalam melakukan komparasi terhadap teknologi lainnya (untuk kasus ini PHP). Pada model *listener* tabel juga dapat secara langsung menghitung angka rata-rata dari sebuah *loop* yang sudah ditetapkan. Hal ini akan sangat membantu dalam menyelesaikan permasalahan pengujian dan pengumpulan data secara efisien. Simulasi tabel data dapat dilihat pada gambar 4.17 berikut.



Gambar 4.18 Tabel data JAVA

Pada tabel tersebut juga dapat menuliskan sebuah laporan pada *loop* secara independen sehingga dapat memudahkan dalam mengelompokkan data yang akan diolah untuk membuat perbandingan dalam pengujian. Data yang sudah didapat dari *output* sebuah tabel tersebut akan dilakukan pengolahan untuk mencari rata-rata pada tiap iterasi yang sudah dibuat untuk mengetahui konsistensi performa dari teknologi yang sedang diujikan.

4.5.2 Service JAVA

Sebuah layanan JAVA yang sudah dibangun akan melibatkan beberapa *request* yang terkait. Request yang digunakan adalah sebuah request GET untuk menampilkan data yang sudah dibuat untuk contoh dalam pengujian ini. Untuk inialisasi pada *service* kali ini cukup berbeda dengan PHP dikarenakan pada format *package* dari JAVA itu sendiri yaitu *.jar*. pada penanganan *.jar* yaitu seperti mengeksekusi aplikasi seperti biasa. Menggunakan perintah yang ada pada terminal *server* memungkinkan untuk melakukan inisiasi pada *service* yang akan dijalankan.

Respon yang dihasilkan dari layanan JAVA tersebut merupakan konfigurasi dasar yang ada pada Spring Boot sendiri. Pada awal konfigurasi, untuk membuat aplikasi dengan Spring Boot diharuskan untuk mengunduh berbagai macam *library*. Salah satunya adalah untuk menampilkan respon data tersebut menggunakan jackson.

4.6 Perbandingan

Pada bagian perbandingan, akan diperlihatkan sebuah hasil yang telah dieksekusi dari *test plan* sudah dibuat. Hasil yang didapat berdasarkan dari tabel *listener* yang kemudian diolah dalam sebuah aplikasi *spreadsheet* untuk menghasilkan tabel dan grafik komparasi.

4.6.1 Tabel Perbandingan

Tabel berikut merupakan sebuah perbandingan dari hasil perancangan *test plan* yang sudah dikerjakan. Tabel ini akan membandingkan tiap hasil *request* yang sudah dikerjakan pada tiap *loop* yang sudah diolah dengan aplikasi pengolah angka.

