Rancang Bangun *Web Service - JSON* Menggunakan Kombinasi *Spring* dan *MyBatis Framework* dalam lingkungan *Java Platform*

Benie Ilman[1], Arief Ginanjar[2]
Fakultas Teknik, Program Studi Informatika Universitas Langlangbuana

Abstrak

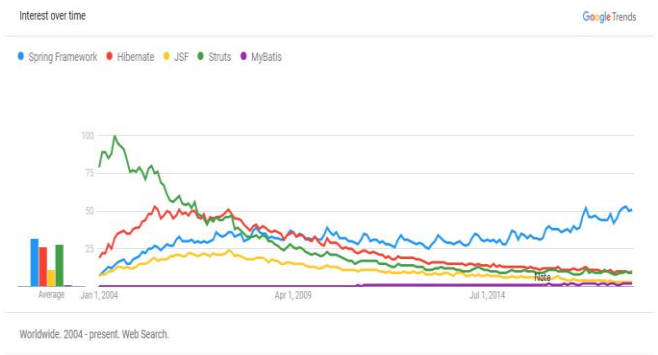
Java Platform merupakan salah satu teknologi ICT yang cukup paripurna, hal tersebut disebabkan karena setiap cabang ilmu pemrograman dapat diimplementasikan menggunakan bahasa pemrograman java, baik itu dalam aplikasi *web*, aplikasi *desktop, mobile programming, console app* serta *smart card*. Ketika fokus pembahasan terhadap *java web application* terdapat satu *framework* yang sudah populer sejak tahun 2004 yaitu *Spring Framework*, ditambah dengan kemampuan *Spring Framework* yang dapat digabungkan dengan berbagai *framework* lain seperti *MyBatis Framework* serta beberapa *framework* lain. Penelitian yang dilakukan untuk mengetahui rancang bangun aplikasi serta kemampuan aplikasi *web service* yang dibangun menggunakan bahasa pemrograman *java* dengan *Spring Framework*, terutama jika dikombinasikan dengan *MyBatis Framework* ditambah dengan kemampuan tambahan teknologi lain yaitu lapisan *data cache* yang terdapat dalam *MyBatis*. Pengujian *web service Spring framework* menggunakan *custom script code* yang dilakukan secara berulang-ulang dengan rentang waktu tertentu.

Kata kunci : *Spring Framework, Javascript Object Notation, JSON, MyBatis Framework, Cache Engine, Web Service*.

I. PENDAHULUAN

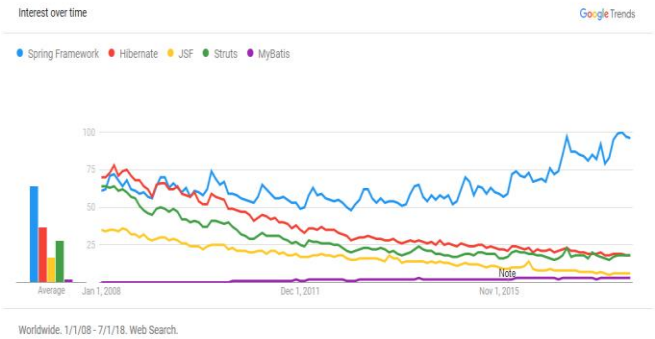
A. Latar Belakang

Spring Framework merupakan salah satu *framework* pertama yang digunakan dalam pembuatan aplikasi *web* dalam lingkungan *java platform* pada masa awal tahun 2000an. Seiring waktu muncul *framework - framework* lain yang menawarkan keunggulan yang beragam, namun tingkat popularitas *Spring Framework* masih bertahan dan digunakan hingga tahun 2018. Hal tersebut diketahui dari infografis yang menggambarkan tingkat minat para *programmer java* terhadap *framework* populer di dunia antara tahun 2004 hingga 2018.

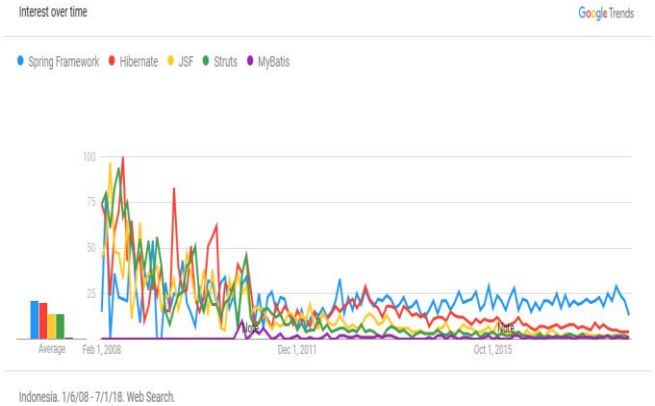


Gambar 1. Popularitas *java framework* di dunia dari 2004 hingga 2018 (Sumber: Google Trend).

Dari Gambar 1 terlihat beberapa *framework* yang cukup populer digunakan oleh para *programmer* untuk diimplementasikan dalam pekerjaan yang mereka lakukan diantaranya *Spring Framework, Hibernate, JSF, Struts* dan *MyBatis*, *framework* inilah yang paling populer diantara *framework* yang ada dan digunakan oleh para *programmer java*.



Gambar 2. Popularitas *java framework* di dunia dari 2008 hingga 2018 (Sumber: Google Trend).



Gambar 3. Popularitas *java framework* di Indonesia dari 2008 hingga 2018 (Sumber: Google Trend).

Pada gambar 3 terlihat peminat *framework* oleh para *programmer java* di Indonesia antara periode 2008 hingga 2010 masih fluktuatif, hal tersebut menandakan *framework* yang digunakan dalam pekerjaan aplikasi masih dalam tahap coba-coba namun setelah tahun 2011 mulai terlihat kecenderungan yang signifikan para *programmer* sudah menemukan kenyamanan dan manfaat dari setiap *framework* populer tersebut.

Dalam implementasi penggunaan *Spring Framework* di pengembangan aplikasi berbasis java, seringkali dikombinasikan dengan framework lain diantaranya adalah; *Java Database Connection* (JDBC), *Object Relational Mapping* dengan merk dagang *Hibernate*, *Java Query Expose Connection* dengan merk dagang *MyBatis*, serta penggunaan *cache engine* yang berfungsi untuk mengoptimalkan akses data terhadap *database* dengan tujuan untuk meningkatkan kecepatan akses data aplikasi.

B. Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari penelitian ini adalah mengketahui perancangan dengan hasil kehandalan yang maksimal dari setiap spesimen yang tercantum dalam Tabel 1 yang akan diuji dengan serangkaian pengujian menggunakan data dummy dengan jumlah 100.000 baris data serta uji performa *data loading* dengan cara *increment load* terhadap *url* data *json*.

TABEL I.
SPESIMEN *FRAMEWORK* YANG AKAN DIUJI

Spesimen	*Framework* Utama	*Bridging Framework*	*Cache Framework*
1	*Spring Framework*	*MyBatis*	-
2	*Spring Framework*	*MyBatis*	*LRU MyBatis*

Dengan pengujian tersebut diharapkan dapat diketahui kehandalan setiap spesimen ketika harus menghadapi uji performa menggunakan *load looping*. Sehingga diharapkan dapat menjadi sumber referensi dalam pengambilan keputusan ketika harus memilih kolaborasi *Spring Framework* yang diterapkan dalam pengembangan aplikasi berbasis java..

C. Batasan Masalah

Penelitian ini hanya berkonsentrasi terhadap *data loading url* dan tidak melakukan pengujian terhadap operasi *insert, update dan delete url json* dengan spesifikasi teknis sebagai berikut:

- Menggunakan sistem basis data *MariaDB* versi 10.1.19 yang diletakkan terpisah Sistem Operasi *Microsoft Windows* 7 *Genuine* 64 *bit* didalam *Oracle VM Virtual Box* versi 5.2.6.
- Menggunakan koneksi jaringan *local host - guest virtualbox connection*.
- Menggunakan Sistem Operasi *Microsoft Windows* 7 *Genuine* 64 *bit* untuk *host environment*.
- Menggunakan *Netbean* 8.2 sebagai *Integrated Development Editor*.
- Menggunakan *Spring Framework* versi 4.0.1 sebagai *MVC Framework*.
- Menggunakan *Spring-json taglib* sebagai *library* untuk menghasilkan *output json*.
- Menggunakan *application container Apache Tomcat* versi 9.0.12 sebagai *media deployment application*.
- Menggunakan *Java Virtual Machine* versi 1.8.0.162 64 *bit* dengan *setting core JVM default* tanpa merubah konfigurasi bawaan sebagai *platform environment*.

II. METODE PENELITIAN

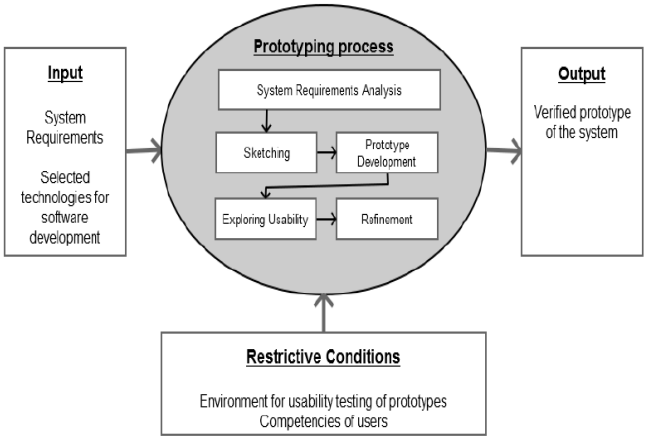
A. Pelaksanaan Penelitian

Metodologi yang digunakan pada pembuatan laporan penelitian dengan menggunakan metode penelitian kuantitatif *prototype* serta ditambah proses uji kemampuan performa *framework* dengan perulangan proses serta pendekatan sistem menekankan terhadap elemen dan komponen [2][6][7], adapun urutan proses dilakukan sebagai berikut:

- Studi Literatur, Mempelajari sumber-sumber pustaka yang dapat dijadikan referensi. Sumber-sumber pustaka dapat berupa buku, *paper* atau halaman *web* yang membahas tentang *Spring Framework, Ibatis Framework* dan *Cache Engine*.
- Analisis, Menjelaskan bagaimana melakukan analisis terhadap arsitektur dan teknik pemrograman terhadap masing-masing *framework* serta bagaimana teknik mengkombinasikan setiap *framework* tersebut.
- Perancangan Perangkat Lunak, Melakukan perancangan terhadap perangkat lunak dikombinasikan diantara *framework* yang akan dibangun berdasarkan hasil yang diperoleh dari analisis. Perancangan *scenario* memenuhi syarat pengujian yang akan dilakukan.
- Implementasi Perangkat Lunak, yang akan dikembangkan berdasarkan hasil yang diperoleh dari perancangan. Implementasi menghasilkan produk perangkat lunak yang berisi kombinasi *framework* seperti yang telah ditentukan dalam *scenario* uji.
- Pengujian dan Evaluasi, terhadap produk perangkat lunak yang telah dibangun dan kemudian melakukan evaluasi kinerja dari setiap *scenario* pengujian yang dilakukan.
- Iteratif merupakan proses pengujian terhadap spesimen perangkat lunak yang akan diteliti ketika proses bisnis belum memenuhi spesifikasi maka lakukan perancangan dan implementasi ulang terhadap perangkat lunak.
- Pengujian Kinerja Aplikasi, proses pengujian aplikasi terhadap tekanan dengan cara *loop loading url* yang dilakukan berkali-kali terhadap spesimen *framework* yang telah dibangun menjadi aplikasi menggunakan metode *url looping script* dengan *script java*.

B. Evolutionary Prototyping Model

Tahapan proses yang telah dijelaskan dalam bagian pelaksanaan penelitian yang merujuk kepada tahapan evolusioner *prototyping model* maka proses tersebut dapat diilustrasikan seperti yang terlihat pada Gambar 5. Proses evolusioner *prototyping* yang terdiri dari proses sebagai berikut *input*, *prototyping process*, dan *output*, tetapi selama pelaksanaan proses juga harus dibatasi dengan syarat; setiap fungsi yang dibangun harus memenuhi syarat yang sesuai dengan *system requirement*, dengan kompetensi orang yang ada dalam *prototyping* memenuhi syarat *system requirement* [8].

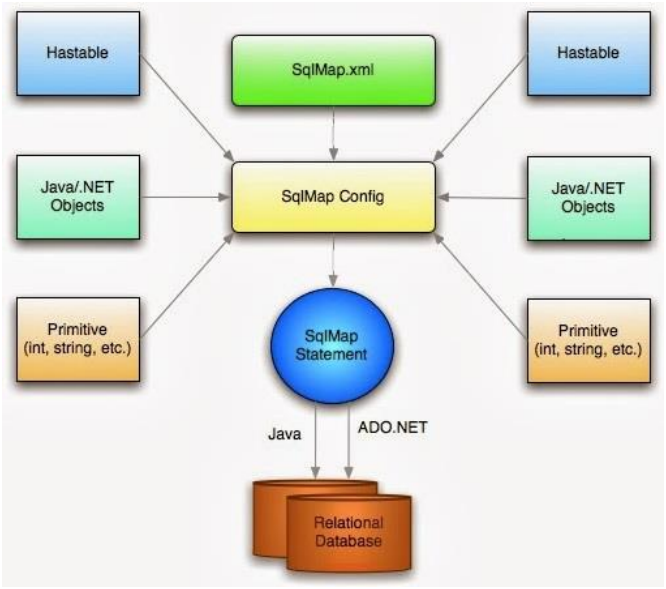


Gambar 5 Metodologi Evolutionary Prototyping Model [8].

C. Arsitektur Spesimen

Dengan pendekatan prinsip dari *Object Oriented Programming* serta pendekatan *Model, View dan Controller* MVC, abstraksi memungkinkan *programmer* untuk mengembangkan pemikiran yang rumit tanpa harus fokus terhadap komponen, sedangkan enkapsulasi memungkinkan kita untuk fokus terhadap kemampuan *software* tanpa harus berpikir detil terhadap kerumitan proses yang terjadi. [4][5].

Dengan menggabungkan arsitektur *Spring Framework* dan *MyBatis Framework* menjadi sebuah sistem yang saling menutupi kekurangan setiap *framework*, diharapkan dengan arsitektur yang dibangun dan dengan pengujian yang dilakukan dapat ditemukan titik optimal dalam penggabungan beberapa *framework* [1][3].

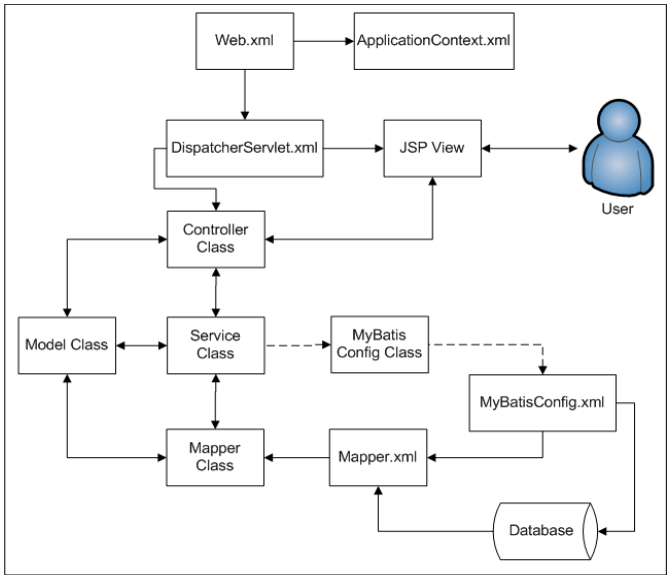


Gambar 6. Arsitektur Sistem *MyBatis Framework* [3].

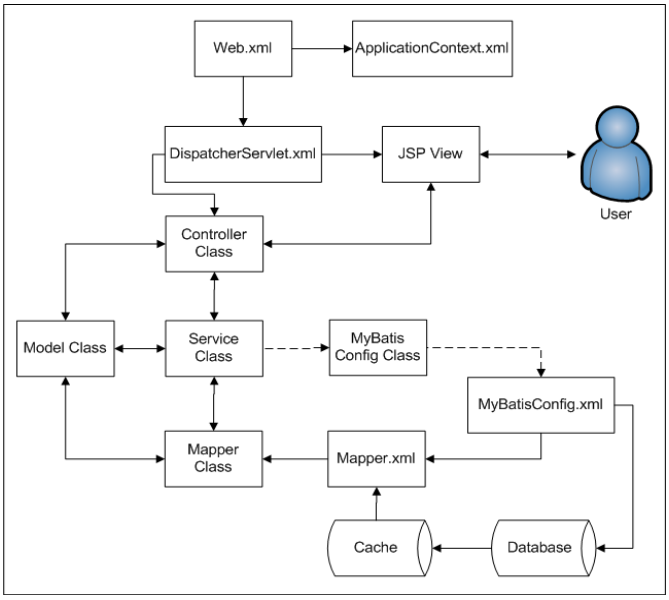
III. PROSES PENELITIAN

A. Perancangan Spesimen

Untuk membangun *specimen* penggabungan *framework Spring* dan *MyBatis* maka peneliti perlu merancang sebuah arsitektur aplikasi sederhana untuk dijadikan sebagai media pengujian performa aplikasi yang dibangun dari *framework Spring* dan *MyBatis*, dengan perancangan sebagai berikut:



Gambar 7. Arsitektur Aplikasi *Spring* dan *MyBatis Framework*



Gambar 8. Arsitektur Aplikasi *Spring* dan *MyBatis Framework* dengan Cache

B. Konfigurasi Spesimen

Pengujian yang dilakukan yaitu; melibatkan beberapa lapisan sistem yaitu, lapisan *database*, lapisan *application container*, lapisan *logic programming* serta lapisan *script* pengujian. Kemudian dalam setiap lapisan terdapat konfigurasi yang dapat digunakan untuk setiap lapisan dapat saling berinteraksi dengan lapisan lain.

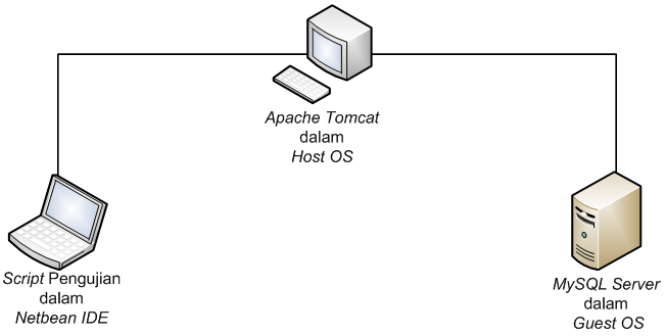
TABEL II.
IMPLEMENTASI KONFIGURASI DI SETIAP *LAYER*

No	Lapisan	Teknologi	Konfigurasi
1	*Database*	*MySQL*	*Default*
2	*Application Container*	*Apache Tomcat*	*Default*
3	*Logic Programming*	*Spring + MyBatis Framework*	Spesifik setiap spesimen
4	*Script* Pengujian	*Java Programming*	Spesifik setiap spesimen

Alur pengujian setiap spesimen yang disimpan dalam server virtual box yang terinstall didalam guest OS, kemudian Apache Tomcat diinstall dalam host OS, selanjutnya script pengujian dijalankan menggunakan Netbean IDE versi 8.2 dengan ilustrasi dapat terlihat pada Gambar 8.

