

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

(Skripsi)

BENI ADI PRANATA



**JURUSAN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
2017**

ABSTRAK

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

Oleh

BENI ADI PRANATA

Application Programming Interface (API) merupakan antarmuka yang dibangun oleh pengembang sistem supaya sebagian atau keseluruhan fungsi sistem dapat diakses secara programatis. Sementara *Representational State Transfer* (REST) merupakan salah satu gaya arsitektur dari pengembangan API yang menggunakan *Hypertext Transfer Protocol* (HTTP) dalam melakukan komunikasi data. 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) sebagai standar format dalam komunikasi data serta *JSON Web Token* (JWT) sebagai kode otentikasi pengguna pengguna. Penelitian ini menunjukkan bahwa pengembangan API berhasil dilakukan pada 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 JWT pada back-end server berjumlah banyak, pengujian API dengan metode *Equivalence Partitioning* dan pengujian fungsional sistem.

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

ABSTRACT

WEB BASED APPLICATION PROGRAMMING INTERFACE (API) DESIGN USING REPRESENTATIONAL STATE TRANSFER (REST) ARCHITECTURAL STYLE FOR SKIN CARE CLINIC PATIENT ADMINISTRATION INFORMATION SYSTEM DEVELOPMENT

By

BENI ADI PRANATA

Application Programming Interface (API) is an interface built by system developer so some or entire functions of the system can programmatically 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), representational state transfer (rest), hypertext transfer protocol (http), javascript object notation (json), json web token (jwt).

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

Oleh

Beni Adi Pranata

Skripsi

Sebagai Salah Satu Syarat Untuk Memperoleh Gelar
SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam



**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2017**

Judul Skripsi

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

Nama Mahasiswa

: **Beni Adi Pranata**

No. Pokok Mahasiswa : 1017032020

Jurusan

: Ilmu Komputer

Fakultas

: Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

 **Dr. rer. nat. Akmal Junaidi, S.Si., M.Sc.**  **Astria Hijriani, S.Kom., M.Kom.**
NIP 19710129 199702 1 001 NIP 19810308 200812 2 002

2. Mengetahui

Ketua Jurusan Ilmu Komputer
FMIPA Universitas Lampung


Dr. Ir. Kurnia Muludi, M.S.Sc.
NIP 19640616 198902 1 001

MENGESAHKAN

1. Tim Penguji

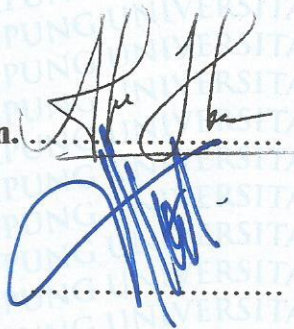
Ketua

: Dr. rer. nat. Akmal Junaidi, S.Si., M.Sc.



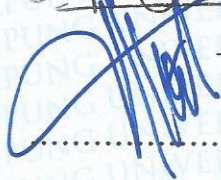
Sekretaris

: Astria Hijriani, S.Kom., M.Kom.

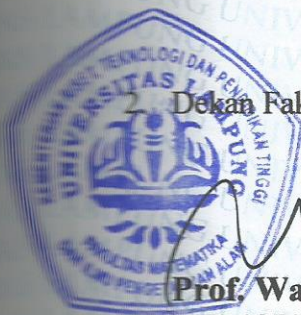


Penguji

Bukan Pembimbing : Didik Kurniawan, S.Si., M.T.



2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam



Prof. Warsito, S.Si. D.E.A., Ph.D.

NIP 19710212 199512 1 001

Tanggal Lulus Ujian Skripsi : 15 September 2017

PERNYATAAN

Saya yang bertanda tangan di bawah ini menyatakan dengan sebenar-benarnya bahwa skripsi yang telah saya buat dengan judul "**Perancangan *Application Programming Interface* (API) Berbasis *Web* Menggunakan Gaya Arsitektur *Representational State Transfer* (REST) Untuk Pengembangan Sistem Informasi Administrasi Pasien Klinik Perawatan Kulit**" merupakan karya orisinil saya sendiri dan bukan hasil karya orang lain. Semua hasil tulisan yang tertuang dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung.

Demikianlah surat pernyataan ini saya buat tanpa ada paksaan dari pihak manapun. Jika dikemudian hari terbukti bahwa skripsi ini merupakan hasil karya pihak lain, saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, Desember 2017



Beni Adi Pranata
NPM. 1017032020

RIWAYAT HIDUP



Penulis dilahirkan di kota Pringsewu provinsi Lampung pada tanggal 11 bulan Juli tahun 1992 sebagai anak kedua dari tiga bersaudara dengan ayah bernama Dolfi dan ibu bernama Lucia Ety Widayati.

Pendidikan formal penulis dimulai pada tahun 1997 di taman kanak-kanak (TK) Xaverius Pringsewu. Penulis melanjutkan pendidikan dasar di sekolah dasar (SD) Fransiskus Pringsewu pada tahun 1998 dan lulus pada tahun 2004. Kemudian dilanjutkan di sekolah menengah pertama (SMP) Xaverius Pringsewu pada tahun 2004 dan lulus pada tahun 2007. Kemudian untuk jenjang berikutnya penulis melanjutkan pendidikan di sekolah menengah atas (SMA) Xaverius Pringsewu pada tahun 2007 dan lulus pada tahun 2010.