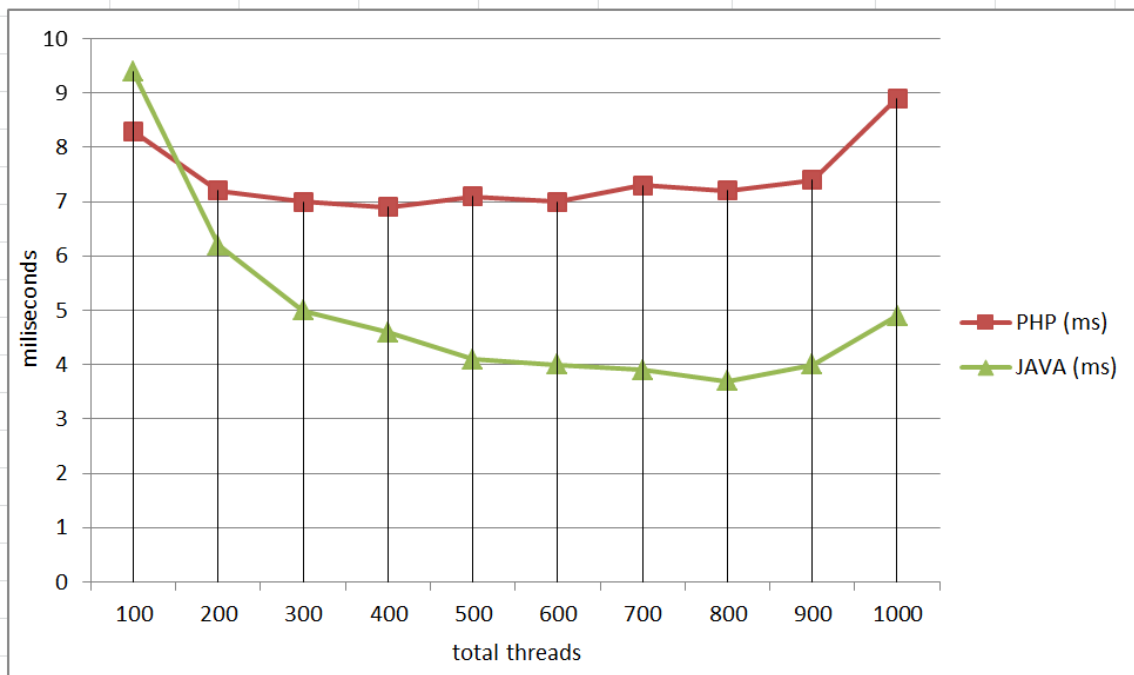
Tabel 4.4 Tabel komparasi

Thread	PHP (ms)	JAVA (ms)
100	8,3	9,4
200	7,2	6,2
300	7	5
400	6,9	4,6
500	7,1	4,1
600	7	4
700	7,3	3,9
800	7,2	3,7
900	7,4	4
1000	8,9	4,9
1500	9,2	4,9
2000	73,5	6
2500	149,2	17
3000	161,6	21,3

Data tabel diatas merupakan sebuah representasi dari hasil pengujian performa berdasarkan pada *test plan* yang sudah dibuat. Tiap *thread* tersebut menjalankan *request* sebanyak 10 kali iterasi. Dari hasil iterasi tersebut kemudian diambil nilai rata-ratanya yang kemudian dapat merepresentasikan kecepatan tiap *thread* yang sudah ditentukan. Pada hasil tersebut dapat diketahui bahwa performa dari JAVA sendiri dapat melampaui PHP pada hampir semua jumlah *thread*. Pada PHP sendiri cenderung memiliki kecepatan akses yang stabil pada angka *thread* dibawah 1000. Untuk eksekusi *thread* yang berada pada jumlah dibawah 1000, pada dasarnya PHP dan JAVA tidak memiliki perbedaan yang signifikan. PHP juga dapat mencapai kecepatan yang stabil dengan lebih cepat daripada JAVA. Pada teknologi JAVA, kecepatan respon tersebut didapatkan secara gradual sehingga untuk mencapai kecepatan maksimum lebih membutuhkan waktu daripada PHP. JAVA sendiri dapat menekan kestabilan kecepatan lebih baik daripada PHP diatas *thread* yang berjumlah lebih dari 1000.

4.6.2 Grafik Perbandingan

Untuk memperjelas dalam melakukan komparasi dapat dibuat sebuah perbandingan dalam bentuk grafik komparasi. Grafik komparasi pada penelitian ini akan dibagi menjadi dua. Pembagian tersebut untuk mempermudah dalam memahami data serta pengujian yang dilakukan memiliki kenaikan *thread* yang berbeda. Grafik tersebut kemudian dapat menunjukkan bagaimana kenaikan dalam tiap *thread* tersebut dan juga bagaimana mengkomparasikannya dengan variabel dengan lebih mudah. Dalam visualisasi grafik tersebut dapat dikatakan bahwa untuk teknologi JAVA dan PHP dapat mencapai kestabilan yang sama pada saat berjalan pada *thread* rendah (di bawah 1000). Seperti yang disebutkan tadi, *thread* dengan jumlah rendah akan menguji sebuah layanan tersebut untuk mencapai kecepatan eksekusi yang maksimal. Pada dasarnya, PHP mencapai kecepatan maksimal yang lebih cepat dan condong untuk lebih stabil pada *thread* rendah tersebut. Pada JAVA, *thread* rendah tersebut kemudian digunakan untuk mencapai kecepatan maksimalnya dan cenderung untuk selalu lebih cepat pada tiap iterasi dan interval pergantian jumlah *thread*. Pada saat kedua dari *thread* pengujian tersebut mencapai *thread* tinggi (di atas 1000 dan kenaikan menjadi 500), maka akan terjadi sebuah perbedaan. JAVA cenderung lebih stabil dan tidak ada lonjakan drastis saat berjalan pada *thread* tersebut. Berbeda dengan PHP yang menjadi tidak stabil. Detail ilustrasi pada grafik pengujian dapat dilihat pada gambar 4.21 berikut.

