PERANCANGAN APPLICATION PROGRAMMING INTERFACE (API) BERBASIS WEB MENGGUNAKAN GAYA ARSITEKTUR REPRESENTATIONAL STATE TRANSFER (REST) UNTUK PENGEMBANGAN SISTEM INFORMASI ADMINISTRASI PASIEN KLINIK PERAWATAN KULIT

[1]Beni Adi Pranata, [2]Astria Hijriani dan [3]Akmal Junaidi

[1,2,3]Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Lampung
Jl. S. Brodjonegoro No. 1, Bandar Lampung, 35145
beniadipranata@gmail.com

Abstract

Application Programming Interface (API) is an interface built by system developer so some or entire functions of the system can programatically be accessed. Representational State Transfer (REST) is one of API development architectural style that uses Hypertext Transfer Protocol (HTTP) for data communication. This research implemented REST in developing API as the back-end of the skincare clinic patient information system. API was developed using Javascript Object Notation (JSON) as the standard format for data communication and JSON Web Token (JWT) as user authentication code. This research indicates that the development of API successfully performed on the patient administration of skin care clinic and implementation of REST makes it easy to develop API structures. This research produced REST API-based back-end for the patient administration information system of skin care clinic. API was tested in three stages: JWT testing on multiple back-end servers, API testing with Equivalence Partitioning and system functional testing.

Keywords: *application programming interface (api), hypertext transfer protocol (http), javascript object notation (json), json web token (jwt), representational state transfer (rest).*

1. Pendahuluan

Dalam bidang ilmu kesehatan, pemanfaatan komputer untuk mendukung prosedur dan cara-cara dalam penyelenggaraan pelayanan kesehatan [12] dikenal dengan istilah komputerisasi administrasi [3]. Dukungan komputer pada administrasi pelayanan kesehatan yang difasilitasi oleh pihak konsultan teknologi informasi umumnya berbentuk sistem informasi. Dalam pengembangannya, konsultan teknologi informasi sebaiknya memperhitungkan bahwa kasus-kasus yang dihadapi kliennya akan terus berkembang. Karena itu dibutuhkan sebuah konsep pengembangan yang baik supaya sebuah sistem informasi mampu beradaptasi. Memisahkan *logic* pada basis data dengan *logic* antarmuka dalam sebuah sistem informasi dianggap sebagai cara yang ideal untuk menanggulangi masalah ini. Dengan cara ini, masing-masing *logic* dapat dikembangkan secara terpisah dan dimungkinkan untuk mengakomodasi lebih banyak fungsi. Representasi fungsi-fungsi pada *logic* basis data yang diakses oleh *logic* pada antarmuka secara programmatis dikenal dengan istilah *Application Programming Interface* (API) [11].

Web merupakan sebuah ruang informasi yang digunakan secara global dan sumber informasinya diakses berdasarkan *Hyper Text Transfer Protocol* (HTTP) antara *client* dan *server* dengan konsep *request-response* [4]. Dalam pengembangan API diperlukan sebuah gaya arsitektur sebagai pedoman cara berhubungan antara *logic* basis data dengan *logic* antarmuka. Salah satu gaya arsitektur pengembangan API berbasis *web* yang menggunakan HTTP dalam komunikasi data adalah *Representational State Transfer* (REST) [1].

Penelitian ini mengimplementasikan gaya arsitektur REST dalam pengembangan API sebagai *back-end* sistem informasi administrasi pasien klinik perawatan kulit. API yang dikembangkan menggunakan *Javascript Object Notation* (JSON) [7] sebagai standar format dalam komunikasi data serta *JSON Web Token* (JWT) [5] sebagai kode otentikasi pengguna sistem. Penelitian ini menunjukkan bahwa pengembangan API berhasil dilakukan untuk administrasi pasien klinik perawatan kulit dan REST yang diterapkan mempermudah pengembangan struktur API. Penelitian ini menghasilkan *back-end* sistem informasi administrasi pasien klinik perawatan kulit berbasis REST API. API diuji dalam tiga tahap yakni pengujian otentikasi dengan JWT pada *back-end server* berjumlah banyak, pengujian API dengan metode *Equivalence Partitioning* dan pengujian fungsional sistem.

