JASIEK

Jurnal Aplikasi Sains, Informasi, Elektronika dan Komputer

ISSN- 2685-497X Vol 2, Issue 1, 2020

Penerapan *RESFUL Web Service* pada Disain Arsitektur Sistem Informasi pada Perguruan Tinggi (Studi Kasus: STARS UKSW)

Penidas Fiodinggo Tanaem [a,1,*], Augie David Manuputty [b,2], George Nicholas Huwae[b,3], Agustinus Fritz Wijaya [b,4]

[a]Universitas Kristen Satya Wacana, Jl. Diponegoro No.52-60 Jawa Tengah, Salatiga, Indonesia
[1]penidas.tanaem@uksw.edu *; [2]augie.manuputty@uksw.edu ; [3]george.huwae@uksw.edu ;
[4]agustinus.wijaya@uksw.edu
* Penulis Koresponden

INFO ARTIKEL

Histori Artikel
Pengajuan 12 Agustus 2019
Diperbaiki 20 September 2019
Diterima 4 November 2019

Kata Kunci
web Service
Restful
Sistem Informasi
Perguruan Tinggi.

ABSTRAK

RESTFul WS merupakan sebuah gaya arsitektur yang diadopsi dari konsep *REST* dengan tujuan yakni sebagai penghubung antara sumber daya dan *client*. Komunikasi *RESTFul* memanfaatkan protocol *HTTP*, dimana dalam *HTTP* terdapat *method* yang digunakan untuk berkomunikasi yang dapat disesuaikan dengan operasi-operasi yang digunakan dari sisi *client* yakni operasi *CRUD* (*CREATE, READ, UPDATE, DELETE*). Hasil dari penelitian yang dilakukan yakni berupa hasil eksperimen dalam membangun *RESTFul STARS*. Cakupan dalam uraian tersebut antara lain berupa kolaborasi antar *system* dan mekanisme interaksi yang dibangun dalam menjawab kebutuhan aplikasi *STARS*. Sistem dibangun menggunakan pendekatan *REST* yang mengandalkan *protocol HTTP* dalam membangun interaksi antara *client* dan *server* dimana terdapat rancangan model *RESTFul* yang dibangun dan mencakup komunikasi dan arsitektur *RESTFul*. Adapun terdapat tiga tahapan yang digunakan dalam pelaksanaannya, yakni tahap *Requirements, Analysis, Design* dan *Implementation*. Berdasarkan hasil penelitian yang dilakukan, terdapat korelasi yang telah disesuaikan antara Metode *HTTP* (*POST, GET, PUT, DELETE*) dan operasi *CRUD* dan batasan dalam membangun *RESTFul* terutama untuk *STARS* yang mengacu pada *Uniform Interface, Connectedness, Self-Describing Messages* dan *Stateless Interactions.*

1. Pendahuluan

Keberadaan *Web Service* memberikan dampak yang besar terhadap pembuatan maupun pengembangan sistem yang digunakan dalam lingkup organisasi. Dampak tersebut dapat dilihat dari *Web Service* yang berperan untuk [1] menghubungkan masing-masing sumber data (*database*) dan *client* (aplikasi). *Web Service* dapat disefinisikan sebagai alat bantu dalam mediasi aplikasi dan *database* dalam hal menemukan, menambahkan, membuat dan menghapus (*CRUD*). Namun, dalam menjawab kebutuhan dalam membangun maupun

mengembangkan sebuah sistem, terdapat kriteria-kriteria yang menjadi perhatian khusus. Diantaranya: multi format, *lightweight system* dan dukungan layanan. Hal ini didasarkan pada pembuatan aplikasi STARS (*Student Activity Record Systems*) Universitas Kristen Satya Wacana (UKSW) Salatiga.

STARS UKSW adalah sistem informasi yang digunakan untuk melakukan pencatatan akan kegiatan-kegiatan yang dilaksanakan dilingkungan UKSW yang berbasis *web*. Ruang lingkup *STARS* adalah rekapan keseluruhan kegiatan internal, peserta (dalam hal ini mahasiswa yang mengikuti sebuah kegiatan), beserta dokumen-dokumen pendukung yang berfungsi sebagai bukti maupun sebagai penunjang pelaksanaan kegiatan yang dimaksud. Adapun dokumen-dokumen pendukung yang digunakan antaralain: *PDF, DOC, IMG* dan lain-lain. Dalam penerapannya, diperlukan sebuah *service* yang dapat digunakan untuk menghubungkan sumber data dan *client* untuk menunjang aktivitas menipulasi data. Hal ini sangat diperlukan dalam melakukan manajeman *request* dari *client*, representasi data dan tipe data yang menjadi pertimbangan dalam pembuatan *STARS*. Dengan demikian, pilihan yang diambil dalam mengatasi pertimbangan tersebut adalah penggunaan *RESTFul Web Service*. *RESTFul Web Service* berfungsi untuk merepresentasikian sumber daya yang dimiliki. Dalam penerapannya, *RESTFul* digunakan untuk manajemen *request client* melalui protokol *HTTP* [2][3][4][5][6] dengan memanfaatkan *Method POST, PUT, GET, DELETE* dan lain-lain. Dalam hal ekplorasi data, *RESTFul* dapat memanfaatkan multi format data, dalam hal ini (XML, JSON, ATOM, dan lain-lain) [7][8][3][6][9] sebagai *interface* yang berbeda-beda. Selain itu, *RESTFul* sangat cocok untuk aplikasi berbasis *web* dan juga menjadi salah satu *lightweight system*. Mengacu pada kondisi yang diuraikan, maka pada artikel ini akan diuraikan hasil eksperimen dalam membangun *RESTFul STARS*. Cakupan dalam uraian tersebut antara lain berupa kolaborasi antar sistem dan mekanisme interaksi yang dibangun dalam menjawab kebutuhan aplikasi *STARS*.

RESTFul Web Service merupakan sebuah pendekatan baru dalam bidang *Web Service*. *RESTFul Web Service* adalah sebuah komponen sistem yang berperan sebagai mediator atau penghubung dalam melakukan ekplorasi sumber daya. Komunikasi yang digunakan yaitu dengan memanfaatkan protokol *HTTP* [2]. *RESTFul* dibangun mengacu pada konsep *REST*. *REST* sendiri merupakan sebuah gaya arsitektur [10][6][3]. Gagasan utama REST terletak pada gagasan tentang sumber daya sebagai komponen aplikasi yang perlu untuk digunakan [10]. Dalam gaya arsitektur REST, sumber daya diidentifikasi melalui URI, yang merupakan jembatan untuk menemukan sumber daya dan Service [1][11], yang mana sebuah URI dapat mewakili maing-masing sumber daya yang akan diakses.

RESTFul yang peranannya sebagai *Back-End* harus dapat diakses melalui internet, dapat menerima *request* dari *client* dan merespon kembali ke *client*. Kondisi tersebut juga diimplementasikan dalam melakukan operasi *CRUD* [12][13] [14][2]. Dalam penerapannya, operasi *CRUD* dapat ditangani menggunakan beberapa *HTTP Method*. Sebagai contoh *Create* dapat ditangani dengan menggunakan *HTTP Method POST*; *Read* dapat ditangani dengan menggunakan *Method GET*; *Update* dapat ditangani menggunakan *Method PUT*; sedangkan *Delete* menggunakan *Method DELETE*.

Gaya arsitektur *REST* memiliki lima batasan yang umum. Lima batasan tersebut antara lain: *Resource Identification* yang berarti sumberdaya diakses melalui URI; *Connectedness* yang berarti setiap sumber daya harus dialamatkan; *Uniform Interface* yang berarti pemanfaatan metode HTTP dengan semantik yang berbeda-beda; *Self-Describing Messages* yakni *support* terhadap presentasi data dalam tipe data yang berbeda-beda; dan *Stateless Interactions* yang berarti bahwa tidak ada ketergantungan terhadap *state* [2][10][6].Rodrıguez dkk menekankan bahwa terdapat batasan-batasan dalam membangun

sebuah *RESTFul*, antara lain: tidak ada nama *CRUD* di *URL* (http://{domain}/add/mahasiswa namun http://{domain}/mahasiswa) dan tidak ada transparansi teknologi yang diimplementasikan pada sisi *server* seperti ekstensi bahasa pemrograman (http://{domain}/mahasiswa.php/ namun http://{domain}/mahasiswa/) [9].

Sebuah penelitian yang dilakukan oleh Xiong dkk., diusulkan pendekatan *deep learning based hybrid* yang baru untuk rekomendasi *Web Service* dengan menggabungkan filter kolaboratif dan konten tekstual. Interaksi pemanggilan antara *mashup* dan *service* serta fungsionalitasnya yang terintegrasi sempurna ke dalam *neural network*, yang dapat digunakan untuk mengkarakterisasi hubungan yang kompleks antara *mashup* dan *service*. Eksperimen yang dilakukan pada data *Web Service* menunjukkan bahwa pendekatan yang ada dapat mencapai kinerja rekomendasi yang lebih baik daripada beberapa metode *state-of-the-art*, yang menunjukkan efektivitas pendekatan yang yang diusulkan dalam rekomendasi service [15][16]. Sedangkan Shodiq dkk. mengimplementasikan algoritma sinkronisasi yang mampu menjaga konsistensi data yang dimiliki oleh berbagai database melalui *Web Service*, sehingga datanya selalu *up to date*. Dalam penelitian ini, implementasi akan menggambarkan perbedaan dari algoritma sinkronisasi data antara sinkronisasi satu arah dengan sinkronisasi dua arah pada perangkat yang berbeda melalui *Web Service* sehingga pengguna masih dapat menggunakan data yang sama meskipun menggunakan perangkat yang berbeda. Implementasi juga menambahkan informasi tambahan yang disebut sebagai penanda untuk memutuskan apakah data perlu diabaikan atau disinkronkan ke *database* lain [17][18]. Selanjutnya penelitian lainnya dilakukan oleh Mariyam dkk bertujuan untuk merancang Sistem Infomasi Poin Non Akademik (*E-Point*) pada STIKOM PGRI Banyuwangi. Manfaat yang diharapkan dari penelitian tersebut adalah untuk dapat mempermudah para mahasiswa untuk mendapatkan informasi terkait *Poin Non* Akademik mereka. Penelitian tersebut menghasilkan sebuah rancangan Sistem *Poin Non* Akademik pada STIKOM PGRI Banyuwangi yang disesuaikan dengan Standar Operasional Prosedur tentang Point Kemahasiswaan, yang berfungsi untuk memudahkan Kepala Bagian Kemahasiswan dalam melakukan rekapan data poin non akademik kegiatan mahasiswa [19].

2. Metode Penelitian

Dalam pelaksanaan penelitian ini, terdapat enam tahapan yang dilalui. Tahapan-tahapan tersebut antara lain: tahap *Requirements, Analysis, Design* dan *Implementation*.

- ***Requirements***

Dalam tahapan ini, dilakukan pengumpulan data melalui wawancara kepada pihak-pihak terkait dalam hal ini adalah Bidang Kemahasiswaan yang salah satu tugasnya adalah kepengurusan kegiatan-kegiatan kemahasiswaan. Tindakan ini diambil dengan tujuan untuk memahami kebutuhantehadap systemyang diinginkan.

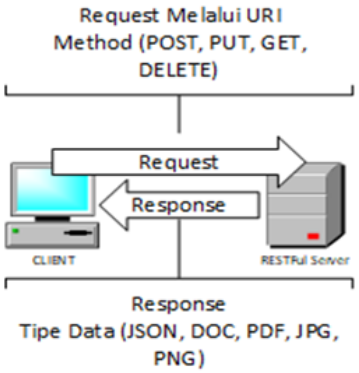
- ***Analysis***

Dari hasil wawancara yang dilakukan pada tahap requirements, maka diterjemahkan kedalam model yang secara konseptual dapat dipahamisecara teknis oleh pengembang. Hasil yang didapatkan dari tahap ini adalah berupa *interface* dari *RESTFul* yang akan dibangun.

- ***Design***

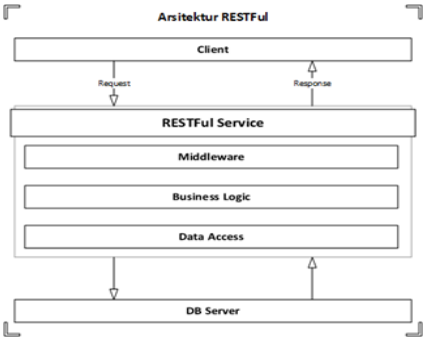
Adapun pada tahap desain, dihasilkan sebuah model yang digunakan dalam membangun *RESTFul*. Rancangan model *RESTFul* yang dibangun mencakup dua komponen

yang menjadi prioritas. Komponen-komponen tersebut diantaranya yakni komunikasi dan arsitektur. Komunikasi (Gambar 1) yang dimaksud adalah komunikasi antara *client* dan *RESTFul*. Hal ini bertujuan untuk pemanfaatan komunikasi melalui *HTTP Method*, yakni *POST, PUT, GET* dan *DELETE*, dalam hal ini adalah *Request*. Keempat *HTTP Method* tersebut menjadi standar dalam model *RESTFul* yang dibangun. Sedangkan *Response* dapat berupa format data *JSON, DOC, PDF, JPG* dan *PNG*.



Gambar 1. Model Komunikasi *RESFul*

Dalam struktur *Service RESTFul*, terdapat beberapa *layer* utama yang digunakan dalam operasinya. *Layer-layer* tersebut antara lain: layer *Middleware*, layer *Business Logic* dan layer *Data Access*. *Layer Middleware* berperan untuk melakukan validasi terhadap setiap *request* yang dilakukan oleh *client*. Sedangkan layer *Business Logic* berfungsi untuk mengimplementasikan fungsionalitas inti dari sistem dan merangkum logika bisnis yang telah disesuaikan. Sedangkan *layer Data Access* memiliki peran untuk mengekspos data berdasarkan batasan yang dimiliki oleh sistem (dalam hal ini, *logic* yang dihasilkan pada layer *Business Logic*) [20].



Gambar 2. Arsitektur *RESFul*

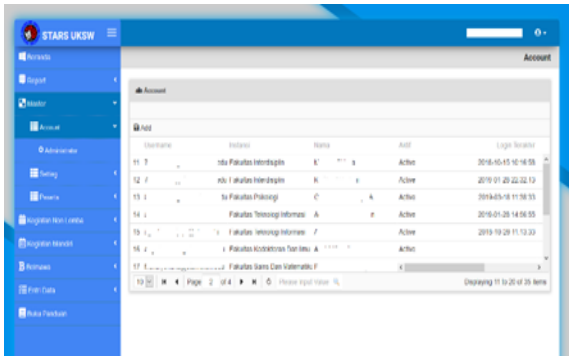
- ***Implementation***

Pada tahap *Implementation* dilakukan implementasi *RESTFul* yang mana pada dasarnya sangat mirip dengan fase pengkodean berbasis komponen perangkat lunak lainnya. Perbedaan utama terletak pada pembuatan antarmuka *RESTFul* (untuk mengekspos komponen sumber daya).

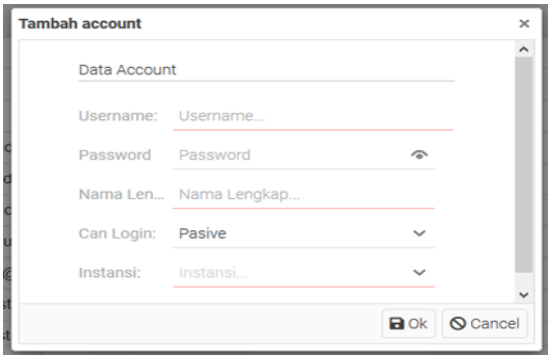
3. Hasil dan Analisis

Berdasarkan pada penelitian yang dilakukan, terdapat beberapa sub-sub sistem yang berperan dalam operasional *STARS*. Sub-sub sistem tersebut dibagi dalam beberapa

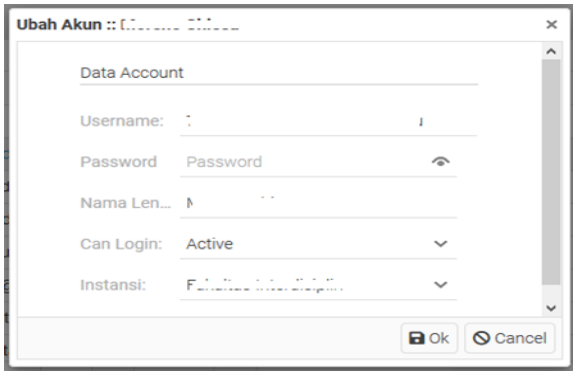
kelompok yang digunakan berdasarkan fungsinya masing-masing. Terdapat 14 sub sistem yang memiliki perannya masing-masing. Salah satu diantaranya adalah kegiatan rekognisi (Gambar 3). Rekognisi merupakan sebuah prestasi *non* kompetisi yang diraih oleh mahasiswa pada sebuah instansi PT dimana rekognisi diberikan oleh pemerintah, komunitas, organisasi, atau masyarakat [19]. Dalam proses pengelolaan data kegiatan Rekognisi, terdapat dua form yang berfungsi untuk proses penambahan (Gambar 4) dan ubah data rekognisi (Gambar 5). Sedangkan untuk proses lainnya seperti *delete*, disematkan pada *panel control* data (Gambar 6).



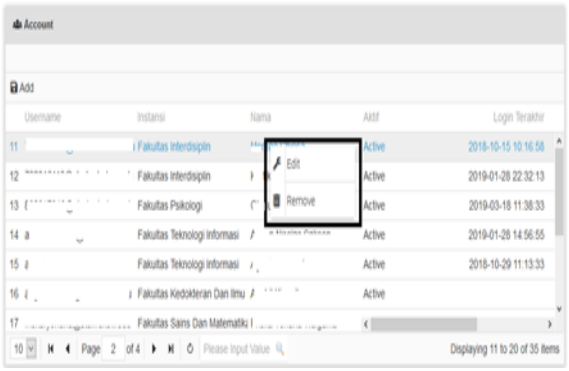
Gambar 3. Tampilan *STARS* Sub Sistem *Account Administrator*



Gambar 4. *Form* Tambahan *Account*



Gambar 5. *Form* Ubah *Account*



Gambar 6. *Panel Control Data*

Dari setiap sub sistem yang ada, dibekali dengan operasi *CRUD* (Tabel 1). Setiap operasi CRUD diwakili oleh empat *HTTP Method* yang telah ditentukan, hal ini membuktikan bahwa terdapat hubungan antara operasi *CRUD* dan *HTTP Method* yang digunakan. Pola tersebut dapat dibuktikan dengan Operasi *Create* menggunakan *Method POST*, Operasi *READ* menggunakan *Method GET*, Operasi *Update* Menggunakan *Method PUT* dan Operasi *Delete* menggunakan *Method DELETE*.

Table 1. Hubungan *CRUD* dan *HTTP Method*

	POST	*GET*	*PUT*	*DELETE*
Create	*v*	*x*	*x*	*x*
Read	x	v	x	x
Update	x	x	v	x
Delete	x	x	x	v

RESTFul yang dibangun untuk kebutuhan *STARS* untuk menjembatani komunikasi antara *client* dan *server*. Adapun komunikasi tersebut digunakan untuk akses terhadap setiap sumber daya yang dimiliki, dalam hal ini masing-masing sub-sistem yang ada padaaplikasi *STARS*. Model *RESTFul STARS* dikembangkan dengan pendekatan dari *REST*. *REST* dapat diuraikan dalam lima batasan, diantaranya: *Resource Identification, Connectedness, Uniform Interface, Self-Describing Messages* dan *Stateless Interactions*. Proses *request* dilakukan oleh *client* dengan menggunakan *jQuery ajax*. Penggunaan *ajax* pada setiap *request* dari pihak *client* karena *library ajax* yang ada saat ini telah mendukung multi operasi yang sering digunakan yakni CRUD. Terdapat empat jenis *request* yang dilakukan, yakni *GET, POST, PUT* dan *DELETE*. *GET* terdiri dari dua pola *GET*, yakni *GET* untuk memanggil banyak data atau multi (Gambar 7) dan *GET* untuk memanggil *single data* (Gambar 8). Hal ini dilakukan karena kebutuhan pemanggilan data, sebagai contoh *multiple* untuk *pagination* dan *single request data* untuk di-*edit* atau *update*.

```
//GET multiple data
$.ajax({
    url: "http://{domain}/accounts",
    type: 'GET',
    dataType: 'JSON',
    data: {
        perpage: 10,
        offset: 0,
        token: 'asdj4367r346q5weh3u7'
    },
    success: function (data, textStatus, jqXHR) {
        console.log(data);
    }
});
```

Gambar 7. *Ajax GET Multiple Data*

```
// GET single data
$.ajax({
    url: "http://{domain}/accounts",
    type: 'GET',
    dataType: 'JSON',
    data: {id: 1, token: 'asdj4367r346q5weh3u7'},
    success: function (data, textStatus, jqXHR) {
        console.log(data);
    }
});
```

Gambar 8. *Ajax GET Single Data*

Metode *POST* digunakan untuk proses menyimpan data (Gambar 9). Data yang disimpan harus didefinisikan satu-persatu dengan memanfaatkan '*datatype: 'JSON*".

```
// POST data
$.ajax({
    url: "http://{domain}/accounts",
    type: 'POST',
    dataType: 'JSON',
    data: {
        nama: 'mahasiswa',
        instansi: 1,
        nim: '123456789',
        token: 'asdj4367r346q5weh3u7'
    },
    success: function (data, textStatus, jqXHR) {
        console.log(data);
    }
});
```

Gambar 9. *Ajax POST Data*

Metode *PUT* digunakan untuk proses *update* data (Gambar 10). Sama halnya dengan *POST*, pada metode *PUT*, data yang diperbaharui harus didefinisikan menggunakan '*datatype: 'JSON*".

```
// PUT data
$.ajax({
    url: "http://{domain}/accounts",
    type: 'PUT',
    dataType: 'JSON',
    data: {
        nama: 'mahasiswa 1',
        instansi: 1,
        nim: '987654321',
        token: 'asdj4367r346q5weh3u7'
    },
    success: function (data, textStatus, jqXHR) {
        console.log(data);
    }
});
```

Gambar 10. *Ajax PUT Data*

Metode *DELETE* digunakan untuk proses *delete* data (Gambar 1). Sama halnya dengan *POST* dan *PUT*, metode *DELETE*, data yang diperbaharui harus didefinisikan menggunakan '*datatype: 'JSON"*.

```
// GELETE data
$.ajax({
    url: "http://{domain}/accounts",
    type: 'DELETE',
    dataType: 'JSON',
    data: {
        id: 1,
        token: 'asdj4367r346q5weh3u7'
    },
    success: function (data, textStatus, jqXHR) {
        console.log(data);
    }
});
```

Gambar 11. *Ajax Delete Data*

Mengacu pada ke-empat metode yang ada, maka dapat dilihat beberapa kesamaan yang *scripting* yang digunakan. Yang pertama adalah '*url*', yakni '*url*' harus memenuhi standar *Resource Identification* dan *Connectedness*, dimana tidak ada perbedaan antara ke-empat *url* dari masing-masing *method* yang ada. Selanjutnya '*type*' yang berfungsi untuk mendefinisikan method *HTTP* yang akan digunakan dalam hal ini adalah *Uniform Interface*. Selanjutnya adalah '*dataType'* yang berfungsi untuk mendefinisikan tipe data yang akan digunakan dalam melakukan *request* yakni tipe data '*JSON'*. Penggunaan '*dataType'* mengacu pada konsep *Self-Describing Messages*. Yang terakhir adalah mendefinisikan 'data'. Setiap proses *request* yang dilakukan, memiliki data yang berbeda-beda, namun didalamnya terdapat pasangan '*key/value*'. Salah satunya yakni '*key'* yakni '*token'* dan '*value'* atau konten dari *token* tersebut. Hal ini merupakan cerminan dari konsep *Stateless Interactions*.Model *RESTFul Stars* dikembangkan dengan pendekatan dari REST. Secara teoritis, terdapat Batasan-batasan yang dimiliki oleh *REST*, diantaranya: *Resource Identification, Connectedness, Uniform Interface, Self-Describing Messages* dan *Stateless Interactions*.

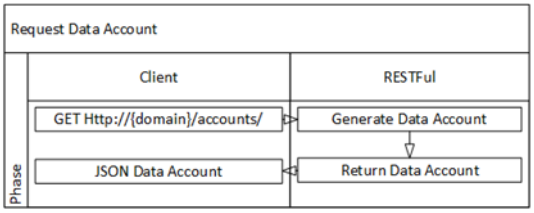
Resource Identification dapat diartikan bahwa setiap sumber daya harus dialamatkan [2][10]. Dengan kata lain bahwa masing-masing sumberdaya diwakili oleh *URI* yang berbeda-beda.

```
Http://{domain}/{resource}/
Http://{domain}/accounts/
```

Gambar 12. *Resource Interaction*

Dalam model *RESTFul STARS*, salah satu sub sistem yakni "*account*" berfungsi untuk manajemen data *account* pengguna. Hal ini mengartikan bahwa setiap sub sistem mewakili masing-masing sumber daya dan masing-masing sumber daya diwakili oleh *URI* yang berbeda-beda. Dengan demikian maka terdapat 14 *URI* yang akan digunakan dalam melakukan eksplorasi sumber daya, dimana hal ini didasarkan pada jumlah sub-sub sistem yang akan digunakan.

Connectedness dapat diartikan bahwa, dalam membangun interaksi dengan *service*, *client* harus mengikuti *URI* untuk menemukan sumber daya dari *service* [2][10].



Gambar 13. *Connectedness*

Connectedness diartikan bahwa dalam mengakses sumber daya, *client* diharuskan mengikuti setiap *URI* yang ada. Hal ini dikarenakan setiap sumber daya harus dialamatkan atau disebut sebagai *Resource Interaction*.

Uniform Interface artinya *client* harus menggunakan *HTTP Method* yang telah disesuikan untuk berinteraksi dengan *Service* [2][10].

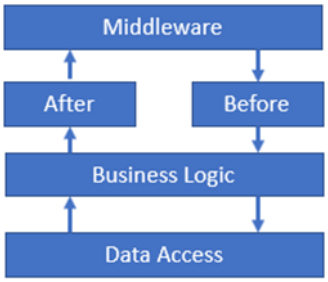
Dalam model *RESTFul* yang dibangun, digunakan operasi *CRUD* untuk proses manajemen data. Sebagai contoh, *data account* dapat di *Create* dengan menggunakan *HTTP method POST*, *Read* dengan menggunakan HTTP *Method GET*, *Update* dengan menggunakan *HTTP Method PUT* dan *Delete* dengan menggunakan *HTTP Method DELETE*. Hal ini dapat diartikan bahwa masing-masing operasi dari *CRUD* diwakili oleh masing-masing dengan *method* yang berbeda-beda namun dialamatkan dengan *URI* yang sama atau dapat juga diartikan bahwa satu *URI* dapat digunakan oleh beberapa operasi.

Self-Describing Messages dapat diartikan bahwa dalam mengekspos *resource* yang ada, *RESTFul* dapat menggunakan lebih dari satu satu format data, contohnya *XML, JSON, RDF*, dan lain-lain) [2][10] dengan demikian peranan *media type* yang didefinisikan melalui *header request* sangat lah penting, kerena *RESTFul* hanya mengembalikan format data yang di*request* melalui *media type*. Dalam model *RESTFul STARS*, format data yang digunakan hanyalah format data *JSON* dan beberapa tipe data *file* yakni *Docx, doc, PDF, JPG* dan *PNG*. Pemilihan format data *JSON* didasarkan atas kesederhanaan *JSON*.

Stateless Interactions yang berarti bahwa setiap *request* dari *client* harus lengkap, dalam arti bahwa semua informasi untuk melayani *request* dari *client* ke *server* harus berisikan setiap informasi yang dibutuhkan agar *request* tersebut dapat dipahami [4], dan tidak ada ketergantungan dengan *state* atau penanda dari *client*. Tidak ada status *session* yang ada di *server*, disimpan sepenuhnya di sisi klien. Jika akses ke sumber daya memerlukan otentikasi, maka klien perlu mengotentikasi dirinya dengan setiap *request*. Dalam penerapannya, setiap *request* yang dilakukan oleh *STARS* ke *RESTFul* selalu di lengkapi dengan token yang berfungsi sebagai penanda *client STARS*.

Tujuan limitasi penggunaan *HTTP Method* (terdiri dari enam sampai sembilan *Method HTTP* yang ada [18][20] pada model ini karena didasarkan pada kompabilitas masing-masing *browser* [21] dan dukungan implementasi *XML HTTP Request* (misalnya: Panggilan *AJAX*) di

semua *browser web* utama (*Internet Explorer, Mozilla Firefox, Safari, Google Chrome*, dan *Opera*). Selanjutnya, limitasi ke-empat *HTTP Method* yang digunakan mengacu pada operasi yang digunakan yakni *CRUD*.Tujuan limitasi tipe data pada proses *response* didasarkan atas kebutuhan penggunaan data pada aplikasi *STARS*, dimana *STARS* menggunakan beberapa *file* dokumen (*Docx, doc, PDF, JPG* dan *PNG*) sebagai *file* koleksi. Sedangkan *JSON* digunakan untuk parsing data. Tipe data json dipilih atas dasar kesederhanaan dari *JSON*. *JSON* adalah tipe data yang independen dan didasarkan pada koleksi pasangan *key / value* dan mempunyai *list value*. Struktur ini memungkinkan untuk digunakan dalam setiap bahasa pemrograman modern (Sitasi 18).



Gambar 14. Lapisan pada *RESFul STARS*

Dalam *RESTFul STARS*, setiap *request* yang dilakukan oleh *client* akan diekstrak (*before*) dan di*encode* (*after*) oleh *layer middleware*. Proses ekstrak *request* yang dimaksud adalah mendefinisikan *HTTP Method* yang digunakan, *request body* dan identifikasi *client* menggunakan token yang disematkan kedalam *request body* (Gambar 7, 8, 9, 10 dan 11). Ekstrak *request* tersebut dilakukan sebelum diteruskan ke *business logic*. Lapisan *business logic* berperan untuk menterjemahkan hasil ekstrasi dari *middleware*yang kemudian akan diteruskan ke lapisan *data access* dengan disesuaikan dengan proses bisnis yang diterapkan. *Business logic* dalam *RESTFul STARS* adalah berfungsi untuk membatasi *rule* akses bagi masing-masing grup user yang menggunakan aplikasi *STARS* dalam hal ini admin dan operator. Dimana masing-masing grup memiliki hak akses terhadap sumber daya yang dimiliki adalah berbeda-beda. Hasil yang diperoleh dari lapisan data *access* akan diproses dan dikembalikan pada lapisan *middleware*. Saat data yang diperoleh dari lapisan *business layer* diterima, maka lapisan middleware akan melakukan *encode* (*after*)data kedalam format data *JSON* yang kemudian akan diteruskan ke *client* sebagai respon.

4. Kesimpulan

Model *RESTFul STARS* dibangun dan diimplementasikan pada aplikasi *STARS*. Terdapat kesimpulan yang dapat ditarik berdasarkan hasil penelitian yang dilakukan, antara lain: Terdapat korelasi antara ke-empat operasi yakni *CRUD* dan *HTTP method* yakni *POST*, *GET*, *PUT* dan *DELETE*. Dalam implementasinya *RESTFul STARS* dapat dikembangkan mengacu pada lima batasan yang ada yakni *Uniform Interface, Connectedness, Self-Describing Messages* dan *Stateless Interactions*. Terdapat batasan-batasan yang digunakan dalam membangun *RESTFul STARS* yakni penggunaan *HTTP Method*, penggunaan JSON dan tipe file yang direkomendasikan.

Daftar Pustaka

[1] V. De Luca, I. Epicoco, D. Lezzi, And G. Aloisioa, "Grb_Wapi, A Restful Framework For Grid Portals," *Procedia Comput. Sci.*, Vol. 9, Pp. 459–468, 2012.
[2] R. T. Fielding, "Architectural Styles And The Design Of Network-Based Software Architectures," University Of California, Irvine, 2000.
[3] S. Price, P. A. Flach, S. Spiegler, C. Bailey, And N. Rogers, "Subsift Web Services And

Workflows For Profiling And Comparing Scientists And Their Published Works," *Futur. Gener. Comput. Syst.*, Vol. 29, No. 2, Pp. 569–581, 2013.
[4] C. M. Lewandowski, *A Brief Mindfulness Intervention On Acute Pain Experience: An Examination Of Individual Difference*. Southern Illinois University At Carbondale, 2015.
[5] Z. Aljazzaf, "Bootstrapping Quality Of Web Services," *J. King Saud Univ. Inf. Sci.*, Vol. 27, No. 3, Pp. 323–333, 2015.
[6] K. Mohamed And D. Wijesekera, "Performance Analysis Of Web Services On Mobile Devices," *Procedia Comput. Sci.*, Vol. 10, Pp. 744–751, 2012.
[7] D. Roman, J. Kopecký, T. Vitvar, J. Domingue, And D. Fensel, "Wsmo-Lite And Hrests: Lightweight Semantic Annotations For Web Services And Restful Apis," *J. Web Semant.*, Vol. 31, Pp. 39–58, 2015.
[8] F. Aijaz, S. Z. Ali, M. A. Chaudhary, And B. Walke, "Enabling High Performance Mobile Web Services Provisioning," In *2009 Ieee 70th Vehicular Technology Conference Fall*, 2009, Pp. 1–6.
[9] C. Rodríguez *Et Al.*, "Rest Apis: A Large-Scale Analysis Of Compliance With Principles And Best Practices," In *International Conference On Web Engineering*, 2016, Pp. 21–39.
[10] D. Guinard, M. Mueller, And V. Trifa, "Restifying Real-World Systems: A Practical Case Study In Rfid," In *Rest: From Research To Practice*, Springer, 2011, Pp. 359–379.
[11] J. Lasmono, A. P. Sari, E. Kuncoro, And I. Mujahidin, "Optimasi Kerja Peluncur Roket Pada Robot Roda Rantai Untuk Menentukan Ketepatan Sudut Tembak," *Jasiek (Jurnal Apl. Sains, Informasi, Elektron. Dan Komputer)*, 2019.
[12] F. M. Besson, "A Framework For Automated Testing Of Web Service Choreographies." University Of São Paulo, 2011.
[13] A. Navarro And A. Da Silva, "A Metamodel-Based Definition Of A Conversion Mechanism Between Soap And Restful Web Services," *Comput. Stand. Interfaces*, Vol. 48, Pp. 49–70, 2016.
[14] M. Athanasopoulos And K. Kontogiannis, "Extracting Rest Resource Models From Procedure-Oriented Service Interfaces," *J. Syst. Softw.*, Vol. 100, Pp. 149–166, 2015.
[15] R. Xiong, J. Wang, N. Zhang, And Y. Ma, "Deep Hybrid Collaborative Filtering For Web Service Recommendation," *Expert Syst. Appl.*, Vol. 110, Pp. 191–205, 2018.
[16] M. Wibowo, S. Suprayogi, And I. Mujahidin, "Rancang Bangun Sistem Pengamanan Rak Senjata M16 Menggunakan Rfid Dan Fingerprint," *Jasiek (Jurnal Apl. Sains, Informasi, Elektron. Dan Komputer)*, Vol. 1, No. 2, Pp. 134–142, 2019.
[17] M. Shodiq, R. Wongso, R. S. Pratama, And E. Rhenardo, "Implementation Of Data Synchronization With Data Marker Using Web Service Data," *Procedia Comput. Sci.*, Vol. 59, Pp. 366–372, 2015.
[18] T. A. S, A. Rabi', D. Minggu, And I. Mujahidin, "Frequency Hopping Video Real Time Untuk Pengamanan Data Pengintaian Operasi Inteligence Tni," *Jasiek (Jurnal Apl. Sains, Informasi, Elektron. Dan Komputer)*, 2019.
[19] S. Mariyam, I. Thalia, M. A. Firdausy, And A. Chusyairi, "Perancangan Sistem Informasi Point Non Akademik (E-Point) Pada Stikom Pgri Banyuwangi," In *Conference On Information Technology, Information Sytem And Electrical Engineering (Citisee), Stmik Amikom, Purwokerto*, 2017, Vol. 7.
[20] P. F. Tanaem, D. Manongga, And A. Iriani, "Restful Web Service Untuk Sistem Pencatatan Transaksi Studi Kasus Pt. Xyz," *J. Tek. Inform. Dan Sist. Inf.*, Vol. 2, Pp. 2443–2229, 2016.
[21] R. Subekti And F. Scholz, "Http Request Methods," *Mozilla Developer*, 2017. .
[22] W3schools, "Http Request Methods," *W3schools*, 2019. .
[23] R. T. Fielding *Et Al.*, "Hypertext Transfer Protocol -- Http/1.1," *Https://Www.W3.Org*, 1999. .