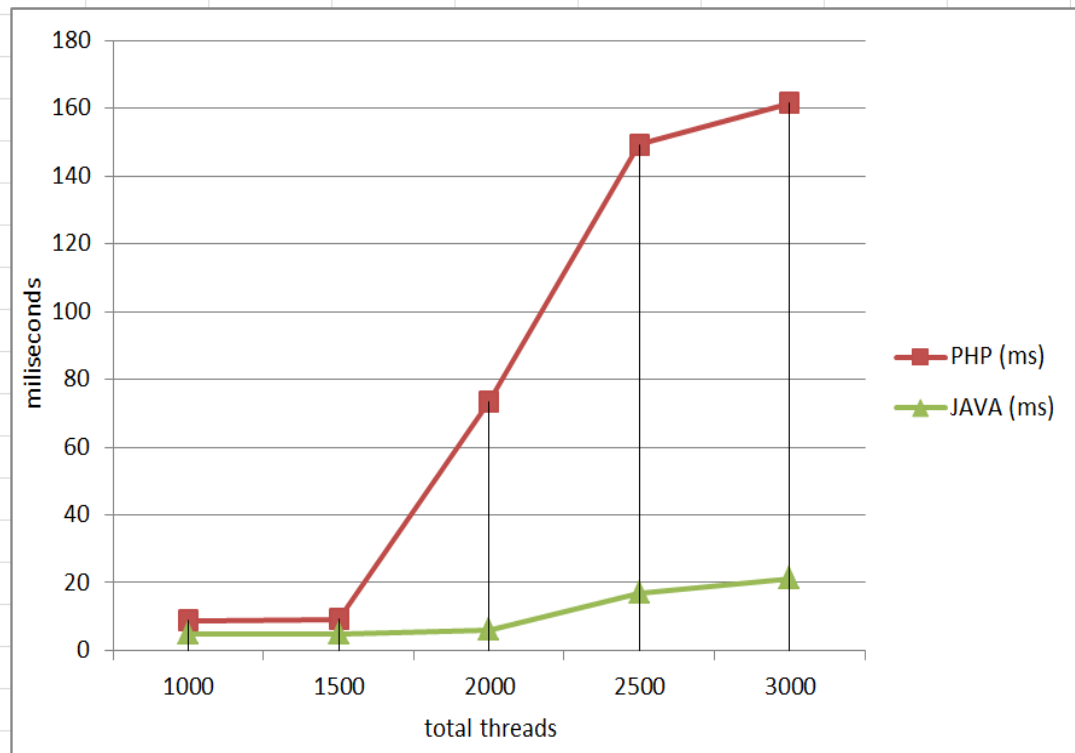


Gambar 4.21 Grafik komparasi (1)

Grafik di atas pada sumbu mendatar atau horizontal menjelaskan jumlah *thread* yang sedang berjalan pada tiap variabel uji. Pada sisi vertikal merupakan sebuah hasil dari rata-rata waktu respon dalam *milliseconds* yang dapat dicapai oleh masing masing variabel uji.

Pada fase pengujian yang berada pada tahap pertama, yaitu kenaikan tiap 100 *threads*, dapat dilihat bahwa kecepatan yang bisa didapat dari JAVA memiliki kecepatan respon yang lebih cepat dibandingkan PHP. Walaupun pada *thread* ke-100 JAVA seperti sedikit terlambat dengan mengawali uji performa tersebut. Performa pada PHP juga termasuk baik karena dapat mencapai kecepatan respon yang maksimal lebih cepat dar JAVA, walaupun sebenarnya PHP masih kalah cepat dalam mencapai kecepatan maksimal. JAVA dalam memperoleh kestabilan performa cenderung cukup lama. JAVA selalu mencatatkan respon kecepatan yang selalu relatif lebih cepat dibandingkan dengan iterasi sebelumnya sampai mencapai titik respon tercepatnya pada *thread* ke-800.

Pengujian fase kedua merupakan pengujian dengan kenaikan yang lebih signifikan pada tiap *thread* yaitu sejumlah 500. Fase ini akan memperlihatkan tingkat kejenuhan dari suatu layanan pada masing-masing teknologi sehingga dapat terlihat mana yang dapat memberikan performa maksimal pada tingkatan *users* yang tinggi. Detail grafik dapat dilihat pada gambar 4.22 berikut.