Pada tahun 2010 penulis terdaftar sebagai mahasiswa program studi Ilmu Komputer jurusan Matematika fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung melalui jalur seleksi nasional masuk perguruan tinggi negeri (SNMPTN).

PERSEMBAHAN

Dengan mengucapkan syukur kepada Tuhan Yang Maha Esa, kupersembahkan karya
kecilku ini secara penuh untuk ibuku tercinta :

Lucia Ety Widayati

Yang tak pernah padam pelita kasih dan harapnya untuk kelulusan studiku di
Universitas Lampung.

Yang tak pernah henti maupun berenggan memanjatkan doa kepada Tuhan Yang
Maha Esa demi kesuksesanku.

MOTTO

“Great minds discuss ideas, average minds discuss events, small minds discuss people.”

– Eleanor Roosevelt

“The programmers of tomorrow are the wizards of the future. You're going to look like you have magic powers compared to everybody else.”

– Gabe Newell

“Education is the most powerful weapon which you can use to change the world.”

– Nelson Mandela

SANWACANA

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, sebab rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penelitian ini dan menuangkannya dalam bentuk skripsi.

Skripsi ini disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer di Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung. Judul skripsi ini adalah **“Perancangan *Application Programming Interface* (API) Berbasis Web Menggunakan Gaya Arsitektur *Representational State Transfer* (REST) Untuk Pengembangan Sistem Informasi Administrasi Pasien Klinik Perawatan Kulit”**.

Sangat banyak kesulitan-kesulitan yang penulis hadapi dalam penelitian dan penulisan skripsi ini. Namun berkat motivasi dan bantuan dari berbagai pihak, akhirnya penulis mampu menyelesaikan penelitian dan penulisan skripsi ini. Untuk itu dalam kesempatan ini penulis mengucapkan terimakasih yang sebesar-besarnya kepada :

1. Ibuku tercinta Lucia Ety Widayati yang tak pernah lelah memberikan motivasi, dukungan dan memanjatkan doa demi terselesaikannya studiku, khususnya dalam penyelesaian skripsi ini. Kakak yang kuhormati Gemara

Adi Pratama serta adikku yang kusayangi Bernadetha Belva Arjanti yang selalu memberikan dukungan dan motivasi untuk menyelesaikan studiku.

2. Kekasihku tercinta Istiasih Fajar Riani yang selalu memberikan dukungan dan motivasi untuk melanjutkan dan menyelesaikan studiku. Serta tak lupa atas perannya sebagai narasumber dalam skripsi ini.
3. Astria Hijriani, S.Kom., M.Kom selaku Koordinator Skripsi dan Dosen Pembimbing yang selalu mengingatkan dan menempehkan mentalku sehingga skripsi ini dapat diselesaikan dengan baik dan tepat waktu.
4. Dr. rer. nat. Akmal Junaidi, M.Sc. selaku Dosen Pembimbing Utama yang selalu mengingatkan dan memberikan kontribusi besar pada pengembangan materi sehingga skripsi ini dapat diselesaikan dengan baik dan tepat waktu.
5. Didik Kurniawan, S.Si., MT selaku Dosen Penguji dan Sekertaris Jurusan Ilmu Komputer yang telah memberikan banyak masukan dalam perbaikan skripsi ini serta memberikan bantuan dalam administrasi penyelesaian skripsi.
6. Teman seperjuanganku dalam menyelesaikan skripsi Togu Christian Situmorang yang selama ini telah banyak membantu dalam penyelesaian skripsi ini, khususnya dalam administrasi penyelesaian skripsi.
7. Seluruh Dosen dan Staff Jurusan Ilmu Komputer serta rekan-rekan mahasiswa Ilmu Komputer.
8. Seluruh Dosen dan Staff Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.
9. Almamater tercinta.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, tetapi penulis berharap skripsi ini dapat memberikan nilai manfaat bagi semua pihak yang membutuhkannya.

Bandar Lampung, Desember 2017

Beni Adi Pranata

DAFTAR ISI

	Halaman
HALAMAN JUDUL	i
LEMBAR PERSETUJUAN	ii
LEMBAR PENGESAHAN	iii
LEMBAR PERNYATAAN	iv
RIWAYAT HIDUP	v
PERSEMBAHAN.....	vi
MOTTO	vii
SANWACANA	viii
DAFTAR ISI.....	xi
DAFTAR GAMBAR.....	xv
DAFTAR TABEL	xvi
BAB I PENDAHULUAN	
1.1 Latar Belakang	1
1.2 Rumusan Masalah	4
1.3 Batasan Masalah.....	4
1.4 Tujuan.....	5
1.5 Manfaat.....	6
BAB II TINJAUAN PUSTAKA	
2.1 Administrasi	7
2.2 Pelayanan Kesehatan	7

2.3 Rekam Medis.....	7
2.4 Teknologi.....	8
2.5 Data	8
2.6 Teknologi Informasi	8
2.7 Sistem	9
2.8 Sistem Informasi.....	10
2.9 <i>Hyper Text Transfer Protocol</i> (HTTP).....	10
2.10 <i>Web</i>	11
2.11 <i>Web Server</i>	11
2.12 <i>Uniform Resource Locator</i> (URL)	12
2.13 PHP.....	12
2.14 <i>Database</i>	13
2.15 MySQL.....	13
2.16 <i>Application Programming Interface</i> (API)	14
2.17 Web API.....	14
2.18 Javascript <i>Object Notation</i> (JSON)	14
2.19 <i>Representational State Transfer</i> (REST)	15
2.20 <i>JSON Web Token</i> (JWT)	17
2.21 <i>Postman</i>	18
2.22 <i>Equivalence Partitioning</i> (EP)	18

BAB III METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian	19
---------------------------------------	----

3.2 Metode Penelitian.....	19
3.2.1 Studi Literatur	21
3.2.2 Pengumpulan Data.....	21
3.2.3 Analisis	21
3.2.4 Desain	35
3.2.5 Implementasi.....	39
3.2.6 Pengujian	40

BAB IV HASIL DAN PEMBAHASAN

4.1 Struktur API	47
4.1.1 Klasifikasi <i>Resource</i> Berdasarkan <i>Path</i> dan <i>Query String</i>	47
4.1.2 Klasifikasi <i>Resource</i> Berdasarkan <i>Request Method</i>	48
4.1.3 Daftar API.....	48
4.1.4 Struktur Kembalian.....	52
4.2 Titik Akses dan Parameter API	53
4.2.1 User.....	54
4.2.2 <i>Token</i>	55
4.2.3 <i>Resource</i>	58
4.3 Pengujian Sistem	65
4.3.1 Pengujian Otentikasi	65
4.3.2 Pengujian Nilai Masukan dengan Equivalence Partitioning	75
4.3.3 Pengujian Fungsional Sistem.....	81

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan.....	91
5.2 Saran	92
DAFTAR PUSTAKA	93

DAFTAR GAMBAR

	Halaman
2.1 Elemen pada URL	12
3.1 Tahap penelitian metode <i>waterfall</i>	20
3.2 Alur umum proses bisnis klinik perawatan kulit dan kelamin	23
3.3 <i>Flowchart</i> proses bisnis administrasi rekam medis	24
3.4 Entitas	25
3.5 Atribut entitas	26
3.6 Kardinalitas entitas	27
3.7 Diagram relasi entitas	28
3.8 Analisis kandidat <i>database</i>	29
3.9 Analisis kandidat API tahap pertama	30
3.10 Analisis kandidat API tahap kedua	31
3.11 Analisis kandidat API tahap ketiga	32
3.12 Analisis kandidat API tahap keempat	32
3.13 Desain <i>server</i>	35
3.14 Struktur tabel <i>database</i>	36
3.15 Alur <i>request database</i>	37
3.16 Struktur folder kode program	38
4.1 Timeline validitas <i>token session</i>	56
4.2 Skema <i>load-balanced server</i>	65
4.3 <i>Login</i> pada <i>back-end server</i> pertama	66

4.4 <i>Request list</i> pasien pada <i>back-end server</i> kedua	67
4.5 <i>Request list</i> pasien pada <i>back-end server</i> ketiga	68
4.6 <i>Login</i> pada <i>back-end server</i> kedua	69
4.7 <i>Request list</i> pasien pada <i>back-end server</i> pertama	70
4.8 <i>Request list</i> pasien pada <i>back-end server</i> ketiga	71
4.9 <i>Login</i> pada <i>back-end server</i> ketiga	72
4.10 <i>Request list</i> pasien pada <i>back-end server</i> pertama	73
4.11 <i>Request list</i> pasien pada <i>back-end server</i> kedua	74
4.12 Proses registrasi <i>email</i>	81
4.13 <i>Token</i> registrasi pada <i>email</i>	81
4.14 Registrasi pengguna	82
4.15 <i>Login</i>	83
4.16 Pembuatan <i>token</i> permanen	84
4.17 Penggunaan <i>token</i> pada <i>software Postman</i>	84
4.18 Pengaturan parameter pada <i>software Postman</i>	85

DAFTAR TABEL

	Halaman
3.1 Kandidat API.....	33
3.2 Rencana pengujian <i>equivalence partitioning</i>	39
4.1 Daftar API	47
4.2 Parameter <i>user</i>	54
4.3 Parameter <i>token</i>	57
4.4 Parameter pasien	58
4.5 Parameter <i>invoice</i>	58
4.6 Parameter dokter	59
4.7 Parameter perawat.....	60
4.8 Parameter konsultasi	60
4.9 Parameter produk	61
4.10 Parameter jasa	61
4.11 Parameter tindakan.....	62
4.12 Parameter penanggung jawab	62
4.13 Parameter asisten.....	63
4.14 Parameter <i>item</i>	64
4.15 Hasil pengujian <i>equivalence partitioning</i>	74
4.16 Hasil pengujian API	86

BAB I

PENDAHULUAN

1.1 Latar Belakang

Ilmu komputer yang terus berkembang mempermudah penerapan teknologi informasi pada bidang ilmu lain. Tidak hanya digunakan untuk mendukung perkembangan bidang keilmuannya, teknologi informasi seringkali memfasilitasi penerapan bidang ilmu lain dalam kehidupan nyata. Sebab itu, secara tidak langsung ilmu komputer dan teknologi informasi berperan penting dalam kehidupan manusia.

Tidak terkecuali pada bidang ilmu kesehatan, teknologi informasi dapat memfasilitasi penyelenggaraan pelayanan kesehatan. Pemanfaatan komputer untuk mendukung prosedur dan cara-cara dalam penyelenggaraan pelayanan kesehatan pada klinik perawatan kulit sebagai salah satunya contohnya. Prosedur dan cara-cara ini lebih dikenal dengan istilah administrasi. Selain memberikan prosedur-prosedur yang tetap, peran administrasi juga vital karena bertanggung jawab menghubungkan proses bisnis yang satu dengan yang lain. Karena itu, pemanfaatan komputer menjadi penting karena dapat meningkatkan kapasitas administrasi, khususnya untuk jumlah besar yang sukar ditangani oleh sumber daya manusia.

Biasanya sebuah konsultan teknologi informasi mengembangkan produk sistem informasi dalam memfasilitasi konsumennya. Namun sangat disayangkan, banyak sistem administrasi yang telah dikembangkan hanya dibuat berdasarkan kebutuhan administrasi pada waktu dan tempat tertentu. Seringkali konsultan teknologi informasi lupa bahwa perkembangan bidang ilmu kesehatan di masa mendatang akan mendukung diagnosa yang semakin spesifik dengan ragam masalah yang terus bertambah serta solusi yang terus berkembang. Pengembangan sistem informasi semacam ini mengakibatkan kode program yang ada tidak dapat memfasilitasi kebutuhan administrasi dalam jangka panjang, baik karena proses bisnis yang tidak lagi sesuai untuk difasilitasi maupun jumlah permintaan yang tidak lagi sanggup ditangani.

Sudut pandang penyelenggara pelayanan kesehatan tentunya mengharapkan masalah ini bisa diselesaikan dengan menghadirkan kembali konsultan teknologi informasi terkait untuk menyesuaikan sistem informasi dengan masalah administrasi yang telah berkembang. Akan tetapi memahami kembali logika-logika berbentuk kode pemrograman dalam sebuah sistem informasi yang dibangun dengan struktur yang terpaku pada kondisi tertentu bukanlah hal mudah, apalagi bagi konsultan teknologi informasi yang bukan pengembang sistem informasi terkait. Pada kasus ini, umumnya sebuah konsultan teknologi informasi lebih memilih untuk mengembangkan sistem baru dengan anggaran pengembangan yang baru pula. Hal ini berdampak pada kerugian waktu dan biaya bagi penyelenggara klinik perawatan kulit.

Sebagai langkah untuk menanggulangi masalah-masalah diatas, peneliti menganggap perlu dikembangkan sebuah sistem administrasi dengan basis *back-end* yang baik. *Back-end* yang dimaksud harus memiliki kemampuan untuk menerima perubahan-perubahan yang akan terjadi pada proses bisnis yang difasilitasi. Dan peneliti menilai hal ini dapat dicapai dengan menerapkan konsep sistem berbasis *Application Programming Interface* (API) dengan gaya arsitektur *Representational State Transfer* (REST) pada pengembangan *back-end* sistem administrasi.

Sistem informasi berbasis API memungkinkan sebuah *back-end* dimanfaatkan dengan cara yang lebih luas karena *logic* pada sistem dengan *logic* pada antarmukanya terpisah. Sistem informasi berbasis API juga akan mempermudah sebuah sistem untuk berkolaborasi dengan sistem lain. Hal ini jadi sangat berguna jika pada waktu mendatang proses bisnis yang difasilitasi menuntut interaksi dengan sistem lain, atau minimal akan sangat berguna pada pengembangan modul-modul baru pada sistem informasi terkait.

Begitu pula dengan gaya arsitektur REST yang diterapkan pada pengembangan API, selain meningkatkan fleksibilitas API dalam menerima perubahan dan penyesuaian, sifat orientasi *resource* yang dimilikinya juga memberikan struktur yang lebih mudah dipahami oleh pengembang-pengembang selanjutnya. Hal ini akan menuntun kerangka berpikir sebuah konsultan teknologi informasi untuk memelihara dan mengembangkan sistem yang telah ada daripada terus-menerus mengembangkan sistem informasi baru dengan kasus serupa.

Gagasan-gagasan diatas menjadi alasan yang kuat untuk melakukan penelitian “Perancangan *Application Programming Interface* (API) Berbasis *Web* Menggunakan Gaya Arsitektur *Representational State Transfer* (REST) Untuk Pengembangan Sistem Informasi Administrasi Pasien Klinik Perawatan Kulit” yang diharapkan mampu menjadi solusi yang lebih baik dalam pengembangan sistem administrasi pelayanan kesehatan.

1.2 Rumusan Masalah

Berdasarkan latar belakang, rumusan masalah dalam penelitian ini merupakan alasan-alasan yang melatarbelakangi penerapan konsep sistem berbasis API dengan gaya arsitektur REST dalam membangun sebuah *back-end* sistem informasi administrasi pasien klinik perawatan kulit. Alasan-alasan tersebut antara lain :

1. Struktur kode pemrograman yang lebih mudah dimengerti untuk pengembang sistem di waktu mendatang.
2. Fleksibilitas sistem informasi untuk menerima perkembangan proses bisnis administrasi pasien klinik perawatan kulit di waktu mendatang.
3. Skalabilitas sistem informasi untuk meningkatkan aksesibilitas sistem terhadap jumlah permintaan yang semakin meningkat.

1.3 Batasan Masalah

Demi menjaga kualitas hasil penelitian, telah ditentukan batasan-batasan masalah sebagai fokus dalam penelitian ini. Secara umum batasan masalah dalam penelitian ini mencakup penerapan konsep sistem informasi berbasis API dengan

gaya arsitektur REST dalam membangun sebuah *back-end* sistem informasi untuk administrasi pasien pada sebuah klinik perawatan kulit. Fokus lain diluar batasan masalah tidak dibahas secara mendalam dan digunakan untuk memperkokoh konsep-konsep penelitian.

Karena fokus utama pada penelitian ini merupakan penerapan konsep sistem informasi berbasis API dengan gaya arsitektur REST, peneliti akan membatasi penerapan tersebut pada satu kasus administrasi pasien saja dan bukan seluruhnya. Adapun administrasi pasien diangkat dari proses bisnis sebuah klinik perawatan kulit secara umum. Peneliti membatasi penerapan konsep sistem informasi berbasis API dan gaya arsitektur REST hanya pada proses bisnis rekam medis pasien klinik perawatan kulit. Dan yang dimaksud rekam medis dalam penelitian ini adalah sebagai berikut :

1. Administrasi identitas pasien.
2. Administrasi konsultasi yang pernah dijalani pasien.
3. Administrasi produk berupa jasa yang pernah dikonsumsi.
4. Administrasi produk berupa barang yang pernah dikonsumsi.

1.4 Tujuan

Adapun tujuan yang ingin dicapai dalam penelitian ini adalah untuk membuktikan bahwa penerapan konsep sistem informasi berbasis API dengan gaya arsitektur REST dalam membangun sebuah *back-end* sistem administrasi pasien klinik perawatan kulit merupakan sebuah cara yang lebih baik untuk menghasilkan produk sistem informasi yang dapat dimanfaatkan secara berkesinambungan.

1.5 Manfaat

Karena penelitian ini ditujukan untuk menghasilkan sebuah *back-end* sistem informasi dengan konsep API bergaya arsitektur REST yang memfasilitasi proses bisnis rekam medis pada sistem informasi administrasi pasien klinik perawatan kulit, peneliti memberikan gambaran manfaat sebagai berikut :

1. Sistem *back-end* hasil penelitian dapat digunakan secara langsung sebagai kerangka sistem administrasi pasien klinik perawatan kulit.
2. Dapat dijadikan rujukan literatur dan obyek pertimbangan secara umum dalam pengembangan sistem informasi dengan kasus sejenis.
3. Dapat digunakan sebagai bahan evaluasi dalam penelitian-penelitian selanjutnya, khususnya yang berhubungan dengan sistem informasi administrasi, pengembangan sistem dengan konsep API berbasis *Web* dan gaya arsitektur REST.

BAB II

TINJAUAN PUSTAKA

2.1 Administrasi

Administrasi adalah proses sistematis dalam penyusunan serta pencatatan data dan informasi oleh individu maupun kelompok. Proses ini bertujuan mempermudah pihak yang berhak dalam memperoleh data dan informasi yang telah disusun dan dicatat (Haryadi, 2009).

2.2 Pelayanan Kesehatan

Pelayanan kesehatan adalah upaya individu maupun kelompok dalam suatu organisasi yang diselenggarakan untuk mempromosikan peningkatan kualitas kesehatan serta mencegah dan menyembuhkan penyakit (Republik Indonesia, 2009).

2.3 Rekam Medis

Rekam medis merupakan berkas tentang identitas pasien, pemeriksaan, pengobatan, tindakan dan pelayanan yang berupa laporan hasil pemeriksaan maupun catatan observasi dan pengobatan yang dapat menunjang pelayanan kesehatan yang dibuat oleh tenaga medis dalam pelayanan kesehatan tertentu.

Rekam medis dapat berbentuk foto radiologi, gambar dan rekaman elektro-diagnostik (Republik Indonesia, 2008).

2.4 Teknologi

Teknologi merupakan produk dari hasil cipta, karsa dan pemikiran manusia pada berbagai bidang kehidupan. Produk teknologi yang umumnya kita gunakan dalam kehidupan sehari-hari misalnya teknologi telekomunikasi, teknologi transportasi, teknologi pertanian, dan lain-lain (Pratama, 2016).

2.5 Data

Secara singkat, data dapat didefinisikan sebagai fakta atau kejadian. Data baru akan bernilai setelah diolah bersama dengan data lain menggunakan teknologi dan diverifikasi oleh sistem pengolahnya (Suryantara, 2017).

2.6 Teknologi Informasi

Teknologi Informasi (TI) yang dalam bahasa Inggris dikenal dengan istilah *Information Technology* (IT) secara sederhana merupakan teknologi yang diciptakan untuk mengolah data menjadi informasi yang bernilai yang bermanfaat menunjang kehidupan manusia. Menurut Adelman (2000), terdapat lima macam pekerjaan yang mampu dilakukan teknologi terhadap informasi demi menunjang kehidupan manusia. Lima pekerjaan tersebut adalah membuat, mengubah, menyimpan, mengomunikasikan dan menyebarkan informasi.

Menurut Pratama (2016), pemanfaatan komputer merupakan hal mutlak dalam teknologi informasi. Karena dalam proses pengolahan data menjadi informasi

yang berarti, berfungsi, bernilai dan bermanfaat, dibutuhkan komputer dan hal-hal lain yang berhubungan dengan komputer (jaringan komputer, multimedia, telekomunikasi dan lain-lain).

2.7 Sistem

Secara sederhana sistem merupakan bentuk keterkaitan yang dilandasi dengan maksud mencapai persamaan tujuan secara bersama-sama (Sutarbi, 2016). Hal yang terikat dalam sistem adalah unsur, komponen atau variabel dengan kriteria berikut :

1. Terorganisir.
2. Saling berinteraksi.
3. Saling bergantung satu sama lain.
4. Terpadu.

Suatu sistem pada dasarnya adalah sekelompok unsur yang erat hubungannya satu dengan yang lain, yang berfungsi bersama-sama untuk mencapai tujuan tertentu.

Pratama (2016) mendefinisikan sistem sebagai sebuah kesatuan kompleks yang tersusun atas sejumlah komponen atau elemen yang saling terhubung. Kesatuan ini dimaksud untuk memudahkan penyelesaian satu atau beberapa proses. Sistem juga dapat disamakan dengan metode dan susunan yang teratur dari pandangan, asa, dan teori-teori yang ada di dalamnya.

2.8 Sistem Informasi

Sistem informasi merupakan sistem buatan manusia yang terdiri dari banyak komponen yang terorganisasi dengan tujuan yang berfokus pada penyajian informasi (Suryantara, 2017).

Sementara menurut Pratama (2014), sistem informasi merupakan sistem yang disusun oleh empat elemen berikut :

1. Perangkat lunak (*software*).
2. Perangkat keras (*hardware*).
3. Infrastruktur.
4. Sumber daya manusia (SDM) yang terlatih.

Keempat elemen diatas bekerjasama dalam mengolah data menjadi informasi yang bermanfaat, termasuk bekerjasama dalam menentukan proses perencanaan, kontrol, koordinasi dan pengambilan keputusan, akibatnya sistem yang mengolah data menjadi informasi merupakan sistem yang kompleks.

2.9 *Hyper Text Transfer Protocol (HTTP)*

Hyper Text Transfer Protocol (HTTP) merupakan sebuah protokol komunikasi antara *client* dan *server* dengan konsep *request-response*. Sebagai sebuah protokol, HTTP menentukan prosedur-prosedur komunikasi baik dalam format dan cara komunikasi hingga aksi dan reaksi antara *web server* dan *browser* (Hidayatullah dan Jauhari, 2015).

Menurut Pratama (2014), protokol yang paling umum digunakan oleh pengguna jaringan komputer adalah HTTP, khususnya dalam mengakses suatu situs (*website*). Dalam komunikasi antara *server* dan *client*, protokol HTTP sedikitnya menjalankan tiga buah fungsi :

1. Menentukan reaksi *web server (server)* terhadap aksi dari *web browser (client)*.
2. Membantu *web browser* menyajikan data dan informasi yang dikirimkan oleh *web server* berdasarkan permintaan dari *client*.
3. Membantu penerjemahan pesan dan perintah yang berasal dari *client* dan ditujukan ke *server*, serta tanggapan yang dikirimkan dari *server* ke *client* (berdasarkan permintaan dari *client*).

2.10 Web

Web merupakan sebuah ruang informasi yang digunakan secara global dan sumber informasinya diakses berdasarkan protokol yang disepakati bersama. *Web* seringkali diartikan sebagai Internet secara keseluruhan, namun faktanya *Web* merupakan bagian dari Internet saja (Darma dkk, 2009).

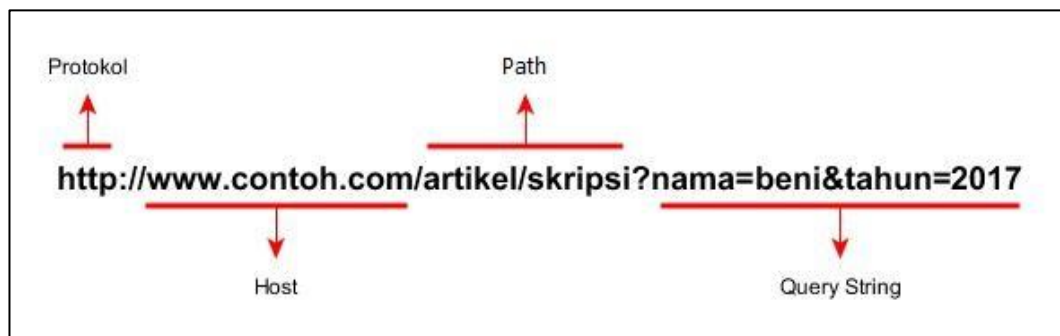
2.11 Web Server

Web server merupakan program komputer yang ditanamkan pada komputer yang diakses publik (*server*) sehingga komputer tersebut mampu menerima dan memberikan respon pada pengaksesnya (*client*). Aksi dan reaksi antara *web server* dan *client* ini diatur oleh protokol yang telah disepakati bersama, dan umumnya protokol yang digunakan adalah HTTP (MADCOMS, 2016).

2.12 Uniform Resource Locator (URL)

Uniform Resource Locator (URL) merupakan serangkaian kode berupa *string* bersifat unik sebagai identifikasi untuk menentukan alamat sumber informasi layanan *web* (Pratama, 2014). Suatu URL memuat empat buah komponen di dalam penulisannya, yaitu :

1. Protokol yang digunakan.
2. *Host* (komputer) yang dituju.
3. *Port* yang digunakan.
4. *Path* yang tepat dari alamat tujuan.



Gambar 2.1 Elemen pada URL

2.13 PHP

PHP (*Hypertext Preprocessor*) merupakan bahasa *script* yang dapat ditanamkan atau disisipkan ke dalam halaman *web*. Banyak *web* dapat dibangun dengan PHP, baik program *web* dinamis, manajemen konten *web*, dan lain-lain. PHP sendiri merupakan bahasa pemrograman *server side* karena diproses pada komputer *server*. Berbeda dibandingkan dengan beberapa bahasa pemrograman yang diproses pada sisi *client* (*client-side*) seperti *Javascript* yang diproses pada *web*

browser (client). PHP bersifat *Open Source* yang artinya dapat digunakan secara bebas tanpa membayar. Kemudahan dan kepopuleran PHP pada saat ini sudah menjadi standar bagi *programmer web* di seluruh dunia (MADCOMS, 2016).

2.14 Database

Menurut Pratama (2016), *Database* merupakan wadah bagi data dan informasi, dimana wadah tersebut memenuhi fungsi-fungsi berikut :

1. Penyimpanan data dan informasi (data yang telah diolah).
2. Pemrosesan untuk manipulasi (pengelolaan) data menjadi informasi.
3. Pengaksesan untuk manipulasi (pengelolaan) data menjadi informasi.
4. Pengaksesan data dan informasi.

2.15 MySQL

MySQL merupakan sistem manajemen *database SQL* yang bersifat *Open Source* dan paling populer saat ini. Sistem *Database MySQL* mendukung beberapa fitur seperti *multithreaded*, *multi-user* dan *SQL database management system (DBMS)*. *Database* ini dibuat untuk keperluan sistem *database* yang cepat, handal dan mudah digunakan. Ulf Micheal Widenius merupakan penemu awal versi pertama MySQL yang kemudian pengembangan selanjutnya dilakukan oleh perusahaan MySQL AB. MySQL AB yang merupakan sebuah perusahaan komersial yang didirikan oleh para pengembang MySQL (MADCOMS, 2016).

2.16 Application Programming Interface (API)

Secara umum API merupakan ekspresi terfokus keseluruhan fungsional dalam suatu modul *software* yang dapat diakses oleh orang yang membutuhkan dengan cara yang telah ditentukan layanan. Representasi terfokus dari fungsi yang dideklarasikan dalam API dimaksudkan untuk menyediakan rangkaian layanan yang spesifik untuk target tertentu. Jika dalam satu modul memiliki API ganda, hal ini sudah menjadi hal yang umum karena setiap API dimaksudkan untuk penggunaan yang spesifik dari modul terkait (Rama dan Avinash, 2015).

2.17 Web API

Web API adalah antar muka program dari sistem yang dapat diakses melalui *method* dan *header* pada protokol HTTP yang standar. *Web API* dapat diakses dari berbagai macam HTTP *client* seperti *browser* dan perangkat *mobile*. *Web API* juga memiliki keuntungan karena menggunakan infrastruktur yang juga digunakan oleh *web* terutama untuk penggunaan *caching* dan *concurrency* (Miller dkk, 2014).

2.18 Javascript Object Notation (JSON)

Javascript Object Notation (JSON) adalah sebuah *general-purpose data encoding* format yang populer. Penerapan JSON telah banyak digunakan pada *database* dan *web service*. Struktur dokumen JSON secara opsional dapat dibatasi berdasarkan skema yang terdiri atas dua hal yakni *map* (pemetaan struktur nilai berdasarkan klasifikasi jenisnya) dan *list* (pengelompokan nilai berdasarkan klasifikasi jenisnya) (Kleppmann dan Alastair, 2017).

2.19 Representational State Transfer (REST)

REST merupakan filosofi desain yang mendorong kita untuk menggunakan protokol dan fitur yang sudah ada pada *web* untuk memetakan permintaan terhadap sumber daya pada berbagai macam representasi dan manipulasi data di Internet (Scribner dan Seely, 2009).

REST adalah gaya arsitektural yang memiliki aturan seperti antar muka yang seragam, sehingga jika aturan tersebut diterapkan pada *web services* akan dapat memaksimalkan kinerja *web services* terutama pada performa, skalabilitas, dan kemudahan untuk dimodifikasi. Pada arsitektur REST, data dan fungsi dianggap sebagai sumber daya yang dapat diakses lewat *Uniform Resource Identifier* (URI), biasanya berupa tautan pada *web*.

REST menggunakan protokol HTTP yang bersifat *stateless*. Perintah HTTP yang bisa digunakan adalah fungsi *GET*, *POST*, *PUT* atau *DELETE*. Hasil yang dikirimkan dari *server* biasanya dalam bentuk format XML atau JSON sederhana tanpa ada protokol pemaketan data, sehingga informasi yang diterima lebih mudah dibaca dan *diparsing* pada sisi *client*.

Dalam penerapannya, REST lebih banyak digunakan untuk *web service* yang berorientasi pada sumber daya. Maksud orientasi pada sumber daya adalah orientasi yang menyediakan sumber daya sebagai layanannya dan bukan kumpulan-kumpulan dari aktifitas yang mengolah sumber daya itu. Bentuk *web service* menggunakan REST style sangat cocok digunakan sebagai *back-end* dari aplikasi berbasis *mobile* karena cara aksesnya yang mudah dan hasil data yang dikirimkan berformat JSON sehingga ukuran *file* menjadi lebih kecil.

Menurut Fielding (2000), REST adalah arsitektur standar *web* yang menggunakan protokol HTTP dalam komunikasi data. Arsitektur tersebut didirikan berdasarkan sumber data dimana masing-masing komponen merupakan sumber data. Sumber data diakses oleh antarmuka yang sama dengan menggunakan metode standar HTTP. Dalam arsitektur REST, *server* yang mengikuti arsitektur REST menyediakan akses ke sumber data dan klien yang mengambil data. Setiap sumber data diidentifikasi menggunakan link URI. REST menggunakan berbagai format untuk menyajikan data, seperti teks, JSON dan XML. Berikut adalah metode HTTP yang umumnya digunakan dalam arsitektur REST :

1. *GET* untuk menyediakan akses untuk membaca sumber data.
2. *PUT* untuk memperbarui data yang tersedia.
3. *DELETE* untuk menghapus data.
4. *POST* untuk membuat data baru.

Fielding juga mengemukakan lima sifat dasar REST antara lain :

1. *Uniform Interface* yang mengharuskan pemetaan *resource* dilakukan secara terstruktur dan memungkinkan setiap bagian dari *resource* sistem (hasil pemetaan) dapat berkembang secara swadaya.
2. *Stateless* yang memastikan setiap satu *request* akan menerima satu *response*. Diluar *request* tidak terjadi interaksi apapun antara *client* dan *server*, dan setelah komunikasi antara *client* dan *server* tidak tersisa nilai apapun yang dipertahankan pada *memory*.

3. *Cacheable* yang memungkinkan *server* untuk memberikan *response* yang sama berdasarkan *request* dari *client* tanpa melalui *back-end* pada pola *request* tertentu.
4. *Client-Server* yang memisahkan antara proses pada *back-end* dengan *interface* sistem dimana antara *client (interface)* dan *server (back-end system)* berfokus pada pengembangannya masing-masing (*logic* mampu berkembang secara terpisah).
5. *Layered System* yang melarang *request* dari *client* secara langsung dihubungkan kepada proses *back-end*. Hal ini ditujukan untuk meningkatkan skalabilitas dan keamanan sistem.

2.20 JSON Web Token (JWT)

JSON Web Token (JWT) merupakan representasi dari format data yang penggunaannya ditujukan untuk ditempatkan pada *header* Otorisasi HTTP atau parameter *query* URI. JWTs mengkodekan data yang dikirim sebagai objek JSON dengan bantuan *Signature* JSON Web (JWS) yang memungkinkan data ditandatangani atau dilindungi secara digital dengan *Message Authentication Code* (Jones dkk, 2015). JWT terdiri dari tiga struktur yang dipisahkan oleh tanda titik (.), yaitu:

1. *Header* untuk memuat jenis *encoding* yang digunakan.
2. *Payload* untuk memuat nilai-nilai informasi yang ditransaksikan.
3. *Signature* untuk memuat nilai *hash* untuk memverifikasi *payload*.

2.21 Postman

Postman merupakan sebuah *software* yang memuat fungsi lengkap pengembangan sistem dalam mengirimkan dan menerima respon *server*. *Software* ini mendukung pengembangan sistem REST API dengan mengklasifikasi *request* berdasarkan *request method*, URL dan parameter-parameter *request* (Postdot, 2017).

2.22 Equivalence Partitioning (EP)

Menurut Jovