Register : Jurnal Ilmiah Teknologi Sistem Informasi 4 (1) 2018 14-22
ISSN 2502-3357 (*Online*)
ISSN 2503-0477 (*Print*)

Tersedia *online* di www.journal.unipdu.ac.id
Unipdu

Halaman jurnal di www.journal.unipdu.ac.id/index.php/register

Interoperabilitas perangkat lunak menggunakan RESTful *web service*

M. Miftakul Amin

Teknik Komputer, Politeknik Negeri Sriwijaya, Palembang, Indonesia

email: mmiftakulamin@gmail.com , miftakul_a@polsri.ac.id

INFO ARTIKEL

Sejarah artikel:
Menerima 10 April 2018
Revisi 3 Agustus 2018
Diterima 3 Agustus 2018
Online 7 Agustus 2018

Kata kunci:
HTTP *method*
interoperabilitas
RESTful *web service*

Keywords:
interoperability
HTTP *method*
RESTful *web services*

Style APA dalam mensitasi artikel ini:
Amin, M. M. (2018). Interoperabilitas perangkat lunak menggunakan RESTful *Web service*. *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, 4(1), 14-22.

ABSTRAK

Pengembangan sistem informasi membutuhkan interoperabilitas dalam lingkungan yang heterogen, dilihat dari sistem operasi, perangkat lunak, bahasa pemrograman, dan basis data, sehingga dapat saling berkomunikasi dan bertukar data atau informasi. RESTful web *service* dapat digunakan sebagai salah satu teknologi untuk mewujudkan interoperabilitas. Sebuah studi kasus tentang aplikasi perpustakaan telah digunakan dalam penelitian ini. Aplikasi tersebut dibangun dengan Slim *Framework* PHP untuk sisi server dan Visual Basic pada sisi *client*. Komunikasi antara client dan server menggunakan HTTP *method* yaitu GET, POST, PUT, dan DELETE. Pengujian telah dilakukan untuk melihat performa dari web *service* yang telah dikembangkan menggunakan perangkat lunak Postman. Hasil dari penelitian ini menunjukkan bahwa, aplikasi *client* dapat mengakses web *service* yang disediakan di sisi server sebagai wujud interoperabilitas.

ABSTRACT

Information development systems need interoperability in heterogeneous environments, seen from operating systems, software, programming languages, and databases, so that they can communicate and exchange data or information. RESTful web services can be used as one of the technologies to realize interoperability. As case studies build library applications using PHP Slim Framework on the server side, while Visual Basic programming language is used on the client side. Communication Between client and server using HTTP Method that is GET, POST, PUT, and DELETE. Testing has been done to see the performance of web service functionality that has been developed using Postman software. The result shows that client applications can access the web services provided on the server side as a form of interoperability.

1. Pendahuluan

Penggunaan Teknologi Informasi dan Komunikasi (TIK) dapat meningkatkan efisiensi, efektifitas, transparansi, dan akuntabilitas sebuah organisasi. Optimalisasi pemanfaatan dari penggunaan TIK belum tercapai sepenuhnya. Hal ini dikarenakan penggunaan aplikasi dan basis data dijalankan di atas platform yang berbeda. Interoperabilitas merupakan isu yang menarik, dikarenakan semakin besarnya kebutuhan pertukaran data dan informasi, serta beragamnya platform yang digunakan. Sementara di sisi lain masing-masing pengguna memanfaatkan sistem operasi, perangkat lunak dan aplikasi yang berbeda. Menurut IEEE *Standard Computer Dictionary*, interoperabilitas didefinisikan sebagai kemampuan dari dua atau lebih sistem untuk dapat bertukar data atau informasi dan saling dapat mempergunakan data atau informasi yang dipertukarkan tersebut (Depkominfo, 2008). Pada awal pengembangan sistem terdistribusi untuk mencapai interoperabiltas tersedia beragam teknologi seperti RMI, CORBA, dan DCOM sebagai komunikasi yang terjadi antara sisi *client* dan sisi server (Mumbaikar & Padiya, 2013). Keterbatasan teknologi yang telah disebutkan adalah pada aspek ketergantungan (*dependency*) pada penggunaan platform. Seperti RMI hanya dapat berjalan di atas platform Java, sedangkan DCOM berjalan di atas platform Windows. Keterbatasan lain yang muncul adalah aspek

ISSN 2502-3357 (*Online*)
M. M. Amin/ Register 4 (1) 2018 14-22 ISSN 2503-0477 (*Print*)

kompatibilitas dan aspek keamanan pada tahap mengimplementasikannya. Selanjutnya muncul teknologi sebagai alternatif, yaitu *web service* sebagai platform *independent*, yang memungkinkan sebuah sistem dapat melakukan interaksi dan komunikasi pada lingkungan yang heterogen.

Dewasa ini menunjukkan bahwa *web service* merupakan pilihan teknologi yang paling sering digunakan untuk melakukan *information exchange and sharing* (Tihomirovs & Grabis, 2016). Modular dan dinamis menjadi salah satu ciri dari *web service*. *Protocol* SOAP maupun REST dapat digunakan untuk membangun *web service* dengan berbagai keunggulan serta kelemahannya. *Web service* memiliki ciri lain sebagai teknologi yang netral, berbasis teks serta dapat dikembangkan dengan memanfaatkan beragam teknologi dalam lingkungan heterogen (Dudhe & Sherekar, 2014). Teknologi *web service* ini telah ada sejak era 1990-an, dengan XML sebagai basis pengembangan *web service* dan dapat diimplementasikan pada cakupan *local*, berbasis web, dan terdistribusi. *Web service* dikembangkan pada teknologi *protocol* terbuka seperti TCP/IP dan HTTP. Representasi *web service* tidak mengharuskan adanya tampilan GUI, tetapi lebih memprioritaskan pada *business logic*, data, serta komunikasi yang dapat dijalankan di atas jaringan komputer. Selanjutnya *web service* dapat ditambahkan pada aplikasi *client* yang berperan sebagai sebuah referensi, selanjutnya dengan *parsing content* yang dihasilkan dari eksekusi *web service* dapat ditampilkan hasilnya pada aplikasi dan *device* yang berada pada sisi *client*. Pada penelitian yang dilakukan ini akan menguraikan implementasi RESTful *web service* sebagai sebuah teknologi untuk mewujudkan interoperabilitas yang dapat menjembatani antara platform yang berbeda, serta melakukan proses pengujian pada tahap pertukaran data yang dilakukan.

2. *State of the Art*

2.1. *Interoperabilitas aplikasi*

Interoperabilitas perangkat lunak secara sederhana merupakan kerjasama antara dua atau lebih program atau aplikasi yang berbeda platform sistem operasi, bahasa pemrograman, serta basis data yang memproses dan mengolah data tertentu. Penelitian yang telah dilakukan oleh (Amran, 2012) telah mengimplemtasikan teknologi *web service* dengan menggunakan *protocol* SOAP untuk mengembangkan penyediaan data KTP elektronik yang dapat diakses oleh aplikasi pada sisi *client* untuk berbagai keperluan.

2.2. *RESTful Web service*

Tabel 1. HTTP *method* dalam RESTful *web service*

Method	**CRUD**	**Keterangan**
POST	*Create*	Membuat data
GET	*Retrieve/Read*	Menampilkan data
PUT	*Update*	Memperbaiki data
DELETE	*Delete*	Menghapus data

Tabel 2. Kode HTTP status

Status	***Code***	**Keterangan**
Sukses	200	Berhasil melakukan operasi *Read*, *Update*, dan *Delete*.
	201	Berhasil melakukan operasi *Create.*
Error or Exception Codes	400 *Bad Request*	Perintah yang dikirimkan ke server tidak diketahui, atau salah dalam mengirimkan alamat URL.
	401 *Unauthorized*	Perintah yang dikirimkan tidak memiliki hak akses dari pengirim.
	422 *un-processable entity*	Perintah yang diberikan gagal dalam melakukan *create, update,* dan *delete*.
	404 *Not Found*	Informasi yang diminta tidak ada, atau tidak memiliki wewenang untuk mengakses informasi.
	500 *Internal server error*	Terjadi kesalahan pada sisi server, biasanya karena kesalahan konfigurasi web server.

Menurut Sinha, Khatkar, dan Gupta (2014) mengungkapkan bahwa pada awal pengembangan *web service, protocol* yang sering digunakan dalam pengembangan *web service* adalah SOAP, serta menggunakan XML sebagai format *data interchange*. Perkembangan terbaru sekarang muncul teknologi

M. M. Amin/ Register 4 (1) 2018 14-22
ISSN 2502-3357 (*Online*)
ISSN 2503-0477 (*Print*)

RESTful *web service* yang menggunakan *protocol* HTTP sebagai standar *protocol* aplikasi web dan format pertukaran data menggunakan Java Script Object Notation (JSON). Pada prinsipnya *request* ke sebuah RESTful *web service* merupakan HTTP *request* (Surendra, 2014). Untuk mengimplementasikan RESTful *web service* digunakan beberapa *method* seperti *POST, PUT, GET*, dan *DELETE*. Tabel 1 memberikan gambaran *method* HTTP yang digunakan dalam RESTful *web service,* dan jika dihubungkan dengan operasi pada *database* juga dikenal dengan operasi CRUD (*Create, Read, Update,* dan *Delete*).

RESTful *web service* menggunakan URI (*Uniform Resource Indentifier*) sebagai informasi unik. Interaksi yang dilakukan antara *client*/*server* dilakukan dengan berikirim informasi melalui URI. Setiap *request* yang dikirimkan ke server melalui *protocol* HTTP akan menghasilkan *response* berupa kode unik seperti pada Tabel 2.

2.3. Protokol web service

Web service merupakan sebuah metode untuk melakukan komunikasi antara dua buah perangkat elektronik atau lebih di atas jaringan komputer. Terdapat dua buah *protocol* yang lazim digunakan dalam pengembangan *web service*, yaitu SOAP dan REST (Kumari, 2015). SOAP adalah sebuah protokol untuk berkirim pesan yang memungkinkan program yang berjalan di beberapa sistem operasi (seperti Windows dan Linux) untuk dapat berkomunikasi menggunakan protokol HTTP serta format data XML. REST merupakan arsitektur dari *web service* yang memanfaatkan *protocol* HTTP untuk melakukan pertukaran data. Konsep REST diperkenalkan pada tahun 2000 pertama kali oleh Roy Fielding. REST server menyediakan jalur untuk melakukan akses *resource* maupun data, sedangkan REST *client* melakukan akses *resource* untuk selanjutnya menampilkan atau menggunakan datanya. *Resource* yang dihasilkan dari proses tersebut berupa data teks, format umumnya adalah JSON dan XML. Menurut Wagh dan Thool (2012) terdapat beberapa pertimbangan jika akan mengembangkan aplikasi menggunakan *protocol* SOAP dan REST seperti dapat dilihat pada Tabel 3.

Tabel 3. Perbandingan SOAP dan REST

SOAP	REST
Dikenal sebagai teknologi tradisional.	Dianggap sebagai teknologi baru penerus dari SOAP.
Komunikasi antara *client* dan server bersifat *tightly coupled*	Komunikasi antara *client* dan server bersifat *loosely coupled* yang berarti perancangan yang dibuat secara bebas dan merdeka, sehingga mudah diubah dan lebih fleksibel.
Perubahan pada sisi server akan sangat berdampak pada perubahan yang ada di sisi *client*	Tidak terlalu banyak perubahan yang harus dilakukan pada sisi *client*, jika terjadi perubahan pada sisi server
Jumlah data yang ditransfer dari server ke *client* cukup besar	Jumlah data yang ditransfer dari server ke *client* cukup ringan
SOAP selalu mengembalikan data dalam bentuk XML.	Data yang dikembalikan cukup fleksibel.
Respone time cukup tinggi.	*Response time* relatif lebih rendah.

Penelitian serupa yang lain juga telah dilaksanakan oleh Dudhe dan Sherekar (2014) yang menguji *protocol* SOAP dan REST untuk mengembangkan *web service* dan diperoleh hasil bahwa *protocol* REST lebih cepat dan memiliki *performance* lebih dibandingkan dengan *protocol* SOAP. Dari sisi *response time protocol* REST lebih cepat dibandingkan dengan SOAP (Johal & Singh, 2014). Kecepatan *response time* ini dipengaruhi oleh beberapa faktor diantaranya kecepatan pemrosesan web server, *bandwidth* jaringan, *payload*, jarak antara *client* dan *server*, serta seberapa banyak *client* yang mengakses *web service*.

2.4. Meningkatkan kinerja RESTful web service

Kinerja dari RESTful *web service* dapat ditingkatkan dengan melakukan beberapa pendekatan dalam pengembangan *Web service* (Tere, Mudholkar, & Jadhav, 2014) diantaranya adalah dengan cara:

1. Menggunakan stringBuilder untuk mengelola data *numeric* dan *string* yang cukup besar
2. Melakukan *streaming*
3. Mengkompres SOAP *response*
4. *Partial representation*
5. Menggunakan *cache*
6. Menggunakan metode kondisional

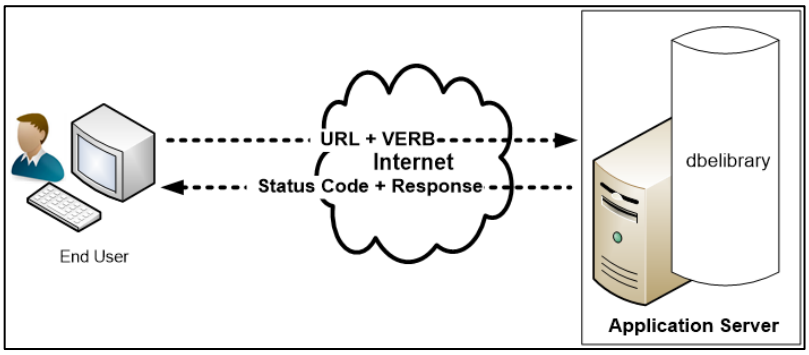
ISSN 2502-3357 (*Online*)
M. M. Amin/ Register 4 (1) 2018 14-22 ISSN 2503-0477 (*Print*)

Penelitian yang telah dilakukan oleh Hidayatullah, Sari, dan Faiqurahman (2017) telah mengimplementasikan RESTful *web service* sebagai teknologi *pull message* pada *Wireless Sensor Network* (WSN) untuk berkirim data antar *sensor node (gateway)*. Hasil penelitian ini menunjukkan bahwa *interval sensing*, ukuran data, dan jumlah *sink node* melakukan *request*, tidak begitu berpengaruh terhadap ketersediaan *free memory heap* pada *sensor node*. Sedangkan ukuran dari data hasil *sensing* yang dikirim mempunyai pengaruh terhadap *request-response time.*

3. Metode Penelitian

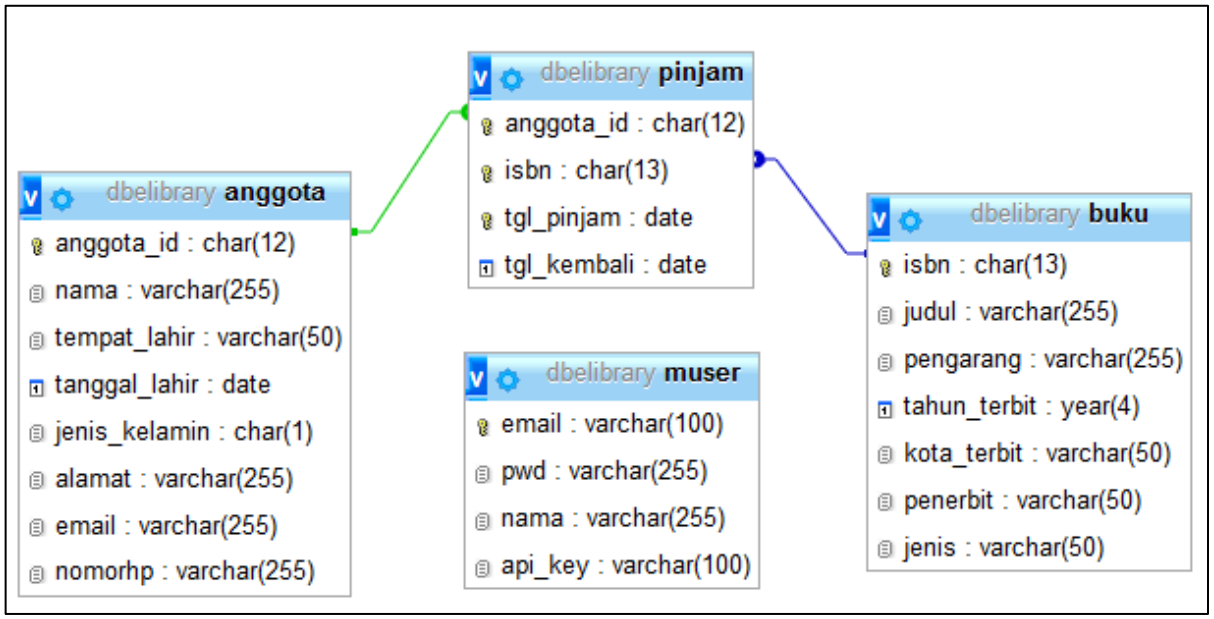
3.1. Arsitektur aplikasi

Secara umum arsitektur *web service* yang dikembangkan menggunakan teknologi RESTful *web service* dapat dilihat seperti pada Gambar 1. *End user* merupakan sebuah aplikasi yang berperan sebagai *consumer* yang memberikan *response* berupa URL dan parameter tertentu sesuai spesifikasi yang telah ditentukan. Selanjutnya server akan memberikan *response* berupa *status code* dan *response* dari *request* yang dilakukan. Pada sisi server terdapat sebuah *application server* yang nantinya berperan melayani *request* dari *client*. Pada sisi server digunakan Slim Framework PHP untuk membangun RESTful *web services*.



Gambar 1. Arsitektur aplikasi RESTful w*eb service*

3.2. Physical Database Diagram (PDM)



Gambar 2. Rancangan *Physical Database Diagram*

Untuk mengimplementasikan teknologi RESTful *web service* digunakan basis data MySQL sebagai *persistent storage* yang akan menyimpan data secara permanen. Walaupun dalam SLIM framework telah disediakan database khusus, tetapi dalam pengembangan sistem diperlukan basis data yang independen tidak terikat dengan framework tersebut untuk memudahkan pengembagnan sistem lebih lanjut. Gambar 2 merupakan rancangan *Physical Database Diagram* (PDM) yang memperlihatkan relasi antar tabel yang digunakan. Rancangan database yang digunakan ini tidak begitu kompleks,

M. M. Amin/ Register 4 (1) 2018 14-22
ISSN 2502-3357 (*Online*)
ISSN 2503-0477 (*Print*)

dikarenakan tujuan dari penelitian ini adalah melakukan simulasi dan menguji teknologi RESTful *web service* yang dapat digunakan dalam lingkungan terdistribusi dan diakses melaui aplikasi client yang berbeda platform.

Gambar 2 diperlihatkan relasi yang terbentuk antara tabel yang mencerminkan proses bisnis sebagai berikut:

1. Relasi antara tabel anggota dan pinjam adalah 1:n (*one to many*), informasi ini memberikan *constraint* dalam perilaku sistem bahwa pada tabel anggota, keberadaan anggota hanya disimpan sebanyak 1 kali saja (maknanya *one*), sedangkan satu orang anggota dapat melakukan peminjaman beberapa kali (maknanya *many*), yang artinya akan tercatat dalam tabel pinjam dengan lebih dari 1 *record*.
2. Relasi yang terbentuk antara buku dan pinjam merupakan relasi 1:n (*one to many*), yang memiliki makna bahwa data buku akan dicatat sebanyak 1 kali (*one*) pada tabel buku, dan akan tercatat lebih dari 1 kali (*many*) pada tabel peminjaman.
3. Sedangkan tabel muser merupakan tabel untuk menyimpan informasi *user* yang nantinya berhak menggunakan layanan API dalam RESTful *web service*. Tabel ini tidak berelasi dengan tabel yang lain, karena secara logika tidak berhubungan dengan tabel-tabel tersebut. Informasi penting yang dimuat dalam tabel muser adalah api_key sebagai salah satu parameter yang diperlukan untuk mengakses setiap URL yang disediakan oleh RESTful *web service*. Tabel ini menyimpan informasi yang berhubungan dengan proses registrasi *user* dan *login user* pada level komunikasi antara *client* dan RESTful web *service*. Proses otentikasi yang dicatat di sini adalah informasi penting seperti email, *password*, nama dan api_key. Keberadaan api_key ini akan dimasukkan dalam proses pengiriman permintaan dari *client* yang menggunakan HTTP *method* GET, POST, PUT, dan DELETE. Informasi api_key akan dikirimkan pada bagian HEADER HTTP dari proses *request* oleh *client* ke server.

Tabel 4. Daftar Rancangan URL API

No.	URL	*Method*	Parameter	Deskripsi
API1	/regUser	POST	email, pwd, nama	*Method* yang digunakan untuk meregister user RESTful *web service*
API2	/loginUser	POST	email, pwd	*Method* yang digunakan untuk login user yang telah teregistrasi
API3	/listAnggota	GET	API_KEY	*Method* yang digunakan menampilkan data seluruh anggota
API4	/addAnggota	POST	anggota_id, nama, API_KEY	*Method* yang digunakan menampilkan data seluruh anggota
API5	/findAnggota/:anggota_id	GET	anggota_id, API_KEY	*Method* yang digunakan untuk mencari data anggota berdasarkan parameter anggota_id
API6	/modifyAnggota/:anggota_id	PUT	anggota_id, nama, API_KEY	*Method* yang digunakan untuk memperbaiki data anggota berdasarkan parameter anggota_id
API7	/removeAnggota/:anggota_id	DELETE	anggota_id, API_KEY	*Method* yang digunakan untuk menghapus data anggota berdasarkan parameter anggota_id
API8	/listBuku	GET	API_KEY	*Method* yang digunakan untuk menampilkan seluruh data buku
API9	/addBuku	POST	isbn, judul, pengarang, API_KEY	*Method* yang digunakan untuk menambah data buku
API10	/findBuku/:isbn	GET	isbn, API_KEY	*Method* yang digunakan untuk mencari data buku berdasarkan isbn
API11	/modifyBuku/:isbn	PUT	isbn, judul, pengarang, API_KEY	*Method* yang digunakan untuk memperbaiki data buku
API12	/removeBuku/:isbn	DELETE	isbn, API_KEY	*Method* yang digunakan untuk menghapus data buku
API13	/listPinjaman	GET	API_KEY	*Method* yang digunakan untuk menampilkan data seluruh pinjaman
API14	/addPinjaman	POST	anggota_id, isbn, tgl_pinjam, tgl_kembali, API_KEY	*Method* yang digunakan untuk memasukkan data pinjaman

ISSN 2502-3357 (*Online*)
M. M. Amin/ Register 4 (1) 2018 14-22 ISSN 2503-0477 (*Print*)

3.3. Struktur URL RESTful *web service*

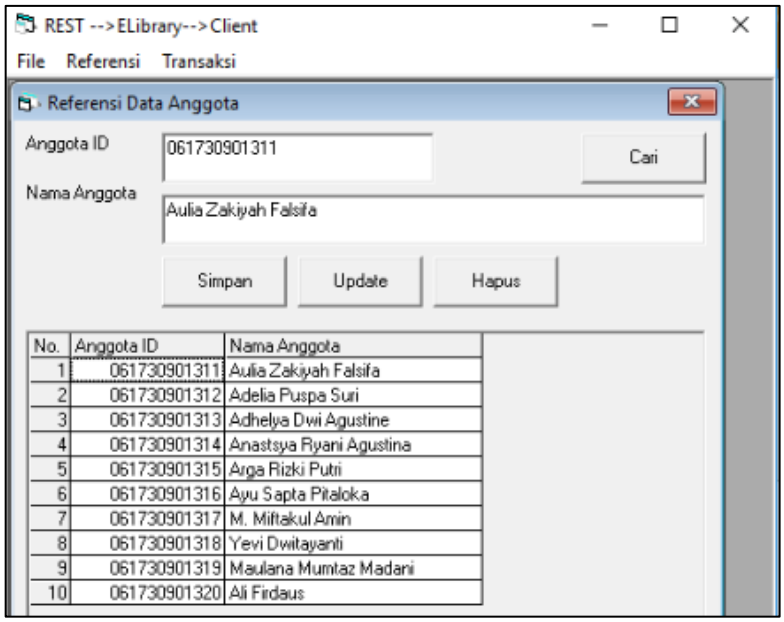
Pada tahap pengaksesan *method-method* atau API (*Application Programming Interface*) yang disediakan oleh RESTful *web service*, digunakan sebuah alamat URL yang rapi supaya memudahkan orang lain yang menggunakannya. Tabel 4 merupakan daftar URL yang digunakan untuk mengakses API. Setiap *user* yang akan mengakses RESTful *web service* ini terlabih dahulu teregister sebagai *user* melalui *method* /regUser, yang selanjutnya akan mendapatkan sebuah API_KEY untuk mengakses seluruh layanan yang ada dalam RESTful *web service*.

4. Hasil dan Pembahasan

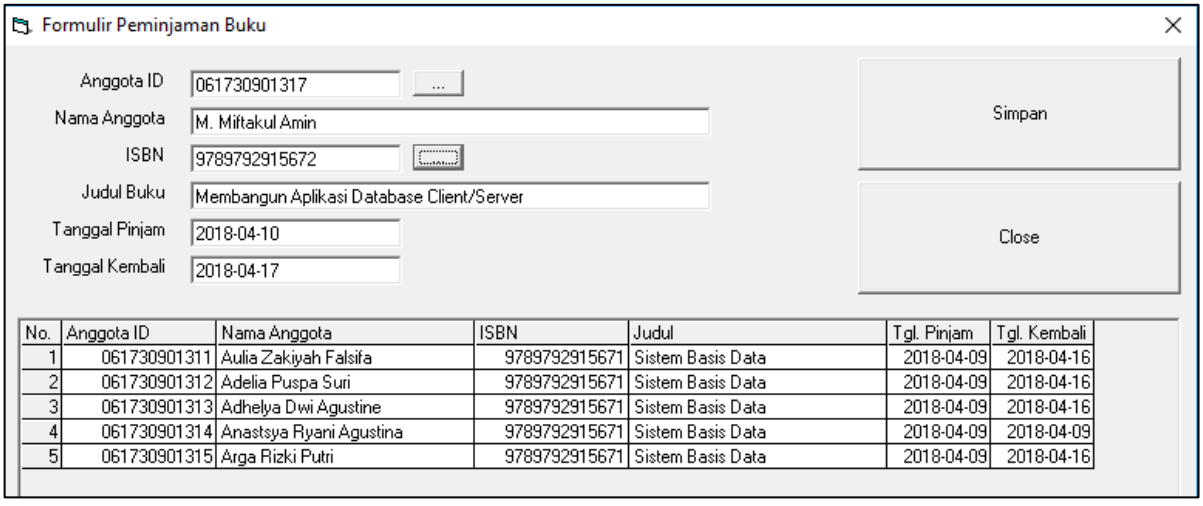
Setelah tahapan rancangan *physical database diagram* dan implementasi dalam pemrograman di sisi server untuk menyediakan API, maka tahapan selanjutnya adalah melakukan pengujian aplikasi dengan menggunakan beberapa perangkat lunak yaitu Visual Basic, dan Postman.

4.1. Pengujian Client Visual Basic

Pengujian pada bahasa pemrograman Visual Basic ini untuk memastikan bahwa proses komunikasi antara platform bahasa pemrograman yang berbeda dan proses pertukaran data dapat dilakukan. Gambar 3 (a) Merupakan data anggota yang berhasil dimuat ke dalam aplikasi, sedangkan Gambar 3 (b) Merupakan tampilan data pinjaman.



(a) Data Anggota



(b) Data Pinjaman

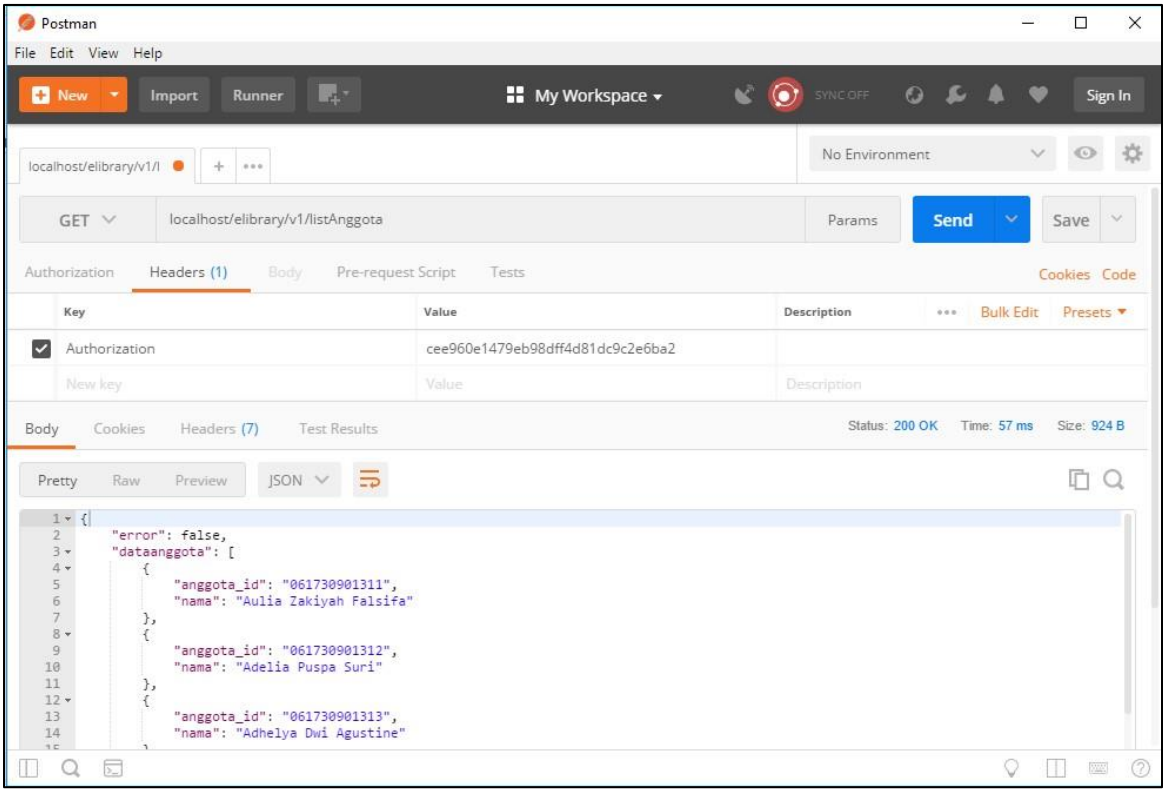
Gambar 3. Pengujian pada *Software* Visual Basic

M. M. Amin/ Register 4 (1) 2018 14-22
ISSN 2502-3357 (*Online*)
ISSN 2503-0477 (*Print*)

Di dalam bahasa pemrograman Visual Basic diperlukan sebuah referensi Microsoft WinHTTP *services version* 5.1 untuk melakukan komunikasi dengan RESTful *web services*. Dengan menggunakan *method* o*pen*, SetRequestHeader dan *Send* yang dimiliki oleh objek WinHTTP tersebut komunikasi antara bahasa pemrograman Visual Basic dan RESTful *web service* dapat dilakukan.

4.2. Pengujian menggunakan software Postman

Pengujian pada perangkat lunak Postman digunakan untuk mengetahui sejauh mana *performance request – response time* yang terjadi antara *client* dan *server* pada saat melakukan komunikasi dan bertukar data. Gambar 4 merupakan tampilan dari RESTful *web service* yang diuji menggunakan perangkat lunak Postman. Dengan menggunakan API_KEY sebagai sebuah parameter yang perlu ditambahkan pada bagian *Header* pada saat melakukan pengiriman URL ke RESTful *web service*, selanjutnya data yang diminta sebagai hasil dari *response* akan ditampilkan pada bagian bawah berupa format data JSON.



Gambar 4. Pengujian RESTful *web service* menggunakan *Postman*

Dari beberapa pengujian yang dilakukan terhadap sejumlah *method* yang disediakan oleh RESTful *web service* diperoleh informasi terkait dengan *payload* sebagai data dan *response time* seperti diperlihatkan pada Tabel 5.

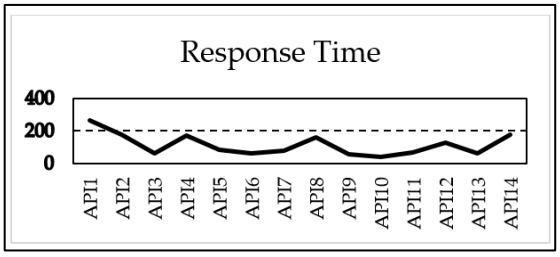
Tabel 5. Pengujian *performance* RESTful *web service*

API	*Size Requested* Data (Bytes)	*Response Time* (ms)
API1	304	261
API2	368	174
API3	857	60
API4	319	170
API5	329	84
API6	296	63
API7	570	78
API8	316	158
API9	322	56
API10	395	37
API11	302	64
API12	327	124
API13	1.070	62
API14	320	176

ISSN 2502-3357 (*Online*)
M. M. Amin/ Register 4 (1) 2018 14-22 ISSN 2503-0477 (*Print*)

Gambar 5 merupakan grafik yang memperlihatkan *performance response time* dari setiap API yang disediakan oleh RESTful *web services*. Dari beberapa pengujian yang dilakukan, nilai *response time* ini bergerak naik turun walaupun dengan jumlah *payload* yang sama. Hal ini dipengaruhi oleh beberapa faktor, diantaranya:

1. Kecepatan pemrosesan *web server*, hal ini ada kemungkinan *resource* komputasi digunakan oleh proses lain yang terjadi di dalam komputer server.
2. *Bandwidth* jaringan, hal ini dikarenakan besarnya *bandwidth* berbeda-beda pada saat mengeksekusi sebuah *method web service*.
3. *Payload*, jumlah data atau *record* yang ditransportasikan antara *client*/*server*.



Gambar 5. *Response time* pengujian *method* RESTful *web service*s

5. Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, maka dapat disimpulkan bahwa RESTful *web service* dapat diimplementasikan dalam lingkungan yang heterogen dengan menggunakan bahasa pemrograman yang berbeda untuk dapat melakukan komunikasi dan bertukar data atau informasi. Pengujian menggunakan perangkat lunak Postman juga menunjukkan bahwa *performance* dari teknologi RESTful *web service* dapat direkomendasikan untuk pengembangan aplikasi terdistribusi. Pertimbangan rekomendasi tersebut dapat dilihat beberapa karakteristik yang diperoleh dari hasil penelitian, diantaranya: 1) Arsitektur RESTful *web service* bersifat terbuka, sehingga dapat diakses oleh siapa saja yang memiliki permission ke dalam sistem; 2)Arsitektur RESTful web *service* menyediakan *resources* berupa sumber data dalam bentuk URL, sehingga dapat diakses oleh *client* menggunakan platform yang berbeda; 3) Data yang dipertukarkan antara *client* dan server berupa format data teks seperti JSON sehingga ringan dalam transportasi jaringan; dan 4) Beberapa pengujian yang telah dilakukan terlihat bahwa *response time* yang cukup cepat pada proses pertukaran data dan informasi. Sehingga penelitian lanjutan dapat dikembangkan ke arah pengujian dari sisi aspek keamanan, dan *code design pattern* untuk menghasilkan kode program yang lebih terstruktur dan memiliki fleksibilitas yang tinggi.

6. Ucapan Terima Kasih

Kami mengucapkan terimakasih kepada tim redaksi jurnal register yang telah memberikan kesempatan kepada penulis, sehingga artikel ilmiah ini dapat dipublikasikan.

7. Referensi

Amran, R. Y. (2012). Interoperabilitas Sistem KTP Elektronik Terdistribusi Berbasis Simple Object Access Protocol (SOAP). *Inspiration: Jurnal Teknologi Informasi dan Komunikasi, 2*(1), 69-76.

Depkominfo. (2008). *Kerangka Acuan dan Pedoman Interoperabilitas Sistem Informasi Instansi Pemerintah.* Jakarta: Direktorat Sistem Informasi, Perangkat Lunak dan Konten; Direktorat Jenderal Aplikasi Telematika; Departemen Komunikasi dan Informatika.

Dudhe, A., & Sherekar, S. S. (2014). Performance Analysis of SOAP and RESTful Mobile Web Services in Cloud Environment. *International Journal of Computer Applications*, 1-4.

Hidayatullah, R. A., Sari, Z., & Faiqurahman, M. (2017). Implementasi Pull Message dengan menggunakan Restful Web Service pada komunikasi Wireless Sensor. *Register: Jurnal Ilmiah Teknologi Sistem Informasi, 3*(2), 65-74.

Johal, A. S., & Singh, B. (2014). Performance analysis of web services for Android based devices. *International Journal of Computer Applications, 92*(11), 0975–8887.

M. M. Amin/ Register 4 (1) 2018 14-22
ISSN 2502-3357 (*Online*)
ISSN 2503-0477 (*Print*)

Kumari, V. (2015). Web Services Protocol: SOAP vs REST. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 4*(5), 2467- 2469.

Mumbaikar, S., & Padiya, P. (2013). Web Services Based On SOAP and REST Principles. *International Journal of Scientific and Research Publications, 3*(5), 1-4.

Sinha, R., Khatkar, M., & Gupta, S. C. (2014). Design & Development of a REST based Web Service Platform for Applications Integration on Cloud. *IJISET - International Journal of Innovative Science, Engineering & Technology, 1*(7), 385-389.

Surendra, M. R. (2014). Implementasi PHP Web Service Sebagai Penyedia Data Aplikasi Mobile. *ULTIMATICS, VI*(2), 85-93.

Tere, G. M., Mudholkar, R. R., & Jadhav, B. T. (2014). Improving Performance of RESTful Web Services. *International Conference on Advances in Engineering & Technology* , (pp. 12-16). Kuala Lumpur, Malaysia.

Tihomirovs, J., & Grabis, J. (2016). Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics. *Information Technology and Management Science, 19*(1), 92–97.

Wagh, K., & Thool, R. (2012). A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host. *Journal of Information Engineering and Applications, 2*(5), 12-16.