2. Metodologi

Pengembangan sistem pada penelitian ini dilaksanakan berdasarkan pendekatan-pendekatan baku yang mencakup cara-cara, teknik dan pemodelan dalam membangun sistem informasi. Hal ini dikenal dengan istilah *System Development Life Cycle* (SDLC) dan penelitian ini menggunakan SDLC tipe *Waterfall.* Tipe *Waterfall* digunakan karena dalam setiap tahapan penelitian mampu menerima perubahan dan pengembangan konsep dari tahapan sebelumnya, yang mana hal ini mendukung tujuan penelitian untuk membangun sistem yang mampu beradaptasi terhadap perkembangan proses bisnis diwaktu mendatang. Selain itu metode *Waterfall* juga telah terbukti berhasil dalam mengadaptasi penelitian dengan kasus serupa seperti "Sistem Perangkum Laporan Jaminan Pelayanan Kesehatan Medical Center PT. Central Pertiwi Bahari" [10] dan "*Token-Based Authentication Using JSON Web Token on* SIKASIR RESTful *Web Service*" [2].

Sebelum masuk ke tahap *Waterfall*, dilakukan studi literatur untuk mengumpulkan informasi penelitian-penelitian yang mengangkat topik pengembangan API dengan gaya arsitektur REST. Peneliti juga membangun komunikasi dan berdiskusi dengan *administrator* salah satu klinik kulit [8]. Kemudian pelaksanaan metode *Waterfall* dimulai dengan analisa kebutuhan sistem. Analisa kebutuhan dilakukan dengan mengembangkan literatur yang telah dikumpulkan berdasarkan proses bisnis administrasi pasien klinik kulit. Selanjutnya berdasarkan hasil analisa kebutuhan dilakukan desain *server*, desain *database*, desain kode program dan desain *web server*.

Tahap desain kemudian direalisasikan pada tahap implementasi. Tahap implementasi dimulai dengan membangun lingkungan pengembangan sistem. Selanjutnya mendesain struktur tabel, *function* dan *stored procedure* pada *database*. Dilanjutkan dengan mengembangkan *back-end* (*coding*) dan menyesuaikannya dengan *database*. Dan pada tahap akhir implementasi dibangun struktur *rewrite* pada *web server* untuk pemetaan akses API. Kemudian API diuji dengan bantuan

software Postman [9] berdasarkan tiga skenario pengujian, antara lain : pengujian otentikasi token, pengujian dengan metode *equivalent partitioning* [6] dan pengujian fungsional.

3. Pembahasan

Hasil dari penelitian ini dibagi menjadi tiga bagian. Pertama adalah struktur API, kedua hasil REST API yang memetakan *resource* dan ketiga adalah pengujian untuk memastikan kesesuaian hasil penelitian.

3.1. Struktur API

Pada penelitian ini, *resource* diklasifikasi secara terpola berdasarkan *path* dan *query string* dalam URL serta *request method* dalam HTTP *request*. Klasifikasi *resource* berdasarkan *path* dan *query string* dibagi menjadi tiga pola sebagai berikut :

1. Titik akses /resource?query=string yang digunakan untuk *request* mendapatkan daftar dan menambah data anggota dari *resource* yang dimaksud.
2. Titik akses /resource/id?query=string yang digunakan untuk *request* mendapatkan detail, mengubah dan menghapus *resource* berdasarkan nomor id *resource* yang dimaksud.
3. Titik akses /parent/id/resource?query=string yang digunakan untuk *request* daftar dan menambah data anggota dari *resource* berdasarkan id *parent* yang dimaksud.

Sementara klasifikasi berdasarkan *request method* dibagi menjadi empat jenis sebagai berikut :

1. *Request method* POST yang digunakan untuk penambahan *resource*.
2. *Request method* GET yang digunakan untuk mendapatkan daftar dari anggota *resource* dan detail dari anggota *resource*.
3. *Request method* PUT yang digunakan untuk mengubah *resource*.
4. *Request method* DELETE yang digunakan untuk menghapus *resource*.

Kemudian klasifikasi diatas dipetakan terhadap *resource*nya. Sebagian daftar API sebagai contoh yang menjelaskan hubungan antara *resource* dengan klasifikasinya terhadap *path* dan *query string* serta *request method* dapat dilihat pada Tabel 1.

Tabel 1 Contoh sebagian daftar API

No	Nama Proses	Request Method	URI	Keterangan
1	Post Registrasi	POST	/register	Mengirim email registrasi ke sistem
2	Registrasi	POST	/user	Mengkonfirmasi email registrasi ke sistem
3	Login	POST	/login	Membuat token untuk session
4	Post Token	POST	/token	Membuat token permanen
5	Post pasien	POST	/pasien	Menambah data pasien
6	List pasien	GET	/pasien	Menampilkan daftar pasien
7	Get pasien	GET	/pasien/<id_pasien>	Menampilkan detail pasien berdasarkan ID pasien

No	Nama Proses	Request Method	URI	Keterangan
8	Update pasien	PUT	/pasien/<id_pasien>	Mengubah detail pasien berdasarkan ID pasien
9	Delete pasien	DELETE	/pasien/<id_pasien>	Menghapus pasien berdasarkan ID pasien

Pada penelitian ini setiap *request* API yang masuk ke *back-end server* akan menghasilkan sebuah kembalian dengan format JSON. Kembalian tersebut akan dikelompokkan menjadi tiga bagian :

1. *Head* yang memuat informasi token.
2. *Body* yang memuat hasil dari eksekusi *database*.
3. *Tail* yang memuat pesan-pesan kesalahan.

Khusus untuk pengujian otentikasi tahap lanjutan, *head* juga akan memuat informasi *back-end server* yang memproses *request*.

3.2. REST API

Pada penelitian ini, REST API yang dihasilkan dipetakan kedalam bentuk tabel berdasarkan *resource* yang direpresentasikan. Untuk dapat memahami tabel REST API, berikut penjelasan dari setiap kolom dan tanda yang tersedia :

1. Kolom no sebagai nomor index dari parameter.
2. Kolom nama parameter sebagai nama dari parameter. Beberapa parameter yang tidak memiliki nomor index tidak dapat digunakan sebagai *order*. Selain itu terdapat empat parameter yang tidak akan memiliki index, parameter tersebut adalah *offset*, *order*, *page* dan *sort*. Keempat parameter ini merupakan parameter pembantu dalam operasi *list*, keterangannya sebagai berikut :
   - *offset* merupakan parameter jumlah data yang ingin ditampilkan.
   - *order* merupakan parameter *order* data berdasarkan nomor index parameter.
   - *page* merupakan parameter halaman pada data yang ingin ditampilkan.
   - *sort* merupakan parameter tipe pengurutan data dengan nilai 0 untuk *ascending* dan 1 untuk *descending*.
3. Kolom tipe data yang merupakan keterangan tipe data dari parameter beserta batasan maksimal dari nilai yang dapat ditetapkan pada parameter terkait.
4. Kolom parameter yang menginformasikan parameter yang wajib, dapat atau tidak boleh dikirimkan terhadap operasi tertentu. Informasi pada kolom ini diklasifikasikan sebagai berikut :
   - (v) yang berarti parameter wajib dikirimkan.
   - (x) yang berarti parameter tidak boleh dikirimkan.
   - (-) yang berarti parameter dapat dikirimkan (opsional).
   - (w) yang berarti parameter wajib dikirimkan melalui URI.
   - (u) yang berarti patameter dapat dikirimkan (opsional) melalui URI.
5. Kolom nilai yang menentukan nilai minimal, maksimal dan nilai *default* yang dapat diterapkan pada parameter terkait.

Kemudian tiga contoh hasil REST API dari penelitian yang dihasilkan dari penelitian ini dapat dilihat pada tabel 2 hingga tabel 4 berikut :

Tabel 2 User

No	Nama Parameter	Tipe Data	parameter				Nilai		
			post (registrasi)	post	post (lupa password)	post (reset password)	Min	Max	Default
1	id	bigint(20) unsigned	x	x	x	x			
2	email	varchar(50)	v	v	v	v	5		
	password	varchar(50)	x	v	x	v	8		
	token	varchar(256)	x	v	x	v	1		

Tabel 3 Token

No	Nama Parameter	Tipe Data	parameter						Nilai		
			login	logout	post	list	get	delete	Min	Max	Default
1	id	bigint(20) unsigned	x	-	x	x	v	v			
2	user_id	bigint(20) unsigned	x	-	x	-	x	x			
3	created	timestamp	x	-	x	x	x	x			<current>
	email	varchar(50)	v	-	v	x	x	x	5		
	password	varchar(50)	v	-	v	x	x	x	8		
	Ip	varchar(15)	x	-	-	x	x	x	7	15	<current>
	offset	tinyint(3) unsigned	x	x	x	-	x	x	1	250	250
	order	tinyint(2) unsigned	x	x	x	-	x	x	0		1
	page	int(10) unsigned	x	x	x	-	x	x	1		1
	sort	tinyint(1) unsigned	x	x	x	-	x	x	0	1	0

Tabel 4 Pasien

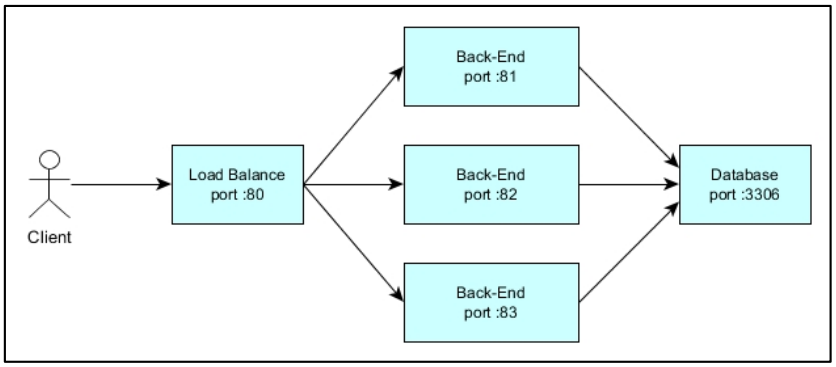
No	Nama Parameter	Tipe Data	Parameter					Nilai		
			Post	List	Get	Update	Delete	Min	Max	Default
1	Id	bigint(20) unsigned	x	x	v	v	v			
2	Nama	varchar(255)	v	x	x	-	x	3		
3	Tempat	varchar(50)	v	x	x	-	x	3		
4	Tanggal	date	-	x	x	-	x			
5	Alamat	varchar(255)	v	x	x	-	x	3		
6	kecamatan	varchar(50)	-	x	x	-	x			
7	kabupaten	varchar(50)	-	x	x	-	x			
8	Provinsi	varchar(50)	-	x	x	-	x			"Lampung"
9	Jk	tinyint(1) unsigned	-	x	x	-	x	0	1	0
10	Kawin	tinyint(1) unsigned	-	x	x	-	x	0	1	0
11	Berat	tinyint(3) unsigned	-	x	x	-	x			50
	Offset	tinyint(3) unsigned	x	-	x	x	x	1	250	250
	Order	tinyint(2) unsigned	x	-	x	x	x	0		1
	Page	int(10) unsigned	x	-	x	x	x	1		1
	Sort	tinyint(1) unsigned	x	-	x	x	x	0	1	0

3.3. Pengujian

Pengujian sistem dalam penelitian ini dibagi menjadi tiga tahap, yakni pengujian otentikasi, pengujian nilai masukan dengan *equivalence partitioning* dan pengujian fungsional sistem.

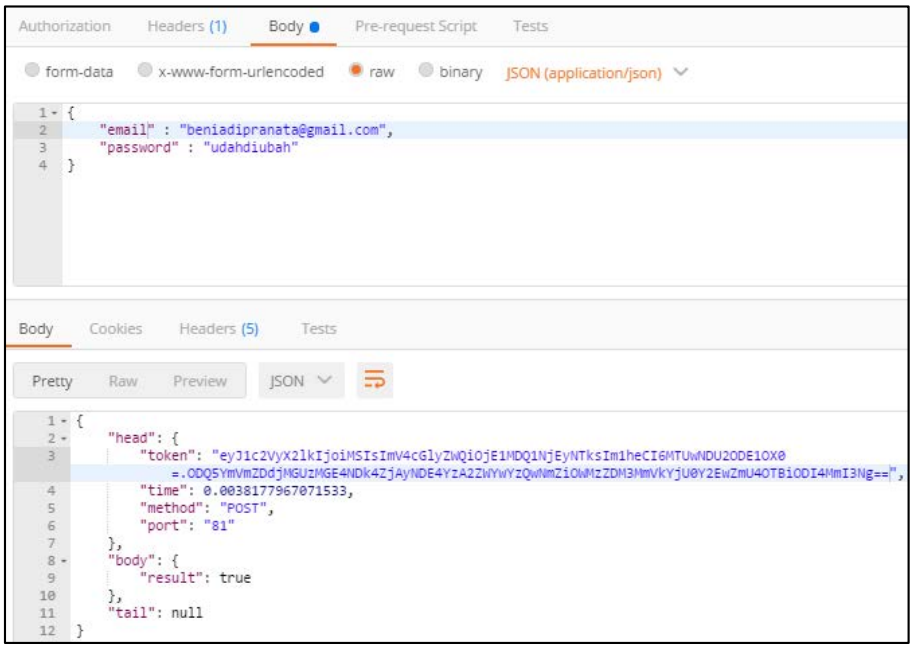
3.3.1. Pengujian Otentikasi

Pengujian otentikasi dimulai dengan membangun sebuah lingkungan pengujian berupa *load-balanced server* bertipe *round-robin* menggunakan Nginx dengan tiga *back-end server* seperti yang terlihat pada Gambar 1.



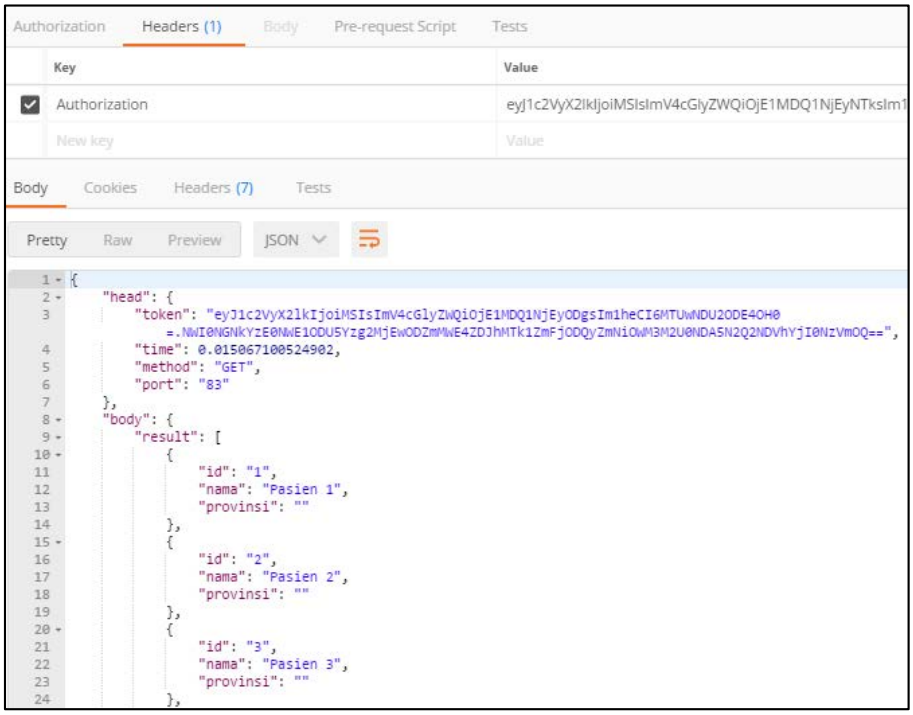
Gambar 1 Skema *load-balanced server*

Pengujian otentikasi dilakukan sebanyak tiga kali dengan melakukan *request login* pada salah satu *back-end server*, kemudian dilanjutkan dengan mengirim HTTP *request* kepada *back-end server* yang berbeda. Hal ini dilakukan untuk memastikan JWT mampu memenuhi tanggung jawab menggantikan sistem *session* konvensional sehingga otorisasi pengguna dapat dimengerti oleh *back-end server* yang berbeda meskipun otentikasi hanya dilakukan pada salah satu *back-end server* saja. *Request* yang dikirim kepada *back-end server* merupakan sampel *request* untuk mendapatkan *list* pasien saja. Untuk membantu identifikasi *back-end server* yang memproses *request*, khusus untuk pengujian ini telah ditambahkan informasi *port server* di bagian *head* pada nilai kembalian.



Gambar 2 Pengujian otentikasi diproses oleh *back-end server* pertama

Pada gambar 2 dapat dilihat bahwa otentikasi pengguna diproses pada *back-end server* pertama (port 81) dan telah menghasilkan token. Token yang dihasilkan selanjutnya akan digunakan sebagai token otorisasi pada *back-end server* yang lain.



Gambar 3 Pengujian otentikasi diproses oleh *back-end server* kedua

Pada gambar 3 dapat dilihat bahwa token yang sebelumnya dihasilkan pada *back-end server pertama* telah dimengerti oleh *back-end server ketiga* (port 83). Dan pada hasil kembalian diperoleh sebuah token baru pada bagian *head* serta daftar pasien pada bagian *body*.

3.3.2. Pengujian Equivalence Partitioning

Telah dilakukan pengujian dengan metode *equivalence partitioning* [5] seperti yang telah direncanakan pada metodologi. Contoh sebagian hasil pengujian dapat dilihat pada Tabel 5.

Tabel 5 Contoh sebagian hasil pengujian *equivalence partitioning*

Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil
	Validasi path	Request method GET untuk /pasien	Sukses mendapatkan list pasien	Berhasil

Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil
Path dan request method	Validasi request method	Request method GET untuk /pasien/1	Sukses mendapatkan detail pasien Id 1	Berhasil
		Request method GET untuk /pasien/1/invoice	Sukses mendapatkan list invoice untuk pasien Id 1	Berhasil
		Request method GET untuk /pasien/1/invoice/1	Error dengan kode 404 - not found	Berhasil
		Request method GET untuk /salah	Error dengan kode 404 - not found	Berhasil
		Request method POST untuk /pasien	Sukses menginput record pasien baru	Berhasil
		Request method GET untuk /pasien	Sukses mendapatkan list pasien	Berhasil
		Request method GET untuk /pasien/1	Sukses mendapatkan detail pasien Id 1	Berhasil
		Request method PUT untuk /pasien/1	Sukses mengedit data pasien dengan Id 1	Berhasil
		Request method DELETE untuk /pasien/1	Sukses menghapus data pasien dengan Id 1	Berhasil

3.3.2. Pengujian Fungsional Sistem

Pengujian fungsional sistem REST API dilakukan secara keseluruhan. Pengujian diklasifikasikan menjadi tiga bagian utama yakni pendaftaran pengguna, proses pembuatan token (pada proses *login* dan pembuatan token permanen) dan pengaksesan API. Contoh beberapa hasil pengujian dapat dilihat pada Tabel 6.

Tabel 6 Contoh sebagian hasil pengujian fungsional

No	Nama Proses	Keterangan	Hasil Pengujian
1	Post Registrasi	Mengirim email registrasi ke sistem	Berhasil
2	Registrasi	Mengkonfirmasi email registrasi ke sistem	Berhasil
3	Login	Membuat token untuk session	Berhasil
4	Post Token	Membuat token permanen	Berhasil
5	Post pasien	Menambah data pasien	Berhasil
6	List pasien	Menampilkan daftar pasien	Berhasil
7	Get pasien	Menampilkan detail pasien berdasarkan ID pasien	Berhasil
8	Update pasien	Mengubah detail pasien berdasarkan ID pasien	Berhasil
9	Delete pasien	Menghapus pasien berdasarkan ID pasien	Berhasil

4. Kesimpulan

Dalam penelitian ini gaya arsitektur REST berhasil diimplementasikan pada sistem administrasi pasien klinik perawatan kulit. Gaya arsitektur REST pada API menggunakan *path*, *query* dan *request method* dalam mengklasifikasi *resource*. Kemudian pengujian otentikasi REST API dalam penelitian ini memperoleh keberhasilan penuh karena JWT telah dimengerti oleh semua *back-end server*. Begitu pula untuk pengujian masukan sistem dengan metode *equivalence partitioning* dan pengujian fungsional sistem.

Hasil dari penelitian ini butuh pengembangan lebih lanjut sebelum dapat digunakan oleh awam, khususnya untuk pengembangan sistem antarmuka. Sebagai saran dari peneliti, dalam pengembangan REST API yang cara akses *resource*nya memiliki banyak kesamaan pola, akan lebih baik jika kode program dibuat berstruktur dengan *class* dan fungsi.

Refference

[1] Fielding, Roy Thomas. 2000. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.
[2] Haekal, Muhamad dan Eliyani. 2016. Token-based authentication using JSON Web Token on SIKASIR RESTful Web Service. International Conference on Informatics and Computing (ICIC).
[3] Haryadi, Hendi. 2009. Administrasi Perkantoran untuk Manajer dan Staf. Jakarta: Transmedia Pustaka.
[4] Hidayatullah, Priyanto dan Jauhari Khairul Kawistara. 2015. Pemrograman Web. Bandung: Informatika Bandung.
[5] Jones, Michael B. dkk. 2015. JSON Web Token (JWT). Internet Engineering Task Force (IETF), Mei 2015.
[6] Jovanoic, Irena. 2009. Software Testing Methods and Techniques. The IPSI BgD Transactions on Internet Research.
[7] Kleppmann, Martin dan Alastair R. Beresford. 2017. A Conflict-Free Replicated JSON Datatype. University of Cambridge Computer Laboratory, Cambridge.
[8] Riani, Istiasih Fajar. 2017. Wawancara Administrasi Pelayanan Kesehatan. Lampung.
[9] Postdot Technology. 2017. Postman is the most complete API Development Environment. [Online]. San Francisco. Tersedia: https://www.getpostman.com/postman [13 September 2017].
[10] Pranata, Beni Adi. 2014. Sistem Perangkum Laporan Jaminan Pelayanan Kesehatan Medical Center PT. Central Pertiwi Bahari. Lampung.
[11] Rama, Girish Maskeri dan Avinash Kak. 2013. Software – Practice and Experience. New Jersey: Wiley Online Library.
[12] Republik Indonesia. 2009. Undang-Undang No. 36 Tahun 2009 tentang Kesehatan. Jakarta: Sekretariat Negara.