Gambar 4.22 Grafik komparasi (2)

Pengujian fase kedua ini kemudian dapat memperlihatkan perbedaan performa yang dapat ditunjukkan dalam pengujian ini. Masing-masing respon dari PHP maupun JAVA mengalami peningkatan waktu respon saat diberikan kenaikan jumlah *thread* sejumlah 500. Perbedaan yang signifikan terjadi pada PHP saat memasuki jumlah *thread* di angka 2000. Terjadi lonjakan hampir 8 kali lipat waktu respon dibanding sebelumnya. Berbeda dengan JAVA yang masih relatif tetap stabil dengan menjaga waktu respon dibawah 20 ms. PHP sendiri pada *thread* selanjutnya juga memberikan hasil yang terbilang cukup tinggi kontras dengan hasil yang didapat dari JAVA. JAVA sendiri dalam pengujian kali ini bisa menjaga kestabilan waktu respon dengan menjaganya tetap rendah. Peningkatan *thread* pada JAVA dapat diatasi dengan baik sehingga kenaikan pada jumlah *thread* tidak memiliki dampak signifikan dengan waktu respon yang didapat.

4.6.3 Kendala

Kendala yang dialami saat melakukan eksekusi terhadap *test plan* tersebut adalah mengenai sebuah perbedaan waktu interval pada setiap pergantian *loop* yang kadang tidak konsisten. Peneliti berpendapat bahwa waktu interval yang tidak konsisten akan sedikit berpengaruh pada hasil uji yang terlibat. Namun, untuk mengatasinya, penulis menggunakan

timer yang dipasang secara *manual* untuk menghitung interval untuk menambah konsistensi. Ada pula batasan perangkat keras yang menjadikan pengujian kali ini menjadi cukup terbatas. Beberapa contoh dari keterbatasan pengujian pada perangkat keras adalah sistem yang digunakan masih menggunakan prosesor *dual-core* sehingga ada limitasi performa pada tiap teknologi yang terlibat.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

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

- a. Teknologi pada JAVA memiliki tingkat kejenuhan respon lebih rendah daripada PHP ditunjukkan dari hasil pengujian ketika memasuki fase kedua.
- b. Pada fase pertama pengujian, kedua teknologi tidak memiliki perbedaan yang cukup signifikan dan cenderung stabil walaupun pada JAVA waktu respon mengalami penurunan.

5.2 Saran

Penelitian kali ini masih ada beberapa kekurangan yang bisa diperbaiki untuk kedepannya yaitu adanya beberapa delay atau interval pada perpindahan *thread* yang kadang tidak konsisten dikarenakan pada *timer* yang digunakan masih menggunakan jenis *manual*. Hal ini dapat diperbaiki dengan cara memberikan skenario pengujian secara realistis dan memberikan *timer* sendiri pada tiap *thread*. Serta dalam beberapa kesempatan dalam pengujian masih terdapat beberapa limitasi koneksi untuk perangkat keras yang mungkin juga dapat berpengaruh pada hasil pengujian secara nyata. Penggunaan *hardware* yang lebih canggih serta koneksi yang dibuat secara fisik dapat membantu memberikan hasil yang lebih akurat.

DAFTAR PUSTAKA

- Baxi, S. (2016). N-Tier _ N-Layer Architecture _ Swapnil Baxi _ Pulse _ LinkedIn.
- BCM. (2018). N-Tier Architecture: Tier 2, Tier 3, and Multi-Tier Explained – BMC Blogs. Retrieved from <https://www.bmc.com/blogs/n-tier-architecture-tier-2-tier-3-and-multi-tier-explained/>
- CATechnologies. (2015). API Strategy and Architecture a Coordinated Approach, 23.
- Dirgahayu, T., & Utama, M. I. (2004). COMPARISON ON EFFICIENCY AND SPEED OF 2-TIER AND 3-TIER OLTP SYSTEMS, 2(2), 23–31.
- Fredrich, T. (2013). RESTful Service Best Practices: recommendations for creating web services, 1–34. Retrieved from https://github.com/tfredrich/RestApiTutorial.com/raw/master/media/RESTful_Best_Practices-v1_2.pdf
- Hass, A. M. J. (2008). *Guide to advanced software testing*. <https://doi.org/10.1007/978-1-84800-044-5>
- Martin, M., & Amir, R. (2015). Computer Architecture, 501. Retrieved from https://www.cis.upenn.edu/~milom/cis501-Fall11/lectures/00_intro.pdf
- Murawski, G., Keck, P., & Schnaible, S. (2014). Evaluation of Load Testing Tools, (193).
- Patton, R. (2001). *Software Testing. Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki*. Retrieved from <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0>
- Suominen, T. (2017). PERFORMANCE TESTING REST APIS Information Technology.
- TechTarget. (2015). What is middleware? - Definition from WhatIs.com. TechTarget. Retrieved from <http://searchsoa.techtarget.com/definition/middleware>