Setiap *framework* yang terlihat dalam Tabel 1 membutuhkan juga beberapa *library* lain sebagai *support* terhadap *framework* utama dalam spesimen penelitian, dengan perincian *library* dapat dilihat dalam Tabel 3.

Pada kolom spesimen merupakan spesimen pengujian untuk *Spring Framework* dengan versi 4.0.1 dikombinasikan dengan *Java Database Connection* (JDBC) membutuhkan *library* lain yaitu *JSP Standard Tag Library* (JSTL) yang mempunyai fungsi untuk menterjemahkan data yang berasal dari lingkungan *java class* terhadap lingkungan *java web*, kemudian *MySQL*-JDBC yang berfungsi sebagai *Java Database Connection* (JDBC) khusus untuk *database server* dengan merk *MySQL*, kemudian dibantu juga dengan *JSON Taglib* yaitu *tools converter* yang berfungi untuk merubah data berasal dari lingkungan JSTL menjadi *Javascript Object Notation.*



Gambar 9. Alur pengujian *performance spring framework* terhadap beberapa *database bridging layer*.

TABEL III.
LIBRARY PELENGKAP DARI SETIAP SPESIMEN

Library	**Spesimen**	
	1	**2**
JSTL	√	√
MySQL JDBC	√	√
JSON Taglib	√	√
Ehcache	×	√

Ehcache merupakan teknologi yang berbasis *hardware* atau *software* yang digunakan untuk dijadikan sebagai penyimpanan data sementara yang digunakan untuk membantu tingkat performa dari aplikasi atau sistem. Penerapan teknologi *cache* sudah cukup banyak digunakan oleh para pengembang aplikasi untuk meningkatkan kecepatan akses terhadap *view* data aplikasi, setiap *framework* mempunyai arsitektur berbeda metode dalam implementasi *cache* terhadap data yang dikelolanya, hal tersebut cukup menarik untuk dijadikan bahan penelitian selanjutnya.

C. Pengumpulan Data

Pengumpulan data yang dilakukan yaitu berupa menjalankan spesimen aplikasi yang terbangun dari komposisi *framework* seperti yang terlihat dalam Tabel 1 dan terinstall dalam lingkungan *application container* dengan nama produk *Apache Tomcat* versi 9.0.12 kemudian dijalankan menggunakan *test script* dalam lingkungan *Netbean* 8.2 seperti yang terlihat pada Gambar 10.

```
public static void main(String[] args) throws IOException, JSONException {
    // TODO code application logic here
    long secondStart = System.currentTimeMillis();
    JSONArray jsonArray = readJsonArrayFromUrl("http://localhost:8080/Research-Spring-Jdbc/json.htm");
    for (int i = 0; i < jsonArray.length(); i++) {
        //get the JSON Object
        JSONObject obj = jsonArray.getJSONObject(i);
        Integer id = obj.getInt("id");
        String nama = obj.getString("nama");
        Integer anak = obj.getInt("anak");
        System.out.println("id = "+id+", nama = "+nama+", anak = "+anak);
    }
    long secondFinish = System.currentTimeMillis();
    long elapseTime = secondFinish - secondStart;
    System.out.println(" second "+elapseTime);
}
```

Gambar 10. Baris perintah pengujian *performance spring framework* terhadap beberapa *database bridging layer* menggunakan *java programming*.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.agien.springmvc.mappers.ResearchMapper" >
    <resultMap id="result" type="research">
        <result property="id" column="id"/>
        <result property="nama" column="nama"/>
        <result property="anak" column="anak"/>
    </resultMap>

    <select id="findAllResearch" flushCache="false" useCache="true"
            timeout="10000" resultMap="result" >
        SELECT ID as id, NAMA as nama, ANAK as anak FROM research_union
    </select>
</mapper>
```

Gambar 11. *Configuration script MyBatis mapper* untuk menghubungkan *class interface* dengan baris perintah *query* basisdata.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.agien.springmvc.mappers.ResearchMapper" >
    <cache
        eviction="LRU"
        flushInterval="360000"
        size="2048"
        readOnly="true" />

    <resultMap id="result" type="research">
        <result property="id" column="id"/>
        <result property="nama" column="nama"/>
        <result property="anak" column="anak"/>
    </resultMap>

    <select id="findAllResearch" flushCache="false" useCache="true"
            timeout="10000" resultMap="result" >
        SELECT ID as id, NAMA as nama, ANAK as anak FROM research_union
    </select>
</mapper>
```

Gambar 12. *Configuration script MyBatis mapper* untuk menghubungkan *class interface* dengan baris perintah *query* basisdata dengan *cache configuration*.

D. Hasil Penelitian.

Selama pelaksanaan pengujian spesimen *Spring Framework* yang dikombinasikan terhadap *Database Bridging Framework* ditemukan bahwa ada kecenderungan positif yang terjadi antara kombinasi *MyBatis Framework* dengan *L2 Cache*. Dari pengujian yang telah dilakukan berdasarkan *scenario* pengujian dan ditemukan kondisi bahwa spesimen satu dan dua dapat berfungsi dengan baik, kemudian spsimen dua mempunyai catatan waktu reaksi cukup tinggi.

Dari data yang tercantum dalam gambar yang diwakili diagram garis seperti yang terlihat diatas peneliti akan mengambil data pokok yaitu; kecepatan load rata - rata setiap *scenario vs* setiap spesimen, kecepatan *load* paling rendah setiap *scenario vs* setiap spesimen, kecepatan *load* paling tinggi setiap *scenario vs* setiap spesimen, kecepatan *load* bernilai sama yang paling sering muncul di setiap *scenario vs* setiap spesimen, serta kecepatan load nilai tengah di setiap *scenario vs* setiap spesimen. Seluruh data tersebut tersajikan dalam Tabel 4.1

TABEL IV.
HASIL UJI AKSES SKENARIO SATU DAN DUA SETIAP SPESIMEN

		Kecepatan *Load* (mili detik)			
		Average	*Slow*	*Fast*	*Modus*
1 Hit 100	Spc 1	2,189.44	3,869.00	1,827.00	1,871.00
	Spc 2	1,667.76	3,995.00	954.00	1,972.00
1 Hit 500	Spc 1	2,823.50	4,526.00	1,831.00	2,930.00
	Spc 2	1,518.25	4,044.00	987.00	1,021.00
2 Hit 100	Spc 1	2,153.55	3,846.00	1,785.00	1,810.00
	Spc 2	1,703.65	4,015.00	984.00	2,013.00
2 Hit 500	Spc 1	2,787.61	4,486.00	1,774.00	2,871.00
	Spc 2	1,554.14	4,094.00	1,006.00	1,058.00

TABEL V.
HASIL UJI AKSES SCENARIO TIGA DAN EMPAT SETIAP SPESIMEN

		Kecepatan *Load* (mili detik)			
		Average	*Slow*	*Fast*	*Modus*
3 Hit 100	Spc 4	2,328.62	3,085.00	1,928.00	2,007.00
	Spc 5	1,587.41	2,522.00	1,421.00	1,468.00
3 Hit 500	Spc 4	2,369.24	5,043.00	1,864.00	1,916.00
	Spc 5	1,370.43	4,025.00	987.00	1,019.00
4 Hit 100	Spc 1	2,295.79	3,969.00	1,898.00	2,010.00
	Spc 2	1,566.00	2,873.00	1,398.00	1,443.00

		Kecepatan *Load* (mili detik)			
		Average	*Slow*	*Fast*	*Modus*
4 Hit 500	Spc 1	2,414.12	4,999.00	1,882.00	1,974.00
	Spc 2	1,408.47	3,100.00	1,004.00	1,048.00

TABEL VI.
HASIL UJI AKSES SCENARIO LIMA DAN ENAM SETIAP SPESIMEN

		Kecepatan *Load* (mili detik)			
		Average	*Slow*	*Fast*	*Modus*
5 Hit 100	Spc 1	2,273.23	4,868.00	1,837.00	1,972.00
	Spc 2	1,537.26	2,501.00	1,336.00	1,399.00
5 Hit 500	Spc 1	2,450.01	5,031.00	1,912.00	1,980.00
	Spc 2	1,444.36	3,139.00	1,014.00	1,088.00
6 Hit 100	Spc 1	2,397.68	4,053.00	1,903.00	1,959.00
	Spc 2	1,586.87	3,489.00	1,438.00	1,470.00
6 Hit 500	Spc 1	2,483.48	5,041.00	1,932.00	2,040.00
	Spc 2	1,410.89	3,113.00	963.00	1,058.00

D. Pembahasan Data Hasil Pengujian

Kecepatan data akses terhadap masing-masing spesimen dibandingkan dengan masing-masing *scenario* berdasarkan data dari tabel 4.1, diketahui bahwa:

- Spesimen satu dan dua dari setiap *scenario* pengujian mempunyai kecepatan akses yang cukup signifikan tinggi baik dalam kecepatan rata-rata dalam setiap *looping access* dan kecepatan tertinggi yang dapat dicapai di beberapa *looping access*.
- Untuk rata-rata kecepatan tertinggi dapat dilihat pada *scenario* tiga *hit* 500 pada spesimen dua dengan kecepatan 1.370,45 mili detik.
- Sedangkan untuk kecepatan tertinggi akses dapat dilihat pada *scenario* satu *hit* 100 pada spesimen dua dengan kecepatan 954 mili detik.
- Untuk rata - rata kecepatan akses paling rendah dapat dilihat pada *scenario* satu *hit* 500 pada spesimen 1 dengan kecepatan 2.823 mili detik.
- Sedangkan untuk kecepatan akses paling rendah dengan yaitu *scenario* empat *hit* 500 pada spesimen 1 dengan kecepatan 4.999 mili detik.

Dari lima pembahasan kesimpulan sementara yang berasal dari Tabel 4, Tabel 5 dan Tabel 6, peneliti menarik kesimpulan bahwa spesimen dua unggul dalam kecepatan akses dibandingkan dengan spesimen satu.

IV. KESIMPULAN

Dengan penelitian tentang pengujian performa *Spring Framework* terhadap *MyBatis Framework* dengan fokus hanya terhadap *data loading url* yang telah dilakukan maka terdapat beberapa hasil yang dapat diketahui yaitu;

- Perpaduan *Spring Framework* 4.0.1 dan *MyBatis Framework* 3.4.2 menghasilkan performa yang sangat tinggi,
- Kemudian ketika perpaduan konfigurasi *Spring Framework* 4.0.1 dan *MyBatis Framework* 3.4.2 ditambah dengan *Cache Engine* performa semakin meningkat.

DAFTAR PUSTAKA

[1] Begin, Clinton., 2007, Ibatis in Action, Manning Publication, New York, USA.

[2] Jogiyanto, H.M., 2008, Analisis dan Perancangan Sistem: Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis, Andi Offset., Yogyakarta.

[3] Johnson, Rod., 2005, Professional Java Development with the Spring Framework, Wiley Publishing., Indianapolis, USA.

[4] Kalelkar, Medha., Churi, Prathamesh., Kalelkar, Deepa., 2014, Implementation of Model-View-Controller Architecture Pattern for Business Intelligence Architecture., International Journal of Computer Application., New York, USA.

[5] Kendal, Simon., 2009, Object Oriented Programming using Java., Ventus Publishing Aps, Frederiksberg, Denmark.

[6] Sugiyono., 2012, Metode Penelitian Kuantitatif Kualitatif dan R&D., Alfabeta, Bandung.

[7] Roger, S. Pressman, Ph.D., 2012, Rekayasa Perangkat Lunak, Edisi 7 : Buku 1 Andi Offset, Yogyakarta.

[8] Nacheva, Radka, 2017, Prototyping Approach in User Interface Development, Conference on Innovative Teaching Methods, University of Economics Varna, Bulgaria.