# PENERAPAN *RESOURCE ORIENTED ARCHITECTURE* UNTUK PENDISTRIBUSIAN DATA MENGGUNAKAN *RESTful APIs* BERBASIS *MULTIPLATFORM*

**SKRIPSI**

**Diajukan untuk memenuhi salah satu syarat dalam menempuh Ujian Sarjana Komputer (S.Kom)**


**Oleh:**

**Yanuar Nurcahyo**

**NPM: 1512018**


**JENJANG STRATA 1 (S1)**

**PROGRAM STUDI TEKNIK INFORMATIKA**


**SEKOLAH TINGGI ILMU KOMPUTER BINANIAGA**

**BOGOR**


**2016**

## LEMBAR PERSETUJUAN SKRIPSI

Judul : PENERAPAN *RESOURCE ORIENTED ARCHITECTURE* UNTUK PENDISITIBUSIAN DATA MENGGUNAKAN *RESTful APIs* BERBASIS *MULTIPLATFORM*

Peneliti/Penulis : Yanuar Nurcahyo, NPM: 1512018

Karya tulis Tugas Akhir ini telah diperiksa dan disetujui sebagai karya tulis ilmiah penelitian.

Bogor,

Disetujui Oleh:

| Pembimbing I | Pembimbing II |
|---|---|
| Dahlia Widhyaestoeti, S.Kom, M.Kom | Anggra Triawan, M.Kom |
| NIP: 11.120.1102 | NIP: 11.102.1003 |

Ketua Program Studi
Teknik Informatika

Irmayansyah, M.Kom
NIP:11.120.0404

Wakil Ketua Bidang Akademik

Irmayansyah, M.Kom
NIP: 11.120.0404

## PERNYATAAN KEASLIAN PENGEMBANGAN

Sejauh yang peneliti ketahui, karya tulis yang dibuat ini benar-benar merupakan hasil karya penulis. Perihal unsur dan juga sumber yang terdapat dalam karya tulis merupakan bagian dari bahan penelitian yang digunakan untuk mendukung penulisan penelitian

.

Demikian surat pernyataan ini saya buat sebenar-benarnya, tanpa ada paksaan dari pihak manapun. Apabila di kemudian hari terjadi gangguan perihal hak-hak kepemilikan sumber data dan melanggar hukum, saya siap menerima sanksi sesuai yang berlaku.

Bogor, Desember 2016

Yanuar Nurcahyo

## RIWAYAT PENULIS

Yanuar Nurcahyo lahir di Jakarta pada tanggal 26 Januari 1995. Ia tinggal di Jakarta hanya 5 bulan dan pindah ke daerah kabupaten Bojonggede kota Bogor.Pada tahun 2006 lulus Sekolah Dasar di SDN Bojonggede 3, tahun 2009 menyelesaikan Sekolah Menengah Pertama di SMP Al-Basyariah, kemudian menamatkan Sekolah Menengah Kejuruan di SMK Tri Dharma 2 Bogor, jurusan RPL (Rekayasa Perangkat Lunak) pada tahun 2012. Tahun 2012 melanjutkan pendidikan di S1 Jurusan Teknik Informatika di STIKOM BINANIAGA Bogor.

Sampai saat ini, ia masih melanjutkan kuliahnya dengan membuat karya tulis ilmiah. Karya ilmiah tersebut adalah “Resource Oriented Architecture Untuk Pendistribusian Data Menggunakan RESTful APIs Berbasis Multiplatform”. Disela-sela aktivitasnya, ia juga bekerja sebagai freelancer dibidang web programming sejak tahun 2014 sampai saat ini.

# Penerapan Resource Oriented Architecture Untuk Pendistribusian Data Menggunakan RESTful APIs Berbasis Multiplatform

Yanuar Nurcahyo, NPM: 1512018

Teknik Informatika, Stikom Binaniaga, Bogor

Email: yanuarxnurcahyo@gmail.com

## ABSTRAK

Penelitian ini dilatar belakangi oleh pendistribusian data journal pada perpustakaan universitas. Dimana pendistribusian data journal ini belum dapat didistribusikan kepada Platform pada masa kini dimana platform pada saat ini berbagai macam jenis platform. Maksud pada penelitian ini adalah menerapakan metode Resource Oriented Architecture dengan membuat web service RESTful APIs yang akan diterapkan pada beberapa platform. Penelitian ini bertujuan untuk membuktikan bahwa metode Resource Oriented Architecture dapat menjadi solusi pendistribusian data yang berbasis Multiplatform serta mencari platform yang pantas digunakan sebagai client web service ini. Data yang digunakan penelitian ini adalah data journal dimana data tersebut didapatkan dari beberapa website universitas ternama di indonesia salah satunya ITB dan UI. Dengan data tersebut akan didistribusikan kepada beberapa platform diantaranya adalah Python, PHP dan Android. Pada pembuatan RESTful API ini menggunakan bahasa pemrograman NodeJS v4.4.7 dan MySQL sebagai databasenya. Untuk mengetahui platformmana yang lebih unggul dalam menggunakan web service ini yang menggunakan HTTP Request sebagai pengaksesannya, peneiti menggunakan rumus kecapatan rata – rata waktu dimana rumus tersebut untuk mengukur seberapa cepat masing – masing platform mengakses RESTful API. Maka dari hasil penelitian adalah Python mendapatkan hasil yang lebih cepat dari pada platform lainnya dimana hasil tersebut adalah PHP (0.0240), Android (0.0188) dan Python (0.0067).

*Keyword: Resource Oriented Architecture, RESTful API, HTTP, NodeJS, Python, PHP, Android*

# KATA PENGANTAR

Puji dan syukur dipanjatkan kehadirat Allah SWT karena atas berkat rahmat dan hidayah–Nya penyusunan skripsi berjudul *"Resource Oriented Architecture* Untuk Pendistribusian Data Menggunakan RESTful APIs Berbasis Multiplatform*"* dapat diselesaikan tepat pada waktunya. Meskipun banyak hambatan yang dialami dalam proses pengerjaanya, namun Alhamdulillah berhasil diselesaikan.

Penulisah skripsi ini diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana pada jurusan Teknik Informatika di STIKOM Binaniaga Bogor. Permasalah yang diangkat adalah kurang lengkapnya fasilitas diperpustakaan saat ini mengenai data-data journal diberbagai universitas lokal maupun internasional mengakibatkan mahasiswa sulit mencari referensi journal.

Dalam penyusunan dan penulisan skripsi ini tidak terlepas dari bantuan, bimbingan serta dukungan dari berbagai pihak. Penulis menyadari bahwa dalam menyusun laporan ini masih jauh dari kesempurnaan.

Akhir kata, penulis sampaikan terima kasih kepada semua pihak yang telah berperan serta dalam penyusunan skripsi ini dari awal sampai akhir. Semoga Allah SWT senantiasa memberkahi segala usaha kita. Aamiin.

Bogor, Desember 2016

Yanuar Nurcahyo

# UCAPAN TERIMA KASIH

Puji dan syukur dipanjatkan kehadirat Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya penyusunan skripsi ini dapat diselesaikan dengan baik. Pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua dan keluarga saya yang telah mendukung hingga ke tahap terakhir kuliah sarjana.
2. Ibu Dahlia Widhyaestoeti, M.Komselaku dosen pembimbing I dan Bapak Anggra Triawan, M.Kom selaku pembimbing II yang telah bersedia meluangkan waktu, tempat, dan tenaga untuk memberikan segala bimbingan, arahan, saran, dan dukungan moril kepada penyusun sehingga dapat menyelesaikan penyusunan skripsi dengan baik dan tepat waktu.
3. Ibu Irmayansyah, M.Kom selaku Wakil Ketua Bidang Akademik yang telah bersedia meluangkan waktu, tempat, dan tenaga untuk memberikan bimbingan, arahan, saran terkait isi dan tulisan yang terdapat dalam karya tulis ini untuk perbaikan dalam penyusunan karya tulis selanjutnya.
4. Seluruh dosen yang telah memberikan ilmu pembelajaran sehari-hari serta dukungan moril selama penyusun menempuh pendidikan di STIKOM Binaniaga sehingga penyusun dapat menyelesaikan pendidikan dengan baik dan tepat waktu.
5. Seluruh rekan-rekan di lingkungan STIKOM Binaniaga yang telah bekerja sama dengan baik selama penyusun menempuh pendidikan di STIKOM Binaniaga sampai dengan menyelesaikan penyusunan skripsi ini.
6. Technology masa kini seperti Google, Stackoverflow dan Microsoft Word yang sudah memudahkan penyusunan skripsi sehingga penulis terbantu dalam menyelesaikannya

Selain nama-nama yang disebutkan diatas, juga diucapkan terima kasih atas partisipasinya dalam penyusunan skripsi ini. Semoga segala dukungan dan partisipasi yang telah diberikan dapat bermanfaat dan dibalas oleh Tuhan Yang Maha Esa. Aamiin…

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I

## PENDAHULUAN

## A. LATAR BELAKANG

Dalam peradaban baru dunia global, kemajuan teknologi dan informasi menjadi infrastruktur penopang bergeraknya globalisasi dan ekonomi neoliberal. Melalui teknologi informasi, pemegang modal raksasa di sektor keuangan dan industri dengan mudah memindahkan modalnya dari satu negara kenegara yang lain. Saat ini Indonesia telah satu tahun memasuki pasar bebas dan bersiap memasuki *AEC* (*Asean Economic Community*) di tahun 2015, perkembangan industri di Indonesia telah melonjak sangat tinggi, termasuk perkembangan di sektor pendidikan pun meningkat pesat.

Perpustakaan pada sektor universitas pada saat ini sangat berperan penting untuk kemajuan bangsa disuatu negara terutama untuk sebuah penelitian. Penting bagi perpustakaan universitas menyediakan refrensi journal-journal nasional maupun internasional secara lengkap. Artinya, untuk mencapai keberhasilan peneliti merupakan aspek yang paling menentukan bagi pengetahuan umum. Salah satu kemampuan dasar yang harus dimiliki oleh seorang peneliti adalah kemampuan profesional di bidang tertentu. Kemajuan teknologi dibidang perpustakaan juga sangat berkembang seiringinya zaman digital saat ini bahkan berbagai platform digunakan untuk membangun sistem tersebut.

Masih banyak pada zaman yang sudah serba digital perpustakaan universitas tidak memanfaatkan dengan maksimal. Pendistribusian sistem journal pada saat ini yang dikembangkan oleh Kementerian Pendidikan dan Budaya dirasa kurang fleksibel dikarenakan harus menggunakan produk yang sama yaitu SLIM (*Senayan Library Management System).* Sehingga menyebabkan kurangnya efektif bila digunakan oleh berbagai multiplatform dalam penggunaan teknik pendistribusian data tersebut.

*Resource Oriented Architecture (ROA)* suatu teknologi arsitektur pengembangan perangkat lunak dengan pendekatan layanan, memungkinkan hubungan dan pertukaran data atau informasi antar bagian menjadi mudah. Resources ini dapat memaparkan fungsi-fungsi dari mereka melalui tampilan interface. Melalui standar bahasa *Web Service Definition Language (WSDL)* atau *Web Application Definition Language (WADL)* sehingga penamaan deskripsi pada suatu Resource mudah dimengerti oleh manusia.

Arsitektur REST, yang umumnya dijalankan via *HTTP (Hypertext Transfer Protocol)*, melibatkan proses pembacaan hlaman web yang memuat sebuah file JSON. File inilah yang akan menguraikan dan memuat konten yang hendak disajikan. (Roberto Lucchi, 2008) pada penelitiannya dengan judul "*Resource Oriented Architecture and Rest",*di mereview konseptual REST style, dengan kesimpulan bahwa RESTful Web Service telah menjadi alternatif yang sangat baik untuk kompleksitas pada WS-*.

Berdasarkan kesimpulan diatas maka Platform teknologi Resource Oriented Arichitecture (ROA) dan REST sebagai solusi untuk spasialis infrastruktur data terdistribusi. Sehingga journal-

journal diberbagai universitas dapat diberikan kepada beberapa perpustakaan dan mendapatkan informasi yang lengkap dan tersetruktur di berbagai platform dan mengetahui *performance* pada setiap platform.

## B. RUMUSAH MASALAH

### 1. Identifikasi Masalah

Identifikasi masalah dalam pendistribusian data journal menggunakan RESTful APIs berbasis multiplatform.

a. Perpustakaan yang memiliki journal tidak menggunakan konsep ROA.
b. Sistem journal hanya dapat menggunakan produk yang sama.
c. Belum ada sistem pendistribusian data yang bersifat multiplatform..

### 2. Problem Statement

Belum adanya sistem pendistribusian data yang berbasis multiplatform sehingga journal diberbagai universitas lambat jika dilakukan secara konvensional .

### 3. Research Question

Bagaimana penerapan *Resource Oriented Architecture* untuk pendistribusian data journal keberbagai perpustakaan yang berbasis secara multiplatform.

## C. MAKSUD DAN TUJUAN

1. Maksud

   Menerapkan metode *Resource Oriented Architecture* menggunakan RESTful API pada web service berbasis multiplatform.

2. Tujuan Penelitian

   memberikan suatu solusi untuk pendistribusian data yang bersifat multiplatform dan flexible yang akan diterapkan di sistem perpustakaan. Serta meninjau platform yang mana yang cocok digunakan untuk data terdistribusi.

## D. KEGUNAAN DAN MANFAAT PENGEMBANGAN

1. Kegunaan

   Kegunaan penelitian ini adalah untuk dapat mendistribusikan data journal ke perpustkaan universitas dengan berbagai multiplatform.

2. Manfaat

   a. Manfaat teoritis ini adalah melanjutkan penelitian sebelumnya dan memberikan sumbangan terhadap IPTEK khususnya untuk membuktikan *Resource Oriented Architecture* dapat digunakan untuk pendistribusian data ke berbagai platform menggunakan *RESTful API* sebagai *development*-nya.
   b. Manfaat praktis pada penelitian ini adalah untuk memudahkan pendistribusian data journal dari berbagai universitas secara efektif dan efisien.

## E. PENTINGNYA PENGEMBANGAN

Penelitian ini dibangun karena adanya masalah dalam pendistribusian data-data journal keberbagai perpustakaan yang tidak terorganisir dengan baik.

Penelitian ini memberikan solusi untuk permasalahan tersebut, yang diuraikan sebagai berikut:

1. Dapat memberikan referensi journal bagi peneliti yang sedang melakukan penelitian.
2. Melengkapi journal yang ada diperpustakaan tersebut.
3. Memudahkan pengintegrasian data yang terdistribusi karna bersifat multiplatform

## F. RUANG LINGKUP DAN KETERBATASAN

Ruang lingkup pada penelitian ini meliputi:

1. Pengumpulan data journal yang disediakan hanya didapatkan dari website. Misalnya seperti http://journal.ui.ac.id/technology, http://journals.itb.ac.id dan http://www.researchpublish.com.

Adapun keterbatasan dari penerapan ini adalah:

1. Security system harus sangat diperhatikan untuk menjaga sebuah database, karena data journal pada setiap sistem perpustakaan tergantung pada system yang didistribusikan oleh provider.

## G. DEFINISI ISTILAH

1. WSDL (*Web Services Description Language*) adalah sebuah XML-based language untuk mendeskripsikan XML.
2. Web service adalah aplikasi perangkat lunak yang dapat diakses secara remote oleh berbagai piranti dengan sebuah perantara tertentu.
3. Representasi adalah proses dimana sebuah objek ditangkap oleh indra seseorang, lalu masuk ke akal untuk diproses yang hasilnya adalah sebuah konsep/ide yang dengan bahasa akan disampaikan/diungkapkan kembali.
4. Resources adalah segala informasi yang dapat diberikan nama, seperti, dokumen, gambar atau daftar masalah terbuka dalam versi software.
5. *Hypertext Transfer Protocol* (HTTP) adalah sebuah protokol lapisan jaringan aplikasi yang digunakan untuk sistem informasi terdistribusi, kolaboratif, dan menggunakan hipermedia.
6. URI (*Uniform Resource Identifier)* adalah pengidentifikasian dokumen tunggal dan dituliskan dalam satu baris teks.
7. URL (*Uniform Resource Locator*) adalah pengidentifikasian sumberdaya di Internet yang dituliskan dalam satu baris teks.

[ Halaman ini sengaja dikosongkan ]

# BAB II

## KERANGKA TEORITIS

## A. TINJAUAN PUSTAKA

1. “Developing Distributed System with Service Resource Oriented Architecture” oleh Hermawan dari Fakultas Teknik Informatika, Universitas Trunojoyo Madura, Indonesia dab Riyanarto Sarnom, Fakultas Teknik Informatika, Institut Teknologi Sepuluh Nopember, Indonesia, 2012. Penelitian ini menjelaskan mengenai fungsionalitas aplikasi sebagai sebuah layanan melalui teknologi layanan web yang disebut dengan *Simple Object Access Protocol (SOAP).* Adapun SOAP bersifat statis dan berorientasikan pada metode layanan sehingga memiliki keterbatasan dalam pembuatan dan pengaksesannya pada jumlah layanan yang besar. Karena itu dalam penelitian tersebut dilakukan kombinasi *Service Oriented Architecture (SOA)* dan *Resource Oriented Architecture (ROA)* yang berorientasi pada sumberdaya layanan menggunakan *REpresentational State Transfer (REST)* untuk memperluas cakupan layanan. Kombinasi arsitektur ini disebut sebagai *Service Resource Oriented Architecture (SROA)*. SROA mampu mengoptimalkan distribusi aplikasi dan integrasi layanan yang diimplementasikan untuk membangun perangkat lunak manajemen. Untuk merealisasikan model ini, perangkat lunak dibangun sesuai dengan kerangka kerja *Agile Model Driven Development (AMDD)* untuk mengurangi kompleksitas pada semua tahapan proses pembangunan perangkat lunak.
2. “Design & Development of a REST based Web Service Platform for Applications Integration on Cloud” oleh Ritesh Sinha, Manisha Khatkar, Subhash Chand Gupta dari Amity University, India, September 2014. Penelitian ini mengenai pendekatan REST based method untuk komunikasi dan transfer data dari satu aplikasi ke aplikasi lain menggunakan Cloud Platform seperti Force.com platform pada Salesforce. Penelitian ini menggunakan standar*Simple Object Access Protocol (SOAP)*, *Extensible Markup Language (XML)* dan *Java Script Object Notation (JSON)* untuk berinteraksi dengan Cloud. Tujuan pada penelitian ini adalah untuk Design & Development pada REST berbasis Web Service platform untuk integrasi dengan Cloud. Penelitian ini juga diselesaikan dengan Apex language, The Propriety Language pada Salesforce.
3. “Preventing Buffer Overflow DOS Attack in Restful Services” oleh Hussam A. Elkurd , Tawfeeq S.Barhoom, Fakultas Teknologi Informasi, Islamic University Gaza, Gaza, Palestine 2015 yang diterbikan oleh International Journal of Information and Communication Technology Research. Pada penelitian ini bermaksud untuk membuat sebuah mekanisme pada pencegahan serangan DOS buffer overflow dan membatasi waktu berlaku *token authorized*yang telah ditentukan. Selain itu penelitian ini menerapkan sebuah web RESTful API dan menunjukan hasil yang mampu mencegah serangan DOS Attack. Salah satu cara mereka untuk mencegah serangan tersebut dengan Token Lifetime biasa digunakan pada akses Authorized pada resources, token tersebut digenerated dari provider dengan limit interval, jika batas waktu telah berakhir maka token menjadi tidak valid dan token baru harus digenerated kembali.

Untuk mekanisme pencegahan ini mereka membuat sebuah REST API menggunakan PHP script.

4. “Rancang Bangun Perangkat Lunak Aplikasi Kesehatan Berbasis Service Oriented Architecture” oleh Sarwosri, Farah Naja, Fakultas Teknologi Informasi, Jurusan Teknik Informatika di Institute Teknologi Sepuluh Nopember, 2011. Dalam penelitian ini peneliti membangun sebuah aplikasi rekam medis untuk memudahkan pasien dalam mendapatkan info dan fasilitas kesehatan yang dibutuhkan tanpa perlu mengunjungi satu persatu web rumah sakit menggunakan *Service Oriented Architecture (SOA)*. Di penelitian ini juga mereka membagi proses bisnis menjadi dua, yaitu: proses bisnis aplikasi pelayanan kesehatan dan proses bisnis aplikasi rumah sakit.
5. “Perancangan Basis Data dan Layanan Akses Berbasis Service Oriented Architecture (SOA) Untuk Dinas Kesehatan Kabupaten Sleman” oleh Inna Yoana Sari Tarigan S, Soedjatmiko, Rudy Hartanto, Program Studi Magister Teknologi Informasi, Program Pasca Sarjana Universitas Gadjah Mada, Yogyakarta, Indonesia, 2010. Pada penelitian ini peneliti membantu sebuah proses pengambilan keputusan kebijakan dan sector kesehatan dengan memanfaatkan teknologi komunikasi dan informasi. Penelitian ini juga bertujuan untuk merancang sebuah basis data dan layanan akses berbasis SOA untuk pusat data transaksional pada Dinas Kesehatan Kabupaten Sleman. Data transaksional dalam penelitian ini adalah data rekam medis pasien, logistic alat-alat kesehatan, pengirimannya, catatan penyimpanan, catatan perpindahan tempat termasuk data tenaga medis disebuah puskesmas. Jalannya penelitian yang dilakukan meliputi tahap analisa kebutuhan dan tahap pengembangan sistem. Tahap analisa kebutuhan dilakukan dengan pengumpulan data studi literatur dan survey kondisi data yang tersedia dipuskesmas dan kebutuhan informasi Dinas Kesehatan Kabupaten Sleman. Sedangkan tahap pengembangan sistem meliputi: Melakukan analysis terhadap sistem informasi puskesmas dan basis data puskesmas, membuat rancangan basis data yang dapat mengintegrasikan data transaksional puskesmas, membuat *prototype* aplikasi untuk mengirimkan data puskesmas ke basis data yang dirancang, menguji prototype aplikasi dengan dengan mengirimkan data puskesmas ke basis data, membuat report dan prototype sistem untuk mengakses report tersebut, guna memenuhi kebutuhan informasi Dinas Kesehatan Kabupaten Sleman, menentukan layanan berbasis SOA untuk mengakses data (informasi) yang dimiliki Dinas Kesehatan, membuat sample client untuk mengakses layanan yang tersedia (untuk kepentingan pengujian), melakukanpengujian ketersediaan layanan dengan mengakses layanan menggunakan program sample yang telah dibuat.
6. “REST and Resource-Oriented Architecture” oleh Roberto Lucchi, University of Western Ontario London, Romania 2011. Pada penelitian ini peneliti mereview REST style, konseptual dan karakteristik ROA, Security pada RESTful Web Service. Pada kesimpulannya peneliti menyimpulkan bahwa, REST adalah sebuah gaya arsitektur yang dibuat untuk menjelaskan sebuah hypermedia distributed system. Kontribusi utama pada journal ini adalah untuk meninjau aspek penting dari arsitektual REST dan Resource Oriented Architecture pada

perbandingan dari WS-*. Dan juga hasil yang lain bahwa, RESTful Web Service telah menjadi alternatif yang sangat baik untuk kompleksitas pada WS-*.

Berdasarkan penelitian yang berhubungan diatas, maka judul yang diajukan pada pnelitian ini adalah “Penerapan *Resource Oriented Architecture* Untuk Pendistribusian Data Menggunakan RESTful APIs Berbasis Multiplatform”.Judul tersebut didasari oleh permasalah di perpustakaan universitas, yaitu belum adanya pendistribusian data journal dari berbagai perpustakaan ke perpustakaan lain yang mengakibatkan lambatnya pencarian data journal oleh peneliti sebagai referensinya. Sedangkan variable yang digunakan adalah*http method* diantaranya yaitu GET dan POST. Berdasarkan penelitian-penelitian sebelumnya terdapat saran untuk menggunakan metode lain selain *Resource Oriented Architecture* yaitu dengan metode *Service Oriented Architecture* sebagai pendistribusian data.

## B. LANDASAN TEORI

### 1. Resource Oriented Architecture (ROA)

ROA didasarkan pada the concept of resource, setiap resource adalah komponen sebuah pendistribusian yang dapat diakses secara langsung yang ditangani oleh beberapa standar. Resource interface semantics didasari oleh operasi-operasi HTTP (Roberto Lucchi, 2008). Pada ROA, prinsip REST menjadi sangat terkait dengan Web protocols, seperti *Uniform Resource Identifier* (URIs), *Hypertext Transfer Protoco l* HTTP and *eXtensible Markup Language* (XML) (R. Fähnrich, 2009). Berikut table rangkuman metode-metode resource dan bagaimana mereka dapat diterapkan pada protokol *Hypertext Transfer Protocol* (HTTP) menurut, (Roberto Lucchi, 2008).

| **HTTP operations** | **CRUD** | **Description** |
|---|---|---|
| POST | Create | Menciptakan sebuah resource |
| GET | Read | Membaca sebuah resource atau collection |
| PUT | Update | Biasa digunakan untuk update sebuah data |
| DELETE | Delete | Menghapus sebuah resource |

*Table 1: HTTP methods, (Roberto Lucchi, 2008)*

Definisi status HTTP codes pada HTTP protocol, berikut table2 yang dijelaskan oleh (Ritesh Sinha, 2014).

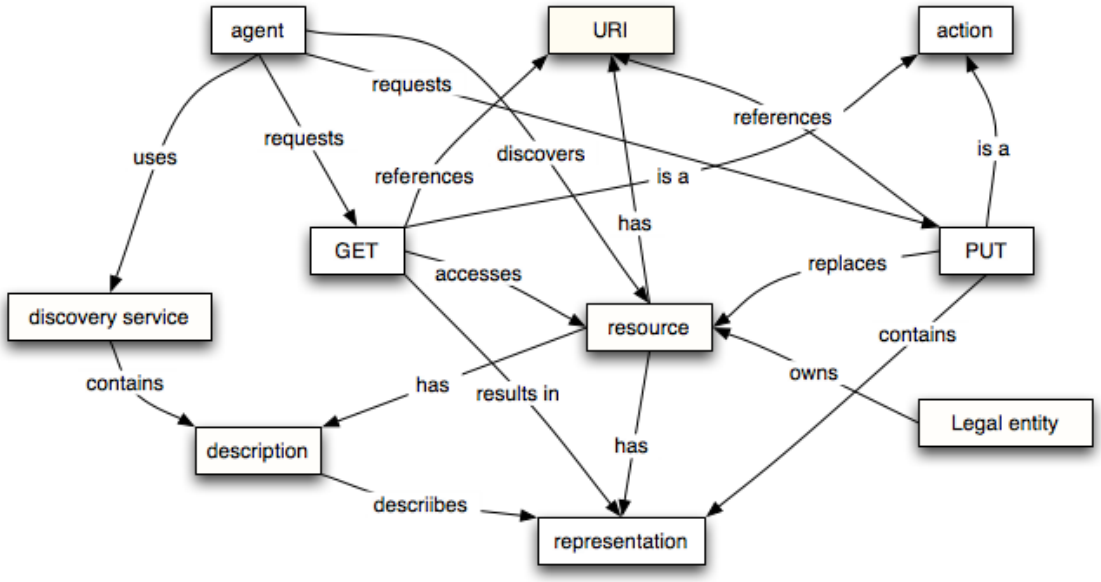
| **Status** | **Code** | **Description** |
|---|---|---|
| Success | 200 | Berhasil mengambil data, mengubah dan menghapus |

| | | |
|---|---|---|
| Unauthorized | 401 | kredensial tidak tersedia atau salah |
| Not Found | 404 | Service tidak disediakan oleh provider |
| Internal server error | 500 | Kesalahan tak terduga |
| Unprocessable Entity | 422 | Tidak berhasil menyimpan, mengubah atau menghapus data |

*Table 2: Menampilkan kode HTTP status, (Roberto Lucchi, 2008)*

Jelas, bahwa konsep utama ROA adalah *resource/informasi.* Dalam struktur ini, sebuah *resource* memiliki nama alamat yang dapat direpresentasikan oleh URI (Universal Resource Identifier).

Istilah 'RESTful Web Services' pada umumnya digunakan untuk menunjukanWeb Service berdasarkan style ini. Menurut (Leonard Richardson, 2007), mereka mengatakan, definisi sebuah arsitektur baru, dikenal sebagai *Resource Oriented Architecture (ROA)*, untuk mendevelop sebuah RESTful Web Services.




*Gambar 1: Konsep dan relasi pada Resource Oriented Model, (W3C, 2003)I*

Resource Oriented Model dapat menampilkan bagaimana *resources* merupakan konsep yang independen, namun manipulasi pada *resources* adalah sebuah turunan dari Service Model: dengan jenis tertentu pada *services* dan diindentifikasikan pada *resources* (W3C, 2003).

**2. RESTful APIs**

REST adalah suatu metode komunikasi yang sering diterapkan dalam pengembangan layanan berbasis web.REST, yang umumnya dijalankan via HTTP (Hypertext Transfer Protocol), melibatkan proses pembacaan laman web tertentu yang memuat sebuah file XML atau JSON. File inilah yang menguraikan dan memuat konten yang hendak disajikan. Setelah melalui sebuah proses definisi tertentu, konsumen akan bisa mengakses antarmuka aplikasi yang dimaksudkan (Saputra, 2015)[6].

Kekhasan REST terletak pada interaksi antara klien dan server yang difasilitasi oleh sejumlah tipe operasional (verba) dan Uniform Resource Identifiers (URIs) yang unik bagi tiap-tiap sumberdaya. Masing-masing verba – GET, POST, PUT dan DELETE – memiliki makna operasional khusus untuk menghindari ambiguitas (Saputra, 2015).

RESTful platform itu sendiri didasari oleh *REST development techolonogy* yang dihasilkan oleh ROA (Roberto Lucchi, 2008). Layanan berbasis web yang biasa menggunakan arsitektur REST semacam itu dinamakan RESTful APIs (*Application Programming Interface*) atau REST APIs.

API *(Application Programming Interface*) telah menjadi bagian penting dari industri komputersejak awal. mereka mendasar untuk cara komputer, perangkat lunak, dan jaringanarsitektur telah berkembang.

Sebuah representasi dapat dimasukkan kedalam pesan dan diproses oleh penerima sesuai dengan data kontrol pesan dan sifat dari jenis mediayang dikirimkan (Fielding, 2000). Beberapa jenis yang dapat dimasukan diantaranya berupa pesan yang dapat dilihat rapih oleh pengguna, sehingga pesan yang disampaikan mudah dimengerti.

**3. JSON**

JSON *(JavaScript Object Notation)* adalah adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer (json.org, 2000). JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

JSON memiliki dua struktur:

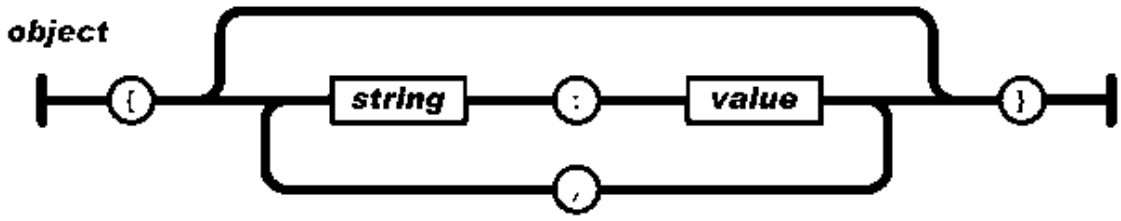
8. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (object), rekaman (record), struktur (struct), kamus (dictionary), tabel hash (hash table), daftar berkunci (keyed list), atau associative array.
9. Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai (array), vektor (vector), daftar (list), atau urutan (sequence).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini (json.org, 2000).

Dibawah ini beberapa bentuk dari format data JSON

1. Object

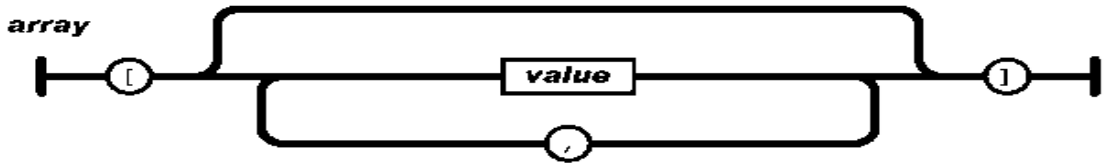
   Object adalah sepasang nama/nilai yang tidak terurutkan. Biasa dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma), (json.org, 2000).




*Gambar 2: Struktur (object) pada format data JSON, (json.org, 2000)*

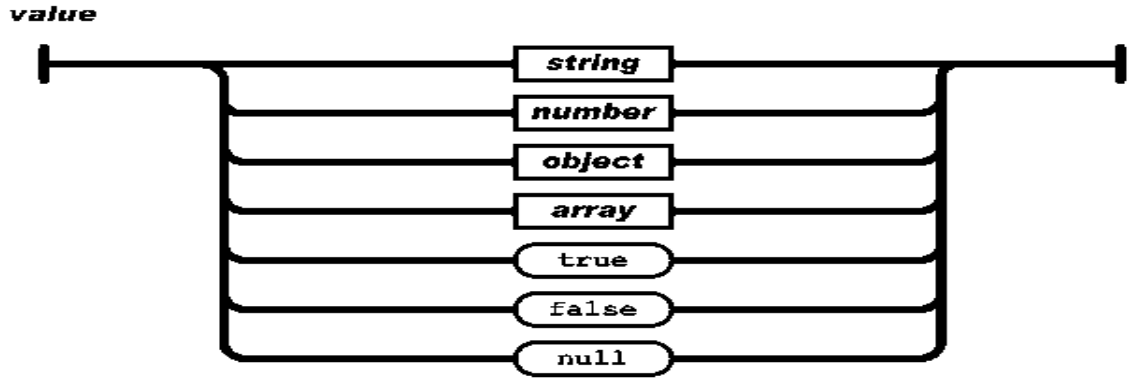
2. Array

   Array adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ (kurung kotak buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma), (json.org, 2000).




*Gambar 3: Struktur (array) pada format data JSON, (json.org, 2000)*

3. Value




*Gambar 4: Struktur (value) pada format data JSON, (json.org, 2000)*

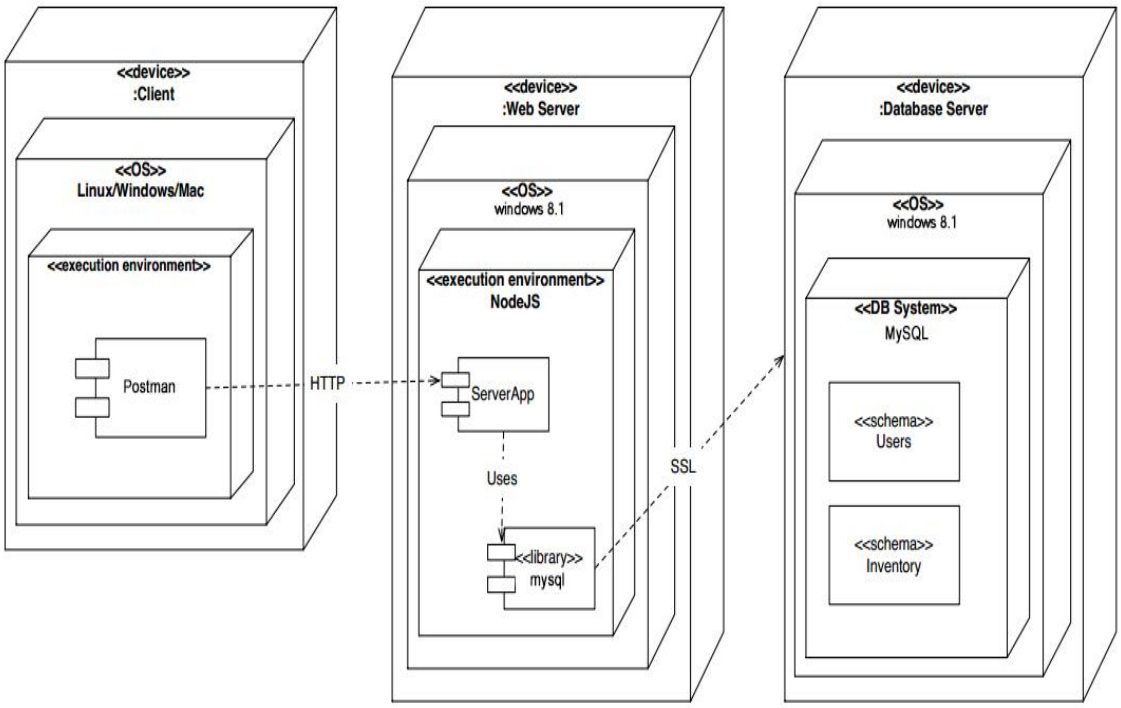
**Nilai** (*value*) dapat berupa sebuah **string** dalam tanda kutip ganda, atau *angka*, atau `true` atau `false` atau `null`, atau sebuah *objek* atau sebuah *larik*. Struktur-struktur

tersebut dapat disusun bertingkat, (json.org, 2000).

**4. Node.js**

Node.js merupakan bahasa pemrograman Javascript yang dijalankan di serverside. Tapi walaupun begitu Node.js tidak termasuk dalam Javascript Framework (Orsini, 2013).Javascript merupakan bahasa pemrograman yang lengkap hanya saja selama ini di pakai sebagai bahasa untuk pengembangan aplikasi web yang berjalan pada sisi client atau browser saja. Tetapi sejak ditemukannya Node.js oleh Ryan Dhal pada tahun 2009, Javascript bisa digunakan sebagai bahasa pemrograman di sisi server kelas dengan PHP, ASP, C#, Ruby dll dengan katak lain Node.js menyediakan platform untuk membuat aplikasi Javascript dapat dijalankan di sisi server.

Javascript biasa digunakan dibagian interface dengan Node bisa dilakukan dengan 2 prespektif yaitu client dan server, berikut gambar deplyoment diagaram (Dahl, 2013).

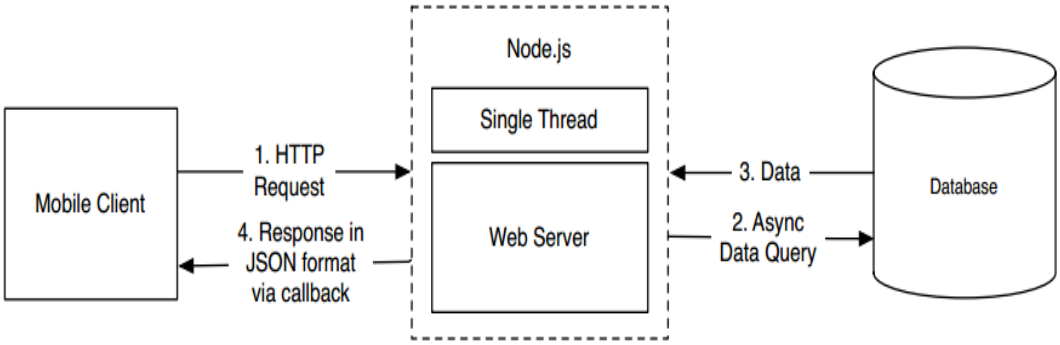



*Gambar 5: Deployment Diagram pada sistem Node.js, (Dahl, 2013)*

Node.js dibagi menjadi dua komponen: core dan modules. Core dibuild menggunakan C dan C++. Itu kombinasi dari Google V8 Javascript dengan Node's Libuv library dan protokol yang digunakan seperti sockets dan HTTP.
Proses instalasi Node sangatlah mudah dengan mengakses platform pada website sehingga user bisa menginstall dengan versi yang sesuai (Benjamin San Souci, 2014).

a. **Architectural Description**

Node JS menggabungkan jaringan arsitektur untuk efisiensiensi penginputan data dan menghasilkan sesuai dengan output. Ini mengikuti struktur yang sangat modular, tapi tetap erat digabungkan ke *library*, khususnya mesin V8 (Letaifa, 2011). Sebagai platform harus berkoordinasi input dan output data pada berbagai *sistem terdistribusi*.




*Gambar 6: Gambar diatas menggambarkan proses NodeJS merequest data akses kedatabase menurut (Benjamin San Souci, 2014)*

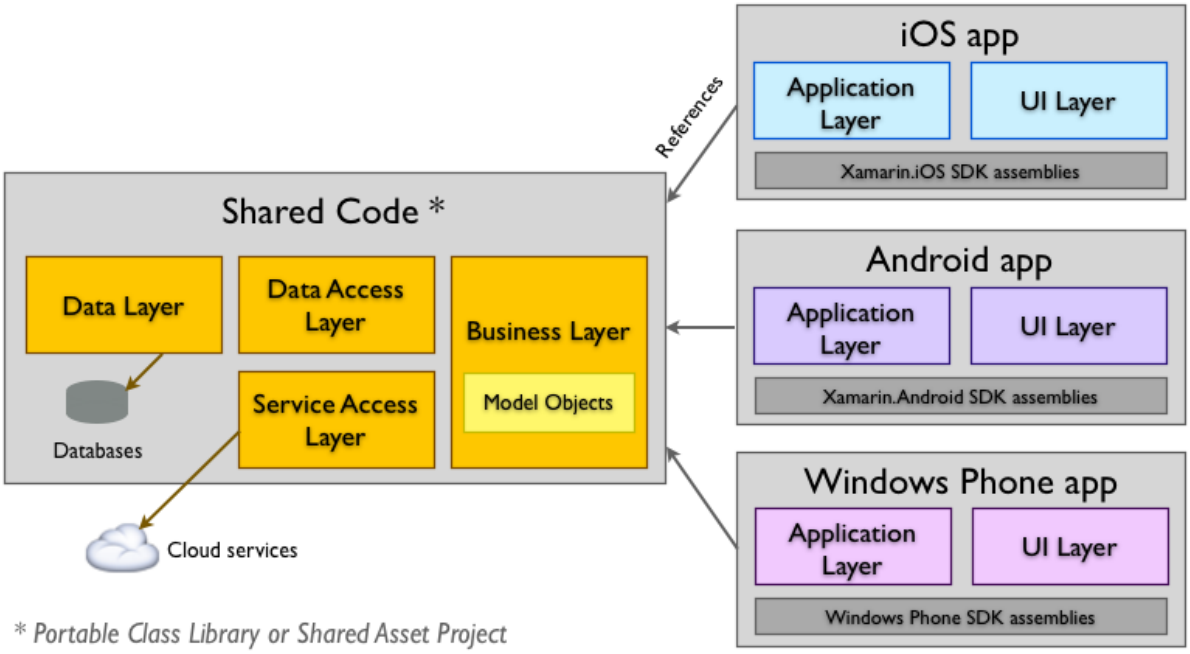
**b. Distributed Style**

Distributed Style architecture membahas keputusan desain yang penting sehingga terkait dengan collection computational dan perangkat penyimpanan yang berkomunikasi melalui jaringan. Pada umumnya, ini terdiri dari komputer remote secara independen mengeksekusi aplikasi, dan berkomunikasi dengan server independen melalui jaringan.

**5. Cross Platform Development**

Cross platform development yang muncul untuk menghadapi developers yang akan menerapkan aplikasi mereka dalam satu tahapan untuk berbagai platform, menghindari pengulangan development dan meningkatkan produktivitas (Henning Heitk¨otter, 2012).

Tahap integrasi adalah dimana aplikasi mobile bisa dilakukan pada fasilitasi IT perusahaan, *cloud* infrstruktur dan kemampuan perangkat asli. Integrasi akan membantu memperluas fungsi dengan fitur tambahan yang bergantung pada system lain (Redda, 2012).




*Gambar 7: Ilustrasi arsitektur aplikasi pada cross platform (Xamarin, 2014).*

**6. Database**

Sebuah *database* dapat digambarkan sebagai repository untuk data. Dengan jelas ini bahwa membangun database benar-benar merupakan kelanjutan kegiatan yang dimiliki manusia. Database dapat diterapkan pada hasil setiap pembukuan atau kegiatan perekaman yang terjadi jauh sebelum munculnya era komputer (Robbins, 1995). DBMS (Database Management System)adalah sistem perangkat lunak untuk membuat dan mengelola database (Rouse, 2015).DBMS juga melakukan fungsi pengaturan, pengawasan, pengendalian, pengolahan, dan koordinasi terhadap semua proses yang terjadi pada sistem basis data. Dibawah ini adalah komponen-kompnen utama dalam DBMS:

a. Query Language

   Digunakan oleh bagian lain dengan sedikit perintah sederhana. Contoh : SQL (Structure Query Language), QBE (Query By Example)

b. DML (Data Manipulation Language)

   Terdiri dari perintah-perintah yang disediakan dalam program aplikasi untuk melakukan manipulasi data seperti append, list, atau update

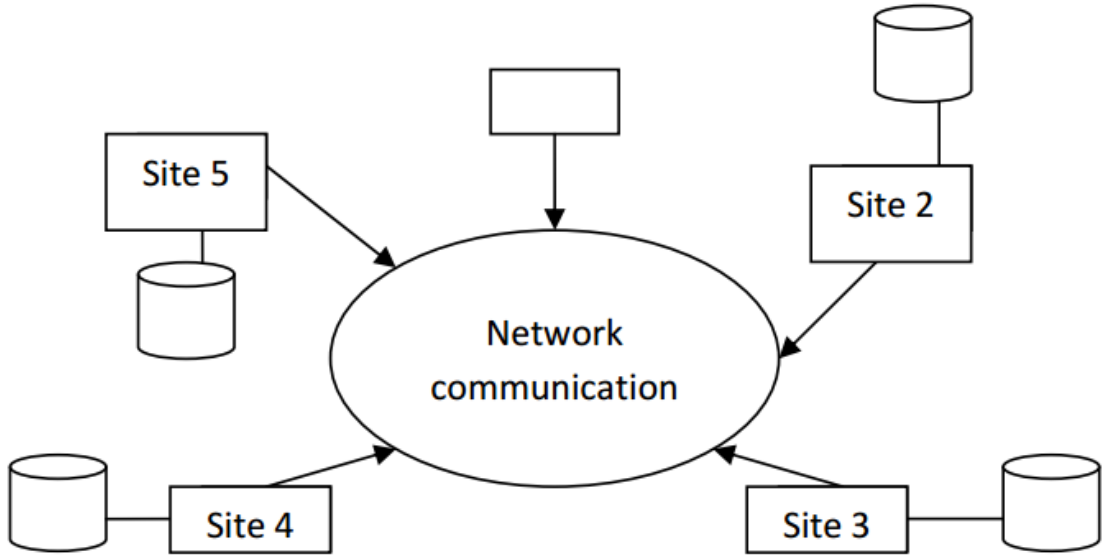
c. DDL (Data Definition Language)

   Dengan bahasa ini kita dapat membuat tabel baru, membuat indeks, mengubah tabel, menentukan struktur tabel, dll. Hasil dari kompilasi perintah DDL menjadi Kamus Data, yaitu data yang menjelaskan data sesungguhnya. Contoh : Create, Modify report, Modify structure

**7. Distributed Database System**

Database terdistribusi sebagai kumpulan beberapa, database secara logis saling terkait didistribusikan melalui jaringan komputer. Sebuah sistem manajemen database terdistribusi (DBMS terdistribusi) lalu didefinisikan sebagai sistem perangkat lunak yang memungkinkan manajemen database terdistribusi dan membuat distribusi transparan kepada pengguna (Aspects of the design of distributed databases, 2011). Biasanya (DDBS) digunakan bersama-sama untuk merujuk ke database terdistribusi dan DBMS terdistribusi. Dua istilah penting dalam definisi ini "secara logis saling terkait" dan "didistribusikan melalui jaringan komputer."

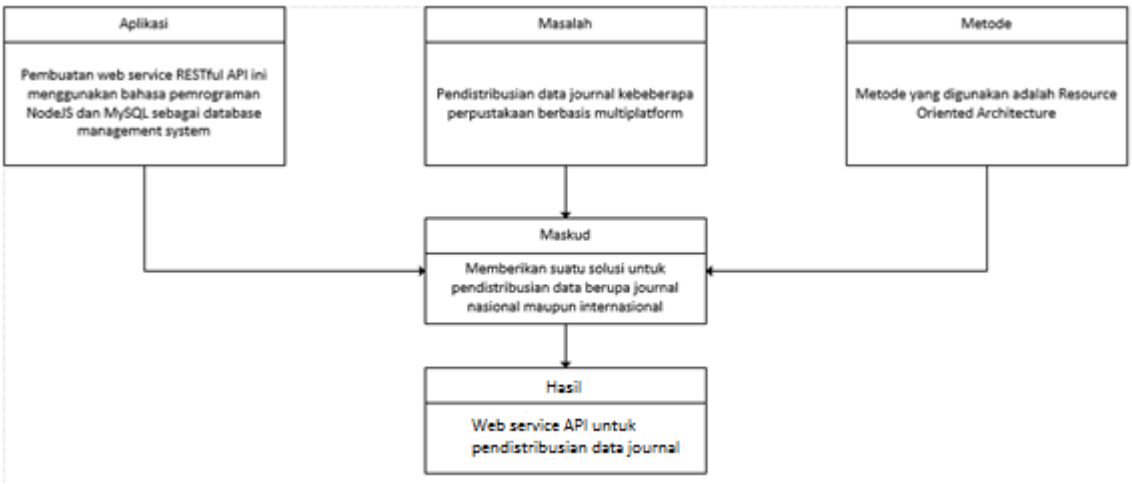
Distribute Database System tidak hanya "sebuah koleksi pada file" tapi itu bisa menjadi media penyimpanan individual dalam setiap node jaringan pada komputer (Aspects of the design of distributed databases, 2011).




*Gambar 8: Environtment Distribute Database System, (Aspects of the design of distributed databases, 2011)*

## C. KERANGKA PEMIKIRAN

Berikut merupakan kerangka pemikiran pemecahan masalah dalam penelitian ini yang digambarkan pada gambar dibawah ini.




*Gambar 9: Kerangka pemikiran*

Dari gambar 8 maka dapat dijelaskan kerangka pemikirannya, yaitu:

1. Terdapat permasalahan distribusi data journal keberbagai perpustakaan
2. Dengan adanya metode *Resource Oriented Architecture* (ROA) distribusi data dapat dilakukan dengan efisien.
3. RESTful API digunakan sebagai media penerapan ROA
4. Pemrograman NodeJS digunakan untuk menarik resource data journal dari database
5. Database yang digunakan adalah MySQL, dengan adanya MySQL yang berorientasi RDBMS dapat memungkinan relasi data keberbagai table menjadi lebih mudah
6. Database ditangani oleh aplikasi sehingga data dapat terdistribusi keberbagai multiplatform menggunakan format data JSON sebagai pertukaran data.

[ Halaman ini sengaja dikosongkan ]

# BAB III

## METODE PENGEMBANGAN

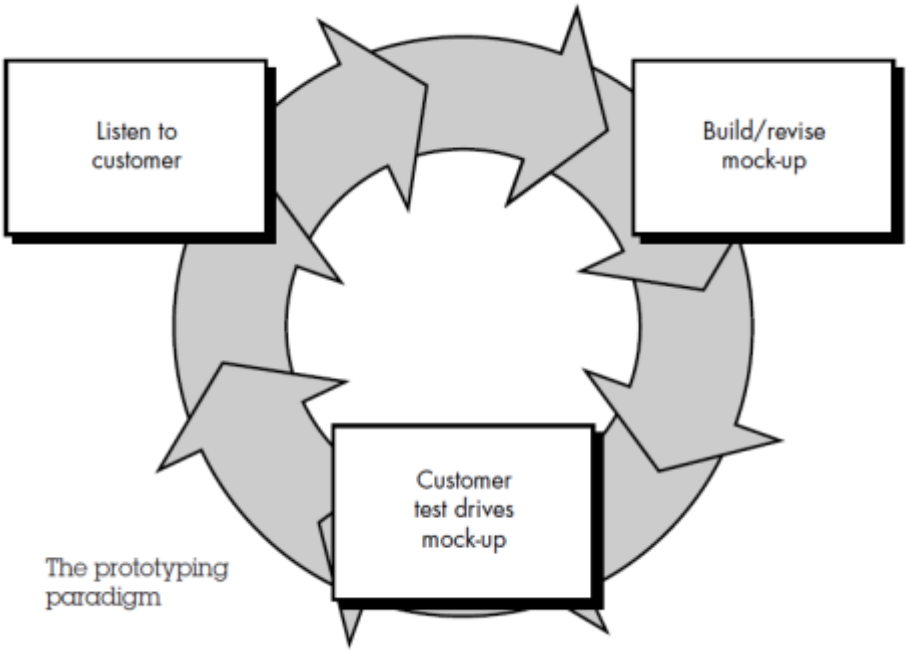
## B. MODEL DAN PROSEDUR PENGEMBANGAN

### 1. Model Pengembangan

Proses pengembangan sistem seringkali menggunakan pendekatan prototipe (prototyping). Metode ini sangat baik digunakan untuk menyelesesaikan masalah kesalahpahaman antara user dan analis yang timbul akibat user tidak mampu mendefinisikan secara jelas kebutuhannya (Mulyanto, 2009).

Prototyping adalah pengembangan yang cepat dan pengujian terhadap model kerja (prototipe) dari aplikasi baru melalui proses interaksi dan berulang-ulang yang biasa digunakan ahli sistem informasi dan ahli bisnis. Prototyping disebut juga desain aplikasi cepat (rapid application design/RAD) karena menyederhanakan dan mempercepat desain sistem (O'Brien, 2005).

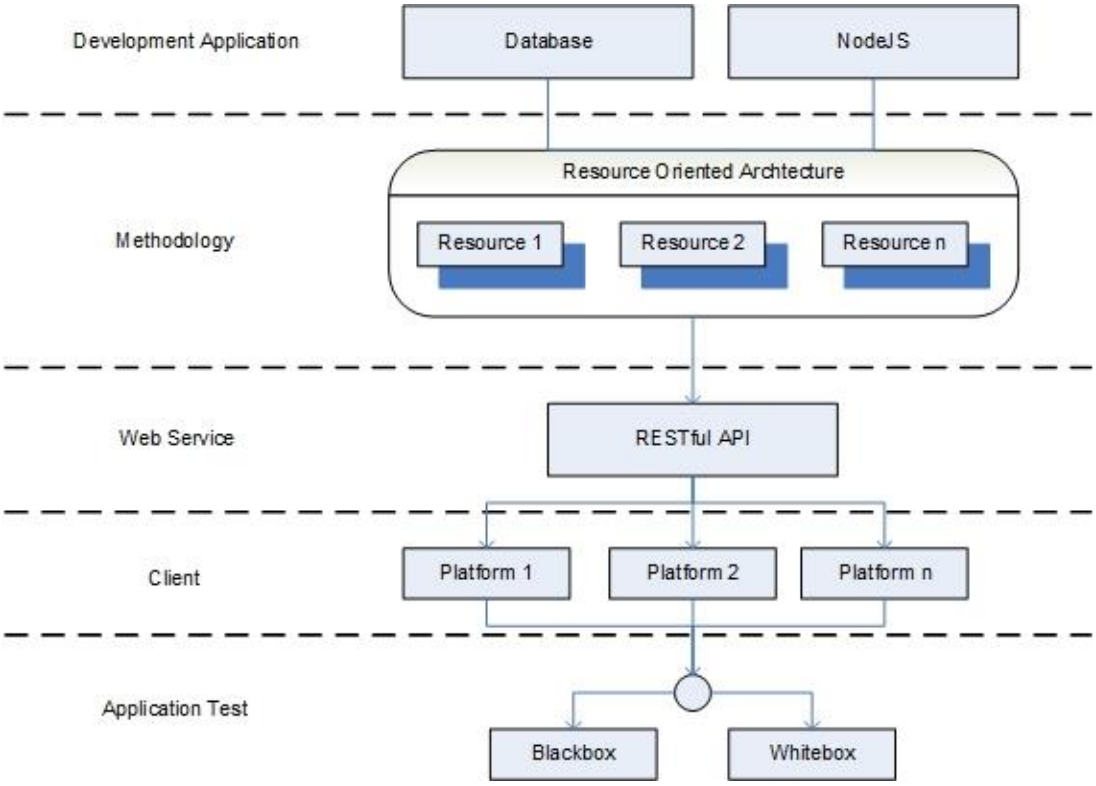
Model tersebut dapat diamati pada gambar 9 dibawah ini




*Gambar 10: Model prototype menurut Roger S. Pressman*

**2. Prosedur Pengembangan**

Merujuk pada model pengembangan yang menggunakan *Protoyping*, maka prosedur dalam pengembangan ini berisi langkah-langkah pembuatan *webservice* untuk pendistribusian data journal




*Gambar 11: Prosedur pengembangan*

a. Development Application

Pada tahap ini adalah tahap untuk membangun sebuah aplikasi berbasis *web service* yang menggunakan database MySQL dan menggunakan NodeJS sebagai bahasa pemrogramannya.

b. Methodology

Metode yang digunakan adalah *Resource Oriented Architecture* sebagai komponen pendistribusian data berbasis *resource* atau sumber data.

c. Web Service

RESTful diperuntukan sebagai *middleware* aplikasi yang mengintegrasikan *resource* kepada platform lain dan HTTP sebagai protokolnya.

d. Client

Pada penelitian pengembangan ini platform-platform adalah sebagai client yang nantinya akan menjadi experiment pendistribusian data

e. Application Test

Pada tahap ini dilakukan pengujian terhadap platform-platform yang menggunakan teknik RESTful API dan diuji dampak yang terjadi saat pendistribusian data dilakukan.

## C. UJI COBA PRODUK

### 1. Desain Uji Coba

Pengujian hanya dilakukan dengan menggunakan metode blackbox, terfokus pada apakah unit program memenuhi kebutuhan yang disebutkan dalam spesifikasi. Pengujian hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan

### 2. Subjek Uji Coba

Yang menjadi subjek uji coba adalah platform dengan bahasa pemrograman Java, Python dan PHP sebagai media pengujian. Pengujian dilakukan untuk mengukur kecepatan dan juga ketepatan data distribusi bagi pengguna. Selain itu pengujian dilakukan untuk mendeteksi kesalahan yang mungkin terjadi dalam penulisan kode program sehingga diharapkan aplikasi dapat berjalan dengan baik.

## D. INSTRUMEN PENGUMPULA DATA

Instrumen pengumplan data dalam proses penelitian pengembangan ini adalah MySQL untuk database pengumpulan data dan Sublime Text 3 sebagai *text editor* bahasa pemrograman NodeJS, dan uji indexing sebagai mengukur selisih waktu antara setiap platform yang menggunakan RESTful.

## E. UJI ANALISIS DATA

Uji analisis data dilakukan untuk mengetahui tujuan penelitian pengembangan yaitu menentukan kecepatan rata-rata waktu antara platform dalam pendistribusian data menggunakan RESTful. Variable yang diuji adalah menggunakan waktu antara platform-platform yang menggunakan *HTTP request* (GET, POST) dan banyak data yang diberikan.

| **PLATFORM** | **METHOD** | **REQUEST** | **DATA**(s) | **WAKTU** (t) |
|---|---|---|---|---|
| 1 | | | | |
| ↓ | | | | |
| n | | | | |
| Total | | | | |

*Table 3: Table analisis data*

Untuk menentukan kecepatan rata-rata yaitu dengan rumus $v = \frac{s_{total}}{t_{total}}$

Halaman ini sengaja dikosongkan

# BAB IV

## HASIL DAN PEMBAHASAN

### A. DESKRIPSI OBJEK PENELITIAN

Objek dari penelitian ini ditekankan kepada data journal dari berbagai universitas ataupun instansi yang dikumpulkan dalam satu database dan di *compile* menjadi beberapa *resource* dengan tipe data *json.* Hal ini dimaksudkan untuk menguji seberapa besar pengaruh antara *resource – resource* tersebut terhadap *multiplatform programming* sebagai objek penelitian ekperimen ini diantaranya yaitu Android, PHP dan Python. Penelitian yang akan digunakan adalah menerapkan teknik RESTful APIs dan NodeJS sebagai development-nya serta *Resource Oriented Architecture* sebagai metode penerapan teknik-teknik tersebut.

Aplikasi ini dibuat menggunakan NodeJS v4.4.7 Beberapa plugin untuk mendukung aplikasi *RESTful* ini antara lain yaitu: express, mysql, body-parser, response-time, crypto.

DBMS yang digunakan untuk membangun *Web Service* ini adalah MySQL v5.6. Table yang digunakan pada web service adalah table journal. Dibawah ini adalah struktur table.

| **Table Journal** | | | |
|---|---|---|---|
| **Fieldname** | **Type** | **Length** | **Description** |
| id (pk) | INT | 11 | Identitas Jurnal |
| title | STRING | 220 | Judul journal |
| author | STRING | 200 | Penulis |
| abstract | TEXT | | Abstract journal |
| keywords | STRING | 255 | Keywords |
| image | STRING | 150 | URL Cover image |
| file | STRING | 150 | URL Download file |
| issn | STRING | 30 | Nomor ISBN/ISSN |
| publisher | STRING | 150 | Nama publisher |
| volume | INT | 11 | Versi Volume journal |
| page | VARCHAR | 11 | Halaman journal |
| number | INT | 11 | Journal number |
| year | INT | 11 | Tahun terbit journal |
| month | INT | 11 | Bulan terbit journal |
| day | INT | 11 | Tanggal Journal |

| | | | |
|---|---|---|---|
| row_status | INT | 2 | Status journal jika 1 = Aktif dan 0 = tidak aktfi |

*Table 4: Table journal*

Aplikasi ini lah yang akan digunakan sebagai *web server* yang membuat *resource - resource* itu dapat didistribusikan kepada *platform* lain. Beberapa *resource* yang dapat digunakan pada aplikasi ini diantaranya adalah memberikan seluruh data journal, mencari data journal, melihat isi journal, menyimpan data journal dan mengedit data journal.

*Resource – resouce* tersebut terbentuk dalam sebuah URI yang berisi data journal bertipe JSON. Request URI yang dapat digunakan dan sesuai standar HTTP diantara lain yaitu:

| Method | URL | Deskripsi |
|---|---|---|
| GET | {{host}}/journals<br>Parameter:<br>- page (page yang akan digunakan)<br>- limit (batas data yang akan dikeluarkan) | Untuk menampilkan seluruh journal |
| GET | {{host}}/journals/search/{{keyword}} | Untuk pencarian data journal |
| GET | {{host}}/journals/detail/{{id_journal}} | Untuk menampilkan detail pada journal |
| POST | {{host}}/journals/add<br><br>Parameter:<br>title (string), author (string), abstract (text), keywords (string), image (string \| url), file (string \| url), issn (string), publisher (string), volume (integer), page (integer), number (integer), year (integer), month (integer), day (integer), | Untuk meyimpan data journal |
| PUT | {{host}}/journals/edit/{{id_journal}}<br><br>Parameter: | Untuk mengedit data journal |

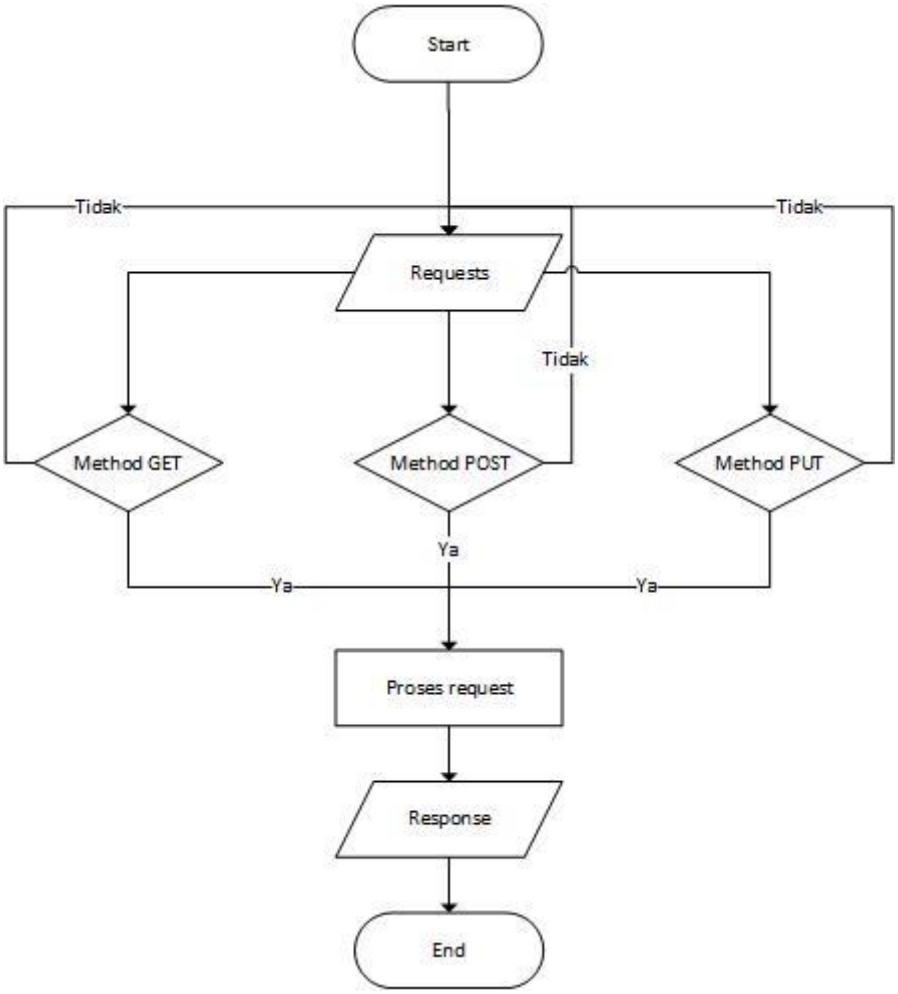
| | | |
|---|---|---|
| | title (string), author (string), abstract (text), keywords (string), image (string \| url), file (string \| url), issn (string), publisher (string), volume (integer), page (integer), number (integer), year (integer), month (integer), day (integer), | |
| Keterangan:<br>{{host}} adalah variable untuk memasukan hostname pada aplikasi berupa base URL.<br>{{keyword}} adalah variable yang akan di isi sebagai keyword untuk pencarian data<br>{{id_journal}} adalah variable yang berisi ID Journal pada database dengan type data *integer\|number* | | |

*Table 5: URL yang dapat digunakan pada aplikasi. Dimana host yang digunakan pada aplikasi RESTful tersebut*

Table tersebut yang akan digunakan sebagai parameter requests yang dikirimkan oleh user kepada web service.

Hasil – hasil requests tersebut akan di uji coba dan mendapatkan hasil berupa kecepatan waktu pada setiap multiplatform dari beberapa data yang ditampilkan pada webservice.

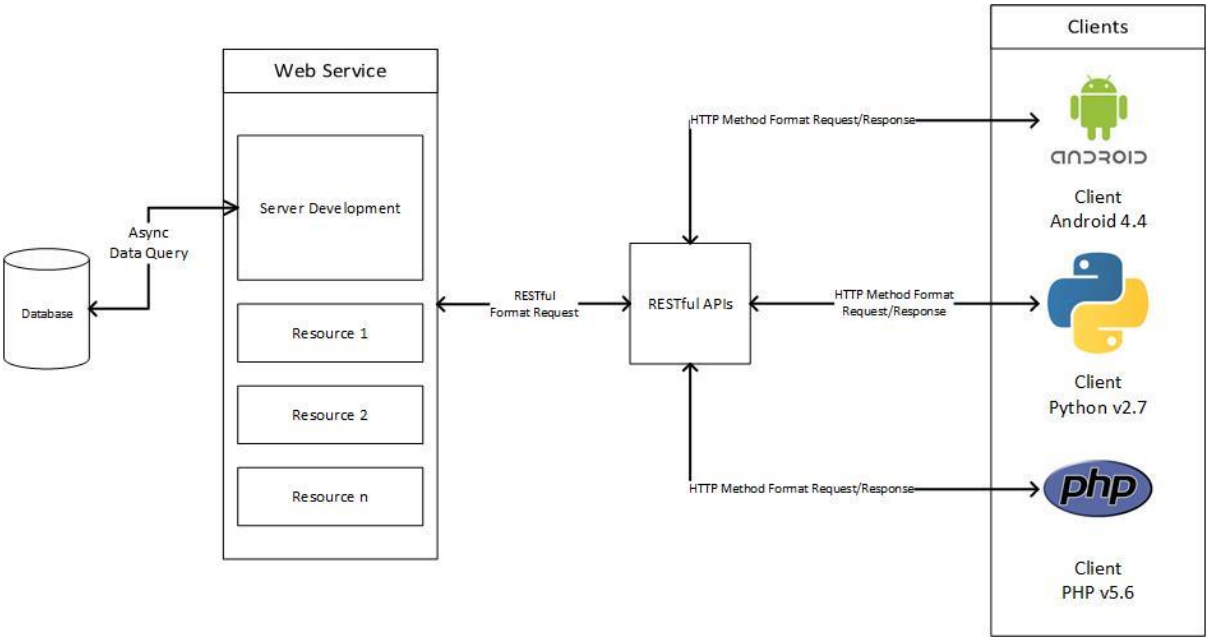
Proses aplikasi ini dimulai dari request user kepada aplikasi dan di cek apakah request tersebut berasal dari *HTTP Method* GET, POST atau PUT. Setelah request tersebut dapat di identifikasi sebagai salah satu *HTTP Method* maka akan diproses untuk menampilkan *response resource* yang sudah direquest. Dibawah ini adalah gambaran flowchart bagaimana aplikasi *web service* ini dapat bekerja.




*Gambar 12: Flowchart aplikasi webservice.*

Dari gambar flowchart tersebut menggambarkan bahwa ada penginputan *requests*, *requests* tersebut adalah *URL* yang dikirimkan user kepada aplikasi sehingga bisa didefiniskan apakah *HTTP Method* yang dikirimkan tersebut berupa GET, POST atau PUT. Jika sudah terdefinisikan maka akan diproses *request* yang memuat *resources* tersebut menjadi *response* yang akan diterima oleh user.

Dengan lebih jelasnya tentang interaksi antara webservice kepada multiplatform lain dapat dilihat pada gambar dibawah ini:




*Gambar 13: Interaksi web service kepada users/clients*

Dari gambar 14 menggambarkan bagaimana client dapat menggunakan *web service*. Pada konten clients terdapat 3 aplikasi untuk pengujian web service ini. Antara lain yaitu: Android v4.4, Python v2.7 dan PHP v5.6. Client tersebut dapat melakukan request kepada RESTful lalu akan dikirimkan kembali kepada development web service untuk mendapatkan resource-resource yang nantinya akan dikembalikan kembali kepada client. Tentunya resource-resource memiliki informasi atau penjelasan pada resource. Informasi tersebut didapatkan dari database lalu diolah kembali ke server development menjadi resource yang berisi informasi yang dibutuhkan oleh client. Webservice akan mengembalikan kembali kepada ke client berupa response melalui jalur protokol HTTP dan jika resource sudah sampai ke client, client dapat menggunakan resource yang bermuat informasi tersebut sesuai kebutuhan aplikasi client yang mereka kembangkan.

## B. HASIL PENGEMBANGAN

Dalam fase ini memuat uraian logis dari proses pencapaian hasil pengembangan beserta hasil-hasil yang diperoleh dari pelaksanaanya secara nyata sesuai dengan perangkat, metode dan alat yang digunakan dalam memproleh hasil tersebut.

Dibawah ini adalah hasil dari pengembangan pada platform-platform yang akan diuji menggunakan client-server dan memiliki web server menggunakan NodeJS v4.4.7.

Hasil total rata – rata yang kecepatan data menggunakan rumus $v = \frac{s_{total}}{t_{total}}$

ketarangan:

v = rata – rata

s = data

t = waktu

1. PHP

i. Spesifikasi client

PHP Native version 5.6.14, Apache 2.0

Windows 8.1 Home Premium 64bit, AMD10, RAM 4GB, HDD 1T

ii. Hasil Output

| Method | Name | URI | HTTP Code/Status | Time (data/second) |
|---|---|---|---|---|
| GET | Mengambil semua data journal | {{host}}/journals | 200 / Success | 100/0.0374 |
| GET | Mengambil semua data journal | {{host}}/journals | 200 / Success | 100/0.0341 |
| GET | Mengambil detail journal | {{host}}/detail/50 | 200 / Success | 1/0.0090 |
| GET | Mencari data journal | {{host}}/search/univerisity | 200 / Success | 4/0.0155 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.3538 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.1509 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.1809 |
| PUT | Mengedit data journal | {{host}}/journals/edit/112 | 200 / Success | 1/0.0909 |
| PUT | Mengedit data journal | {{host}}/journals/edit/111 | 200 / Success | 1/0.0775 |
| PUT | Mengedit data journal | {{host}}/journals/edit/66 | 200 / Success | 1/0.0810 |
| | | | Total | 211/1.0310 |

*Table 6: Tabel hasil output pada setiap HTTP Method menggunakan PHP*

Hasil output pada tabel 6 menampilkan hasil waktu yang dibutuhkan pada request setiap *resources*. Waktu yang dibutuhkan pada setiap *requests* didapatkan dari nilai selisih waktu pertama kali client merequest sampai client mendapatkan hasil *resources* tersebut berdasarkan total data pada setiap *requests*. Sehingga dapat dirumuskan $v = t2 - t1$ dimana *v* sebagai nilai selisih, *t2* sebagai waktu akhir dan *t1* sebagai waktu awal.

iii. Rata – Rata pada setiap Method HTTP

| Method | Time (data/second) |
|---|---|
| GET | 51/0.0240 |
| POST | 1/0.2285 |
| PUT | 1/0.0831 |

*Table 7: Table rata - rata waktu pada setiap HTTP Method*

Tabel rata – rata waktu didapat dengan rumus $v = \frac{s_{total}}{t_{total}}$ dimana *s* sebagai total data pada setiap *requests* dan *t* sebagai total setiap selisih waktu yang dibutuhkan pada setiap *requests*.

2. Python

i. Spesifikasi client

Python 2.7, Flask 0.11

Windows 8.1 Home Premium 64bit, AMD10, RAM 4GB, HDD 1T

ii. Hasil output

| **Method** | **Name** | **URI** | **HTTP Code/Status** | **Time (data/second)** |
|---|---|---|---|---|
| GET | Mengambil semua data journal | {{host}}/journals | 200 / Success | 100/0.0174 |
| GET | Mengambil semua data journal | {{host}}/journals | 200 / Success | 100/0.0024 |
| GET | Mengambil detail journal | {{host}}/detail/50 | 200 / Success | 1/0.0042 |
| GET | Mencari data journal | {{host}}/search/univerisity | 200 / Success | 4/0.0027 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.0035 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.0343 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.0016 |
| PUT | Mengedit data journal | {{host}}/journals/edit/112 | 200 / Success | 1/0.0020 |
| PUT | Mengedit data journal | {{host}}/journals/edit/111 | 200 / Success | 1/0.0015 |
| PUT | Mengedit data journal | {{host}}/journals/edit/66 | 200 / Success | 1/0.0118 |
| | | | Total | 211/0.0814 |

*Table 8: Tabel hasil output pada setiap HTTP Method menggunakan Python*

Hasil output pada tabel 6 menampilkan hasil waktu yang dibutuhkan pada request setiap *resources*. Waktu yang dibutuhkan pada setiap *requests* didapatkan dari nilai selisih waktu pertama kali client merequest sampai client mendapatkan hasil *resources* tersebut berdasarkan total data pada setiap *requests*. Sehingga dapat dirumuskan $v = t2 - t1$ dimana *v* sebagai nilai selisih, *t2* sebagai waktu akhir dan *t1* sebagai waktu awal.

iii. Hasil output pada setiap Method HTTP

| **Method** | **Time (data/second)** |
|---|---|
| GET | 51/0.0067 |
| POST | 1/0.0131 |
| PUT | 1/0.0051 |

*Table 9: Table rata - rata waktu pada setiap HTTP Method*

3. Android

i. Spesifikasi client

Kitkat v4.4.4, Kernel version 3.4.0

Sony Xperia Z1, RAM 1GB, Memory Internal 12GB

ii. Hasil output

| Method | Name | URI | HTTP Code/Status | Time (data/second) |
|---|---|---|---|---|
| GET | Mengambil semua data journal | {{host}}/journals | 200 / Success | 100/0.0201 |
| GET | Mengambil semua data journal | {{host}}/journals | 200 / Success | 100/0.0211 |
| GET | Mengambil detail journal | {{host}}/detail/50 | 200 / Success | 1/0.0192 |
| GET | Mencari data journal | {{host}}/search/univerisity | 200 / Success | 4/0.0148 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.0139 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.0265 |
| POST | Menginput data journal | {{host}}/journals/add | 200 / Success | 1/0.0257 |
| PUT | Mengedit data journal | {{host}}/journals/edit/112 | 200 / Success | 1/0.0329 |
| PUT | Mengedit data journal | {{host}}/journals/edit/111 | 200 / Success | 1/0.1095 |
| PUT | Mengedit data journal | {{host}}/journals/edit/66 | 200 / Success | 1/0.0272 |
| | | | Total | 211/0.3109 |

*Table 10:Tabel hasil output pada setiap HTTP Method menggunakan Android*

Hasil output pada tabel 6 menampilkan hasil waktu yang dibutuhkan pada request setiap *resources*. Waktu yang dibutuhkan pada setiap *requests* didapatkan dari nilai selisih waktu pertama kali client merequest sampai client mendapatkan hasil *resources* tersebut berdasarkan total data pada setiap *requests*. Sehingga dapat dirumuskan $v = t2 - t1$ dimana *v* sebagai nilai selisih, *t2* sebagai waktu akhir dan *t1* sebagai waktu awal.

iii. Hasil output pada setiap Method HTTP

| Method | Time (data/second) |
|---|---|
| GET | 51/0.0188 |
| POST | 1/0.0220 |
| PUT | 1/0.0565 |

*Table 11: Table rata - rata waktu pada setiap HTTP Method*

## C. PEMBAHASAN

Fase ini merupakan fase hasil pembahasan dari pengujian web service berbasis RESTful API. Proses pengujian ini dilakukan secara berkala dengan mengelompokan terlebih dahulu metode HTTP

yang digunakan diantaranya GET, POST dan PUT. Lalu diuji pada 3 platform menggunakan rata-rata kecepatan waktu pada setiap pertama kali *requests* sampai menghasilkan *response* pada client hingga mendapatkan *response HTTP Code 200* yang menunjukan bahwa *web service* berhasil memberikan *response* kepada client. Dan dari hasil pengembangan tersebut maka didapatkan hasil analisis yang dapat menyimpulkan pemecahan masalah. Hasil uji webservice dapat dilihat pada table 12.

| **Fungsional Resource** | **Method HTTP** | **Platform** | | |
|---|---|---|---|---|
| | | **PHP** | **Python** | **Android** |
| Menampilkan seluruh data journal | GET | Sukses | Sukses | Sukses |
| Pencarian data journal | GET | Sukses | Sukses | Sukses |
| Menampilkan detail journal | GET | Sukses | Sukses | Sukses |
| Menyimpan data journal | POST | Sukses | Sukses | Sukses |
| Mengubah data journal | PUT | Sukses | Sukses | Sukses |

*Table 12: Hasil uji webservice dengan status 200/success.*

**1. Output web service dengan format JSON**

Output pada client berupa file json dimana data tersebut dapat diolah untuk kepentingan masing – masing aplikasi yang mereka buat. Contoh potongan output file json yang diberikan oleh web service dengan menggunakan *resource* detail journal.


```
{
  "status": {
    "code": 200,
    "message": "Success",
    "description": "",
    "time": "0.003ms"
  },
  "data": {
    "journals": {
      "id": 80,
      "title": "Measuring the Performance of Autoregressive Integrated Moving
Average and Vector Autoregressive Models in Forecasting Inflation Rate in Rwanda ",
      "subtitle": null,
      "author": "Joselyne INGABIRE, Dr. Joseph K. Mung'atu",
      "book_type": "text-book",
      "description": "The aim of this study is to test and distinguish which of
ARIMA and VAR models performs best in forecasting inflation in Rwanda. In order to
fulfil this objective observed quarterly data from 2000Q1 to 2015Q1 on economic
variables such as the Consumer Price Index, ",
      "image": "",
      "page": null,
      "issn": "2348-5736",
      "publisher": "International journal of mathematics and physical sciences
research",
      "volume": 4,
      "year": 2016,
      "month": 4,
      "date": 0,
```

```
    "row_status": 1
  }
}
```

**2. Rata – rata waktu yang dibutuhkan client**

Dengan menggunakan rumus kecepatan rata – rata waktu tempuh, maka didapatkan hasil yang dibutuhkan pada setiap client saat melakukan *request* hingga mendapatkan sebuah *response* dari *web services.*

| | GET | POST | PUT |
|---|---|---|---|
| **Python** | 0.0067 | 0.0131 | 0.0051 |
| **PHP** | 0.0240 | 0.2285 | 0.0831 |
| **Android** | 0.0188 | 0.0220 | 0.0565 |

*Table 13: Hasil uji rata-rata kecepatan waktu yang dibutuhkan untuk me-request sebuah resource*

Table 13 diatas didapatkan dari hasil rata – rata pada setiap method HTTP yang digunakan oleh client. Dimana hasil dari rata – rata tersebut menunjukan bahwa client yang menggunakan platform *Python* nilainya lebih rendah yang berarti Python menjadi yang lebih cepat untuk mengakses web service berbasis *RESTful APIs* ini di antara platform yang lain.

## D. PENGUJIAN SISTEM

Pada penelitian ini pengujian sistem dilakukan dengan menggunakan pengujian white box dan black box. Pengujian bertujuan untuk mencari kesalahan. Pengujian yang baik adalah pengujian yang memiliki kemungkinan besar dalam menemukan kesalahan. (Roger S. Pressman, 2010, p.584). Dalam penelitian ini pengujian dengan whitebox dan blackbox tidak mempengaruhi hasil akhir dari penelitian karena pengujian dilakukan terhadap software yang telah dibuat.

**3. Pengujian Whitebox**

Whitebox testing yang digunakan untuk merancang test case adalah Cyclomatic Complexity. Cyclomatic Complexity merupakan suatu sistem pengukuran yang menyediakan ukuran kuantitatif dari kompleksitas logika suatu program. Pada Basis Path Testing, hasil dari cyclomatic complexity digunakan untuk menentukan banyaknya independent paths. Independent path adalah sebuah kondisi pada program yang menghubungkan node awal dengan node akhir. Berikut rumus matematika yang akan dibutuhkan.

$$v(G) = E - N + 2$$

Menggunakan grafik aliran ini, kita dapat menghitung jumlah jalur independen melalui kode. Kami melakukan ini dengan menggunakan metrik disebut nomor cyclomatic (McCabe, 1976), yang didasarkan pada teori grafik.

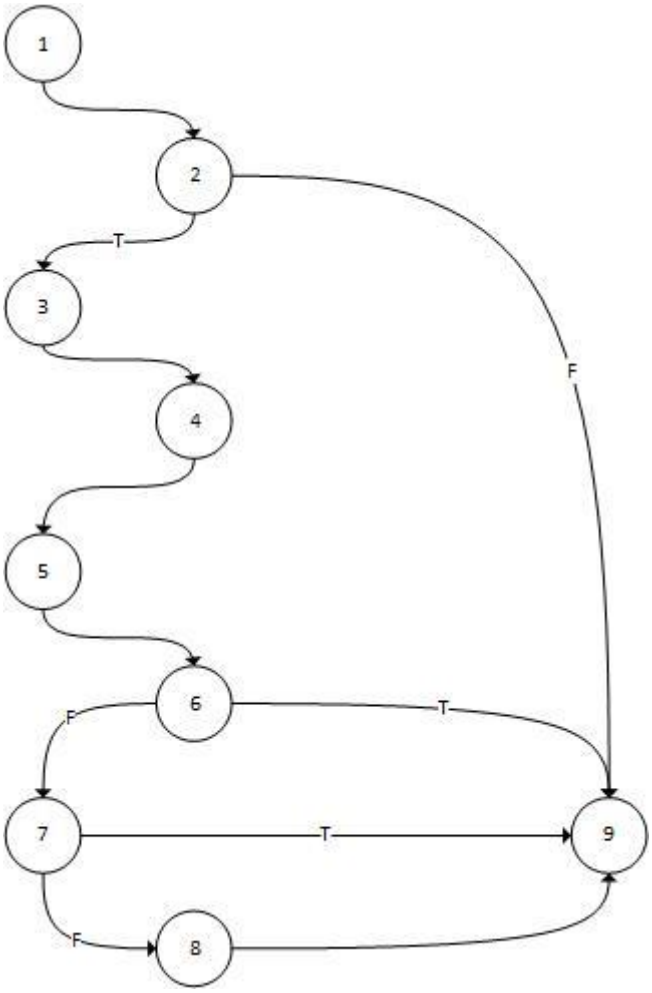
**a. Langkah pertama**

Prosedur dibawah ini menunjukan bagaimana laporan algoritma dipetakan ke node grafik, nomor di samping kiri.

```
 exports.detail = function( req, res ) {
 1.  var id = req.params.id;
 2.  if ( validation.checkInt( id ) ) {
 3.      var conditions = [ 'journal.id', 'journal.row_status' ];
 4.      var query = modelInstance.getDetail( conditions, null );
 5.      db.query( query , [id, 1], function(err,rows) {
 6.          if( err )
                    data = { error: err }; code = 422;
 7.          if( rows && rows.length > 0 )
                    data = { journal: rows[0] }; code = 200;
 8.          else data 'Data is not found'; code = 404
         });
     } else {
             data = { error: 'ID must integer' }; code = 422;

     }
 9.  return response( res, data, code );
}
```

Dari script diatas dapat dibuatkan node – node pada setiap logikal aplikasi sehingga membentuk flowchart. Hasil node diagram ini akan digunakan untuk menghitung nilai kompleksitas *cyclomatic* dari grafik aliran.




*Gambar 14: Graph flowchart dari fungsional pengambilan detail journal.*

Dari grafik alir yang telah dibuat sebelumnya, maka didapat hubungan bobot antar node. Hubungan bobot diberi nilai 1 jika terdapat hubungan antara node satu dengan yang lain, dan akan bernilai 0 jika tidak ada hubungan antar node.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 9 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

*Table 14: Nilai bobot pada node.*

Diketahui : Koneksi = Jumlah bobot per simpul – 1

Didapat : Simpul 1 = 1-1 = 0

Simpul 2 = 2-1 = 0

Simpul 3 = 1-1 = 1

Simpul 4 = 1-1 = 0

Simpul 5 = 1-1 = 0

Simpul 6 = 2-1 = 0

Simpul 7 = 2-1 = 1

Simpul 8 = 1-1 = 1

Simpul 9 = 0-1 = -1

Maka dapat dihitung nilai cyclomatic complexity nya dari grafik hubungan bobot berdasarkan jumlah total dari nilai koneksi yang didapat dari masing-masing simpul yaitu:

Cyclomatic Complexity = Jumlah koneksi simpul + 1

= 3 + 1

= 4

**b. Langkah kedua**

Menentukan kompleksitas *cyclomatic* dari grafik aliran.

```
v(G) = E – N + 2
     = 11 – 9 + 2
     = 4
```

Keterangan:
E = Jumlah busur atau link
N = Jumlah simpul

**c. Langkah ketiga**

Menentukan dasar jalur independen. Jalur independen pada flow graph atau node yang akan diuji adalah sebanyak 4 jalur. Berdasarkan urutan alur flow graph diatas, didapat kelompok basis flow graph sebagai berikut:

Path 1: 1 – 2 – 3 – 4 – 5 – 6 – 9

Path 2: 1 – 2 – 3 – 4 – 5 – 6 - 7 – 9

Path 3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9

Path 4: 1 – 2 – 9

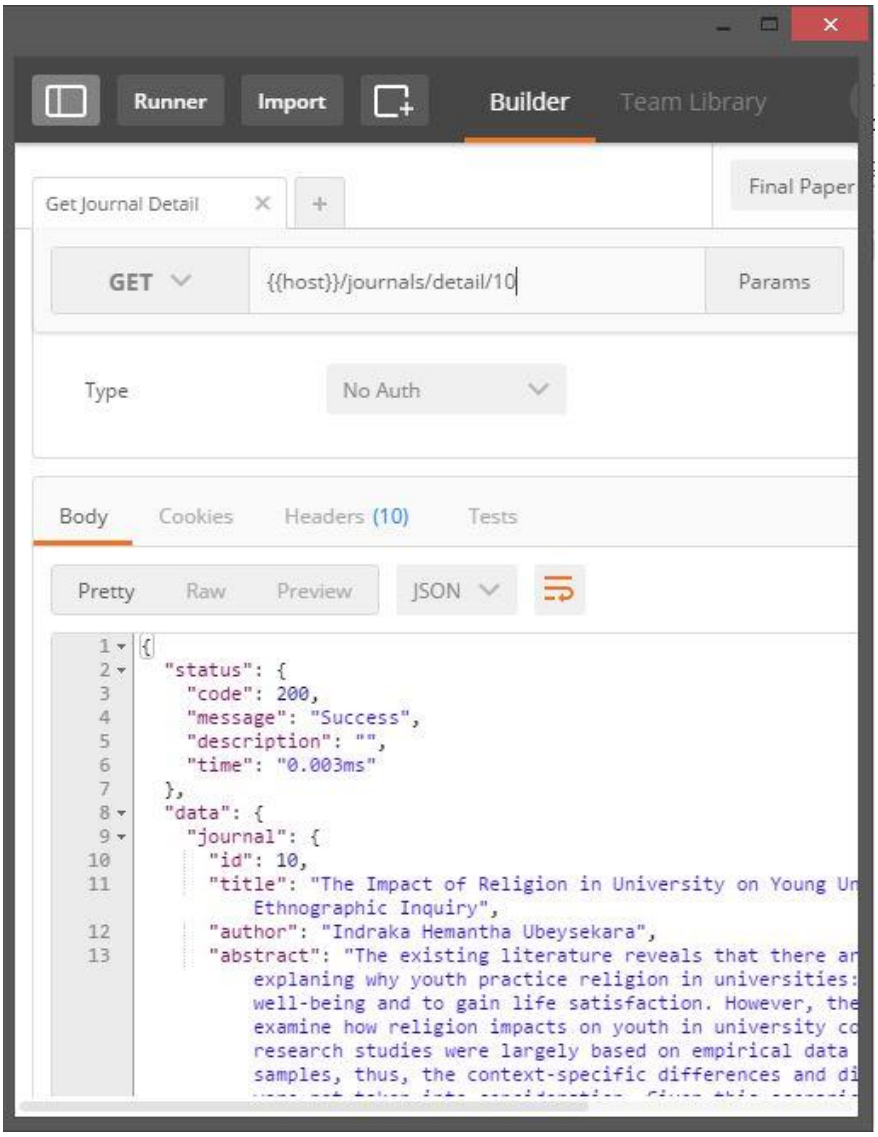
### 4. Pengujian Blackbox

Pengujian selanjutnya adalah pengujian black box, yang dilakukan untuk menguji aplikasi dari sisi validitas parameter setiap tahapan kegiatan yang dilakukan oleh paltform – platform terhadap web service data journal. Adapun tahapan pengujian black box dapat dilihat pada tabel pengujian black box.

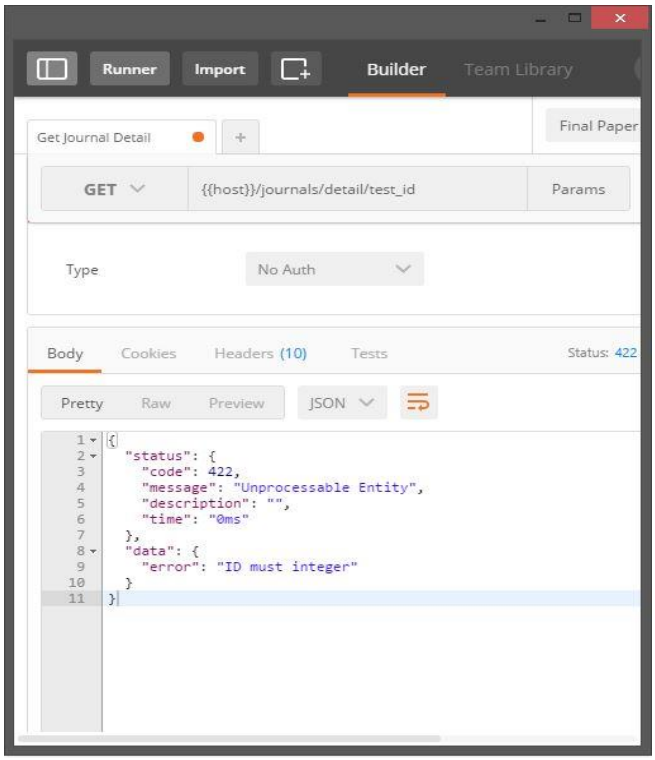
| Web service : Detail journal<br>URI: host/journals/detail/{journal_id} | | | | | |
|---|---|---|---|---|---|
| **No** | **Skenario pengujian** | **Test case parameter** | **Hasil yang diharapkan** | **Hasil pengujian** | **kesimpulan** |
| 1 | Merequest web service dengan mengisi id journal | journal_id: 10 | Menampilkan detail journal berdasarkan id 10 | Sesuai harapan | Valid |

| | | | | | |
|---|---|---|---|---|---|
| 2 | Merequest web service dengan journal id-nya bukan integer | Journal_id: test_id | Memberikan http code 422 dan menampilkan message error “ID Must Integer” | Sesuai harapan | valid |
| 3 | Merequest web service dengan journal id tidak ada di database | Journal_id: 123123 | Memberikan HTTP Code 404 dan menampilkan message Not Found | Sesuai harapan | valid |

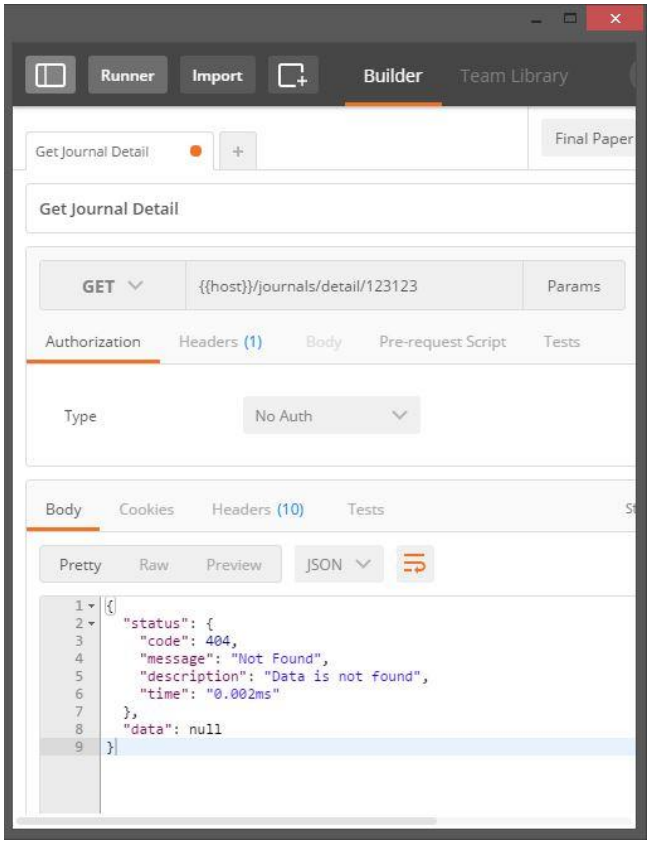
*Table 15: Hasil pengujian blackbox.*




*Gambar 15: Hasil output pengujian langkah pertama*




*Gambar 16: Pengujian blackbox langkah kedua*




*Gambar 17: Pengujian blackbox langkah ketiga*

# BAB V

## KESIMPULAN dan SARAN

### A. KESIMPULAN

Dari uraian dan pembahasan "*Penerapan Resource Oriented Architecture Untuk Pendistribusian Data Menggunakan RESTful APIs Berbasis Multiplatform*" maka dapat diambil kesimpulan sebagai berikut :

1. Pendistribusian menggunakan metode ROA dapat berjalan dengan baik di setiap platform diantaranya adalah Python, PHP dan Android.
2. Aplikasi yang tercepat diantara multiplatform lainnya yaitu multiplatform berbasis *Python* dengan nilai rata – rata 0.0067 (GET), 0.0131 (POST), 0.0051 (PUT).

### B. SARAN

Dalam penggunaan dan pengimplementasian pada aplikasi agar lebih sempurna dalam tampilan dan penggunaanya, maka diberikan saran sebagai berikut :

1. Untuk mengakses web service ini diharapkan menggunakan metode *Tokenize* sehingga keamanan pada setiap *resource* dapat terjaga.
2. Metode lain yang dapat digunakan sebagai pendistribusian data adalah Service Oriented Architecture
3. Untuk pengumpulan data sebaiknya menggunakan data yang valid dari setiap perpustakaan universitas sehingga data tersebut dapat terjamin keasliannya.

[ Halaman ini sengaja dikosongkan

# DAFTAR PUSTAKA


Aspects of the design of distributed databases. (2011). *Journal of Knowledge Management, Economics and Information Technology*.

Benjamin San Souci, M. L. (2014). An Inside Look at the Architecture of NodeJS. *McGill University*.

Dahl, R. (2013, December 13). About node. (D. Synodinos, Interviewer)

Danks, S. (2011, 11). The ADDIE Model. *ASQ The Global Voice of Quality, IV*.

Fielding, R. T. (2000). Architectures, Architectural Styles and the Design of Network-based Software. *University of California*.

Henning Heitk¨otter, S. H. (2012, April). Evaluating Cross-Platform Development Approaches for Mobile Applications. *Springer Berlin Heidelberg, 140*, 120-138.

json.org. (2000). *JSON is a lightweight data-interchange format. It is easy for humans to read and write.* Retrieved from JSON (JavaScript Object Notation): http://www.json.org/json-id.html

Leonard Richardson, S. R. (2007). *RESTful Web Services.* (M. Loukides, Ed.) United States of America: O'Reilly Media.

Letaifa, N. (2011, April 10). *Zenika is a firm specializing in Computer Architecture and Agile methods.* Retrieved from Zenika: http://blog.zenika.com/2011/04/10/nodejs/

Orsini, L. (2013, November 7). *ReadWrite is now one of the most widely read and respected tech news sites in the world.* Retrieved from ReadWrite: http://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs/

R. Fähnrich, K. F. (2009). Proceedings First International Symposium on Services Science ISSS'09. *REST and Resource-Oriented Architecture*, 161.

Redda, Y. A. (2012, Juni). Cross platform Mobile Applications Development. *Norwegian University of Science and Technology*.

Ritesh Sinha, M. K. (2014, 8 7). Design & Development of a REST based Web Service Platform for Applications Integration on Cloud. *IJISET - International Journal of Innovative Science, Engineering & Technology, 1*, 387.

Robbins, R. J. (1995). Database Fundamentals. In R. J. Robbins, *Database Fundamentals.* Johns Hopkins University.

Roberto Lucchi, M. M. (2008). Resource Oriented Architecture and Rest – Assesment of impact and advantage on INSPIRE. *JRC Scientific and Technical Reports*.

Rouse, M. (2015, January 23). *This definition is part of our Essential Guide: Relational database management system guide: RDBMS still on top*. Retrieved from database management system (DBMS): http://searchsqlserver.techtarget.com/definition/database-management-system

Saputra, E. Y. (2015, July 30). *ekajogja is an Indonesian web designer-developer, WordPress consultant, who loves working on software localization & documentation.* Retrieved from ekajogja: http://ekajogja.com/definisi/rest-representational-state-transfer/

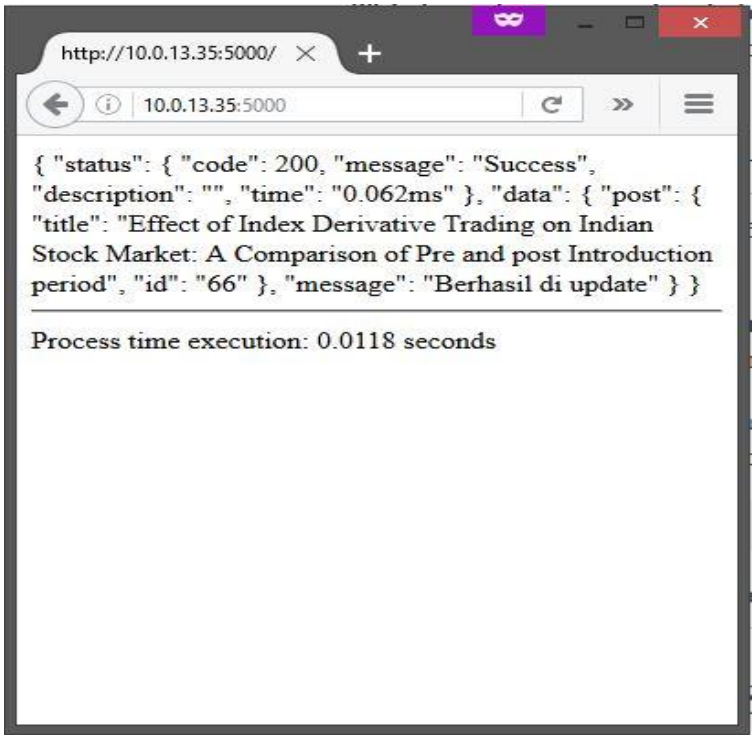
W3C. (2003, March). *The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards.* (D. Booth, Editor) Retrieved from Web Services Architecture: https://www.w3.org/TR/2003/WD-ws-arch-20030808/

Xamarin. (2014, Juni). *Setting Up A Xamarin Cross Platform Solution*. Retrieved from We officially open sourced the Xamarin SDK for Android, iOS, and Mac under the same MIT license used for the Mono project. https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_3_-_setting_up_a_xamarin_cross_platform_solution/

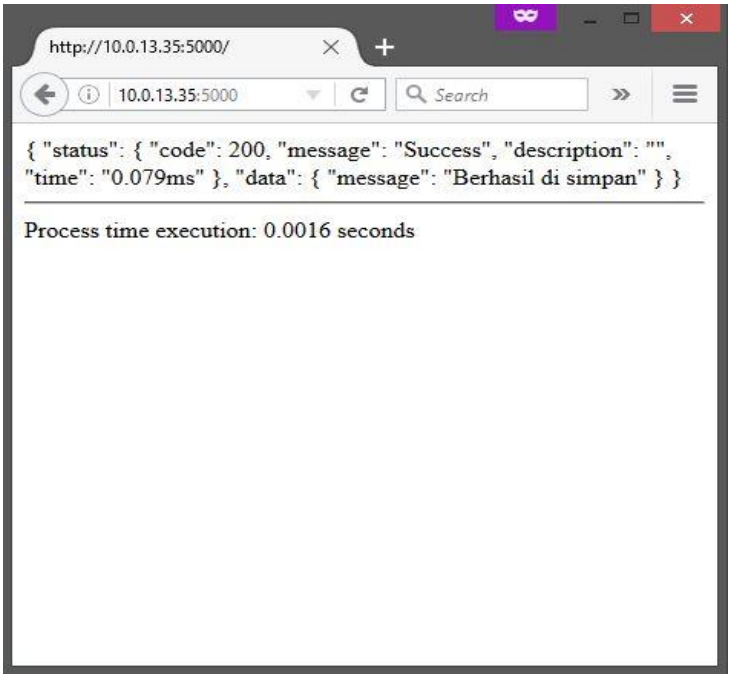
# LAMPIRAN

## A. HASIL OUTPUT

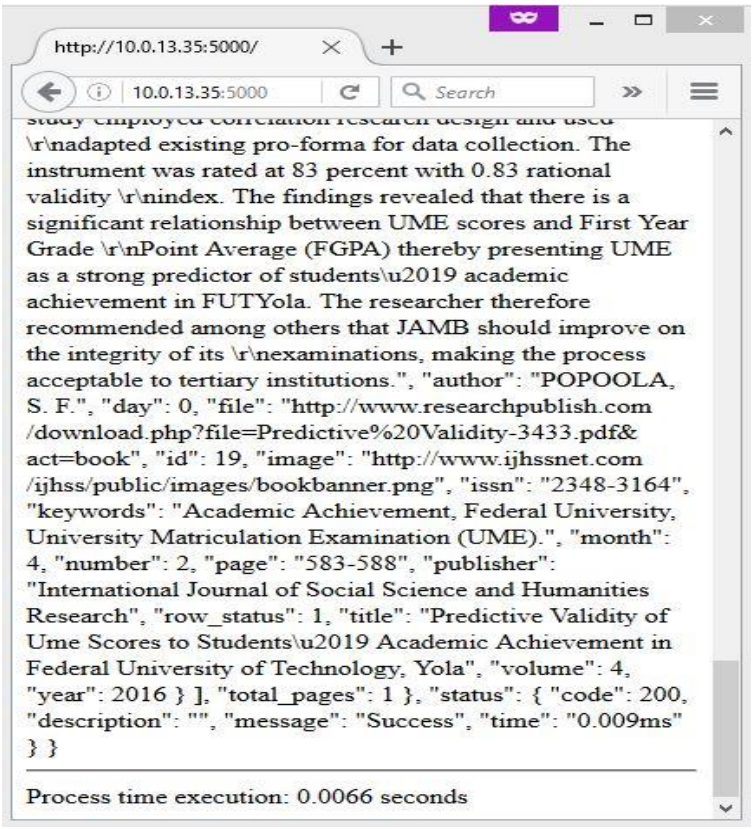
1. PYTHON



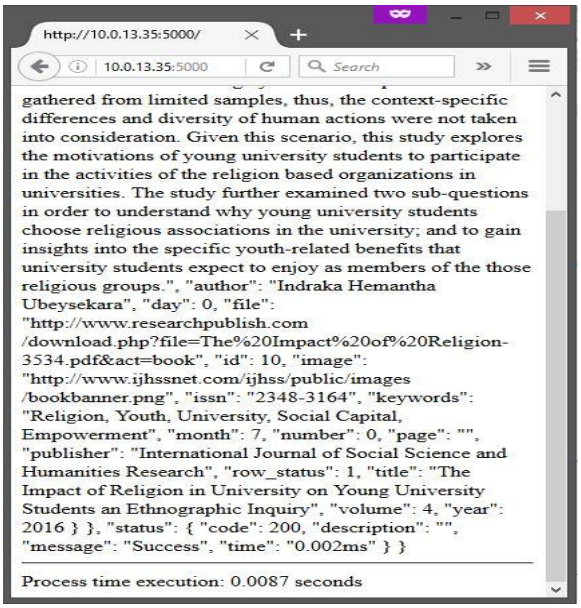

*Gambar 18: Python merequest edit journal dengan dimana resource tersebut adalah /journals/edit/(journal_id) yang menggunakan method HTTP PUT*




*Gambar 19: : Python merequest menambahkan journal dengan dimana resource tersebut adalah /journals/add yang menggunakan method HTTP POST*



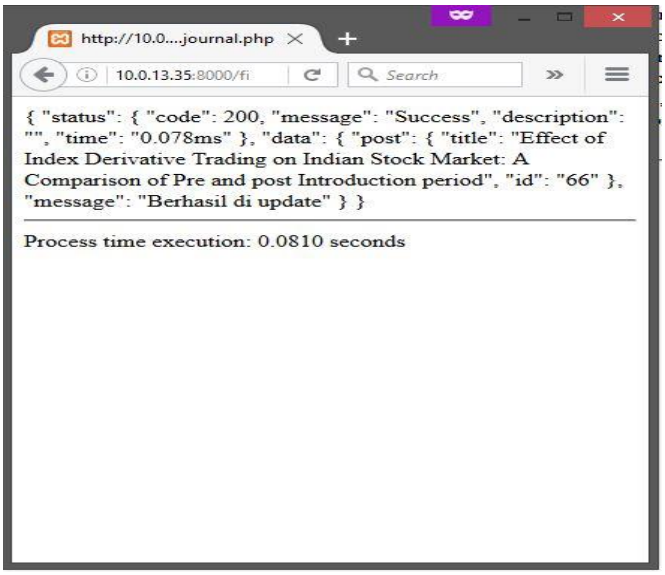

*Gambar 20: Python merequest pencarian journal dengan dimana resource tersebut adalah /journals/search/a yang menggunakan method HTTP GET*



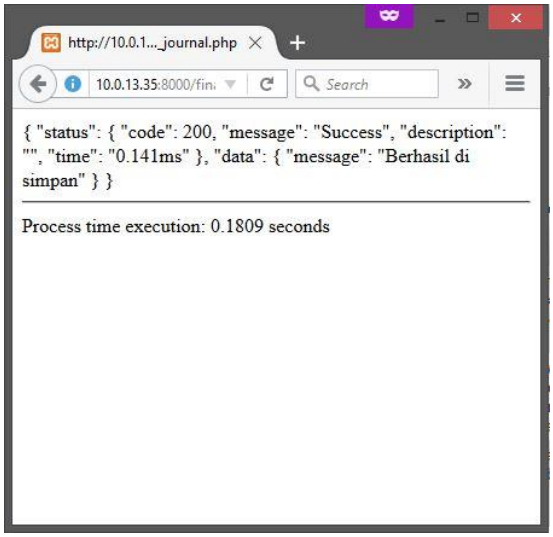

*Gambar 21: Python merequest detail journal dengan dimana resource tersebut adalah /journals/edit/(journal_id) yang menggunakan method HTTP GET*


*Gambar 22: Python merequest data journal dengan dimana resource tersebut adalah /journals yang menggunakan method HTTP GET*

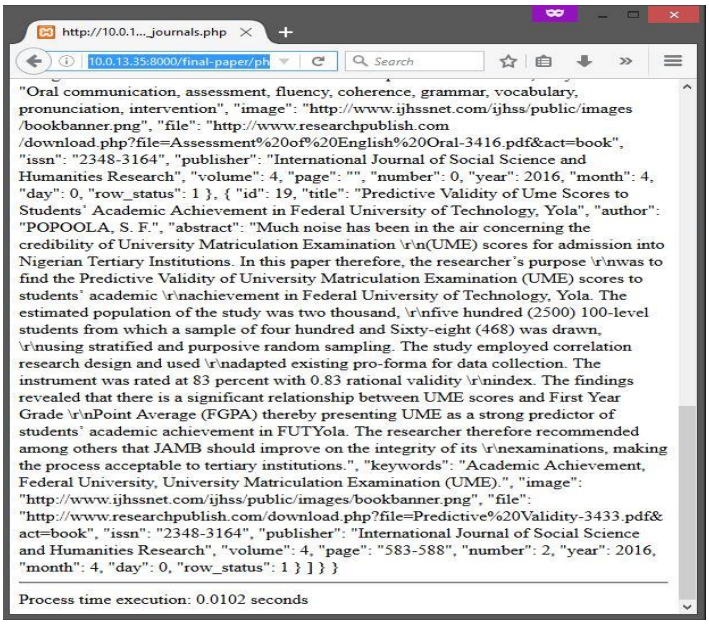
2. PHP




*Gambar 23: PHP merequest edit journal dengan dimana resource tersebut adalah /journals/edit/(journal_id) yang menggunakan method PUT*




*Gambar 24: PHP merequest tambah journal dengan dimana resource tersebut adalah /journals/add yang menggunakan method POST*

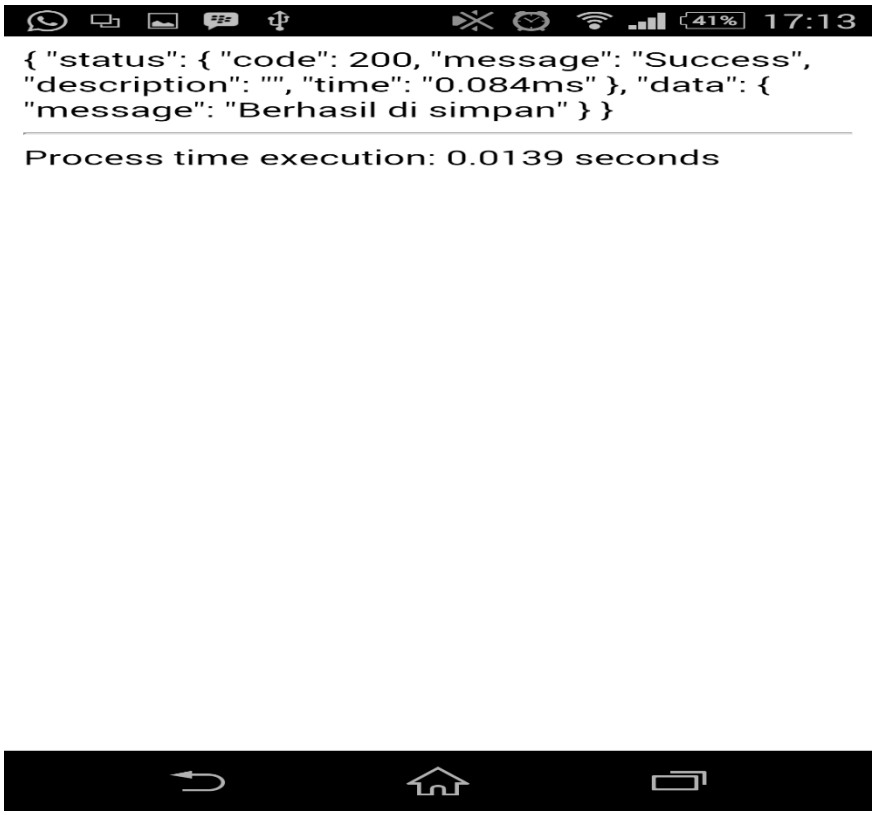



*Gambar 25: PHP merequest pencarian journal dengan dimana resource tersebut adalah /journals/search/a*


*Gambar 26: PHP merequest detail journal dengan dimana resource tersebut adalah /journals/detail/(journal_id) yang menggunakan method GET*

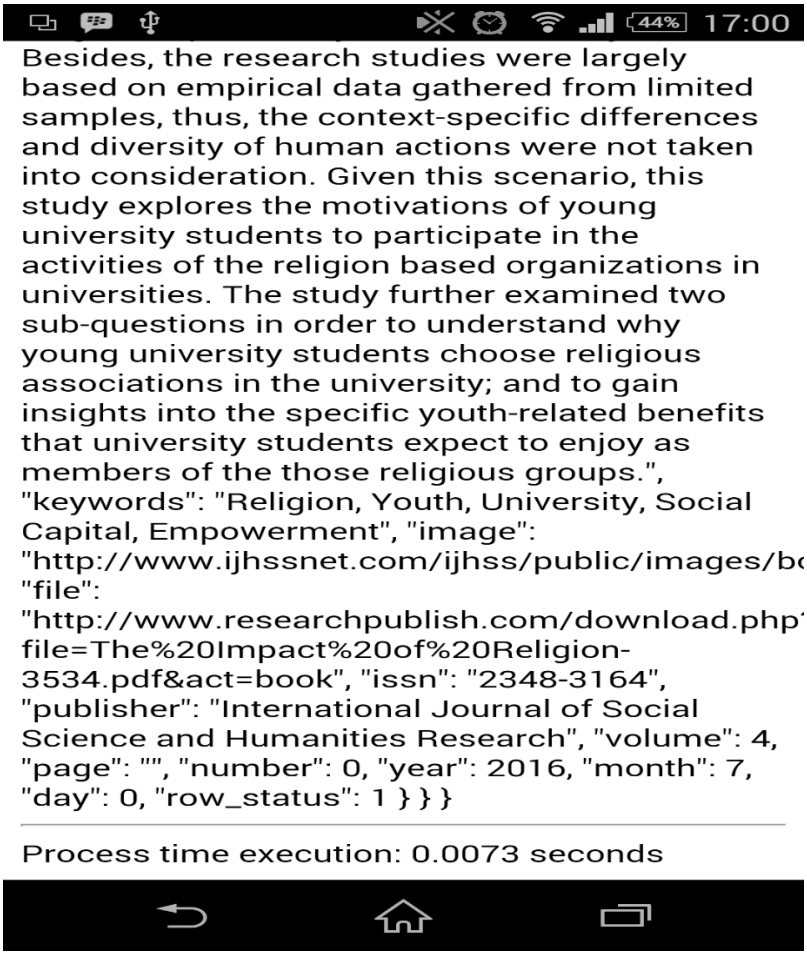

*Gambar 27: PHP merequest pencarian journal dengan dimana resource tersebut adalah /journals/search/a*

3. ANDROID

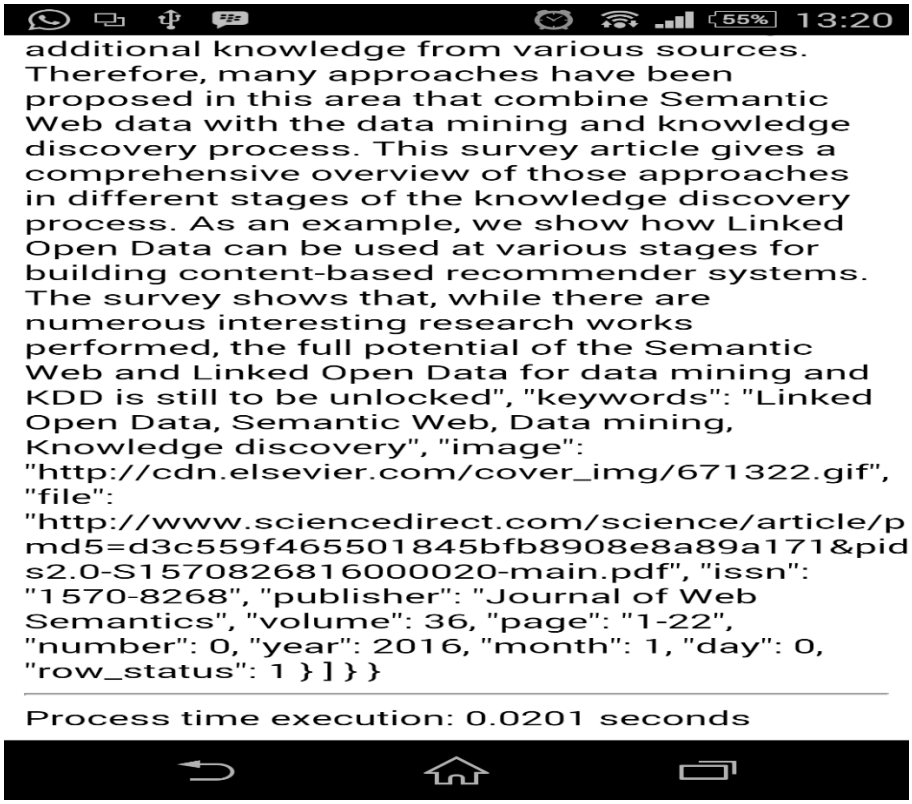



*Gambar 28: Android merequest menggunakan method POST*

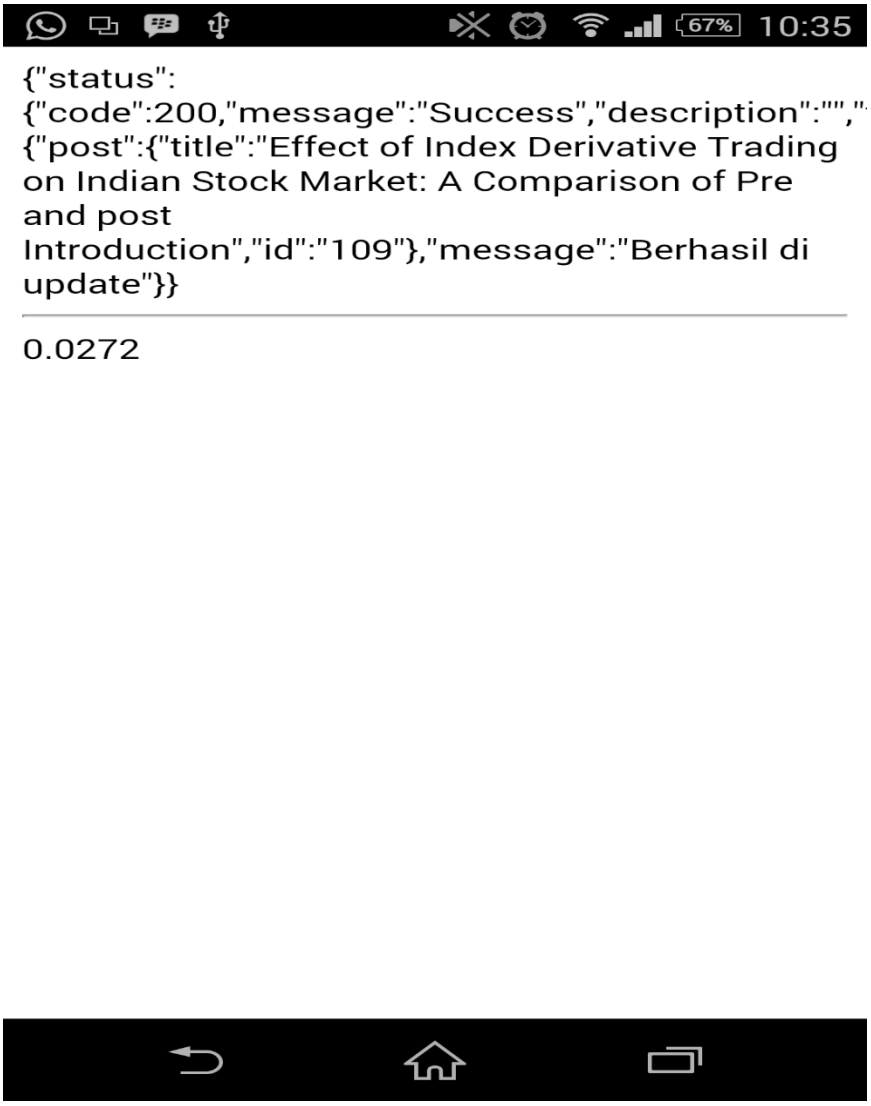

*Gambar 29: Andoird merequest resource pencarian data journal*




*Gambar 30: Android merequest detail journal*




*Gambar 31: Andoird merequest data journal*




*Gambar 32: Andoird mengupdate data journal menggunakan method PUT*

## B. SCRIPT

### 1. App.js

```
/**
 * Mendapatkan beberapa module untuk mendukung fungsi Web Service API
 * @return functions
 */
var express      = require( 'express' );
var http         = require( 'http' );
var path         = require( 'path' );
var mysql        = require( 'mysql' );
var bodyParser   = require( 'body-parser' );
var responseTime = require( 'response-time' )
var crypto       = require( 'crypto' );
var jwt          = require( 'jsonwebtoken' );

db              = require( './config/database.js' );
secretKey       = require( './config/secret-keys.js' );

start = new Date();
```

```
app   = express();

var users        = require( './controllers/users' );
var journals     = require( './controllers/journals' );
var publishers   = require( './controllers/publishers' );
var authenticate = require( './controllers/authenticate' );

merge      = require( 'merge' );
fn         = require( './services/global-functions.js' )();
buildQuery = require( './services/build-query.js' )();
md5        = crypto.createHash( 'md5' );

user_data  = null;
user_level = null;

app.use( bodyParser.urlencoded({ extended: true }) );
app.use( bodyParser.json() );
app.use( responseTime() );

app.use( function(req, res, next) {
        res.header( "Access-Control-Allow-Origin", "*" );
        res.header( "Access-Control-Allow-Headers", "Origin, X-Requested-
With, Content-Type, Accept" );
        res.header( 'Access-Control-Allow-Methods',
'GET,PUT,POST,DELETE,OPTIONS');
        res.header( 'Content-Type', 'text/json' );
        next();
});

app.use( function( req, res, next ) {
        start        = new Date()
        responsetime = (new Date() - start) / 1000;
        next()
});

app.get ( '/journals', journals.index );
app.get ( '/journals/:type', journals.index );
app.post( '/journals/add', journals.add );
app.put ( '/journals/edit/:id', journals.edit );
app.get ( '/journals/search/:query',journals.search );
app.get ( '/journals/detail/:id', journals.detail );
app.get ( '/publishers', publishers.index );
app.post( '/publishers/add', publishers.add );
app.put ( '/publishers/edit/:id', publishers.edit );
app.get ( '*', function(req, res){ fn.getResponse(res, 'Url is not found'
,404) });

app.set( 'port', process.env.PORT || 2000 );
http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port ' + app.get('port'));
});
```

2. Journals.js

```
var validation = require( '../services/validations' )();
var modelInstance = require( '../models/journals-model' );

var total_pages, conditions;
limit = 10;

exports.index = function( req, res ) {
```

```
        var  count,
                 total_pages,
                 conditions = 'journal.row_status = 1 ',
                 page   = 1,
                 offset = 0,
                 order  = 'journal.id DESC';

        if( req.query.limit != undefined ) {limit = req.query.limit;}

        db.query( modelInstance.getCount(conditions), function( err, rows
){
                count = rows[0].count;
                total_pages  =  Math.ceil( count/limit );
        } );

        if ( req.query.page )
        {
                page   = req.query.page - 1;
                offset = page*limit;
                page   += 1;
        }

        if ( req.query.sort )
        {
                order = req.query.sort
                if ( req.query.direction )
                {
                        order += ' ' + req.query.direction
                }
        }

        var options = {
                order: order,
                offset: offset,
                limit: limit
        }

    db.query( modelInstance.getList( conditions, options ), null,
function(err,rows) {

         if( err ) fn.getResponse( res, { error: err }, 422 );
        if( rows ) fn.getResponse( res, {
                count: count,
                total_pages: total_pages,
                current_page: page,
                total_rows: rows.length,
                journals: rows
        } );

    });
}

exports.detail = function( req, res ) {

        var id = req.params.id;
        if ( validation.checkInt( id ) ) {

                var conditions = [ 'journal.id', 'journal.row_status' ];
                var query = modelInstance.getDetail( conditions, null );

            db.query( query , [id, 1], function(err,rows) {
```

```
                if( err ) fn.getResponse( res, { error: err }, 422 );
                if( rows && rows.length > 0 ) fn.getResponse( res, {
journal: rows[0] } );
                else fn.getResponse( res, null, 404, 'Data is not found' );

            });
                return
        }
        fn.getResponse( res, { error: 'ID must integer' }, 422 );
}

exports.search = function( req, res ) {

        var  count,
                total_pages,
                conditions = 'journal.row_status = 1 ',
                page   = 1,
                offset = 0,
                order  = 'journal.id DESC';

        if ( req.params.query != undefined ) conditions += ' AND title LIKE
"%' + req.params.query + '%" ';

        if( req.params.type != undefined ) conditions += ' AND
journal.journal_type = "' + req.params.type + '" ';

        db.query( modelInstance.getCount(conditions), function( err, rows
){
                count = rows[0].count;
                total_pages  =  Math.ceil( count/limit );
        } );

        if ( req.params.page )
        {
                page   = req.params.page - 1;
                offset = page*limit;
                page   += 1;
        }

        if ( req.params.sort )
        {
                order = req.params.sort
                if ( req.params.direction )
                {
                        order += ' ' + req.params.direction
                }
        }

        var options = {
                order: order,
                offset: offset,
                limit: limit
        }

    db.query( modelInstance.getList( conditions, options ), null,
function(err,rows) {

        if( err ) fn.getResponse( res, { error: err }, 422 );
        if( rows ) fn.getResponse( res, {
                count: count,
```

```
                total_pages: total_pages,
                current_page: page,
                journals: rows
        } );

    });
}

exports.add = function( req, res ) {

        var post  = req.body;

        // delete post.token;
        db.query(
                buildQuery.insert( 'journals' ),
                post,
                function( err, rows ) {
                 if( err ) fn.getResponse( res, { error: err }, 400 );
                if( rows ) fn.getResponse( res, { 'message': 'Berhasil di
simpan' } );
                }
        )

}

exports.edit = function( req, res ) {

        var id = req.params.id;
        var post  = req.body;

        db.query(
                modelInstance.updateData( id ),
                post,
                function( err, rows ) {
                        post.id = id;
                 if( err ) fn.getResponse( res, { error: err, post: post },
400 );
                if( rows ) fn.getResponse( res, { post, 'message':
'Berhasil di update' } )
                }
        )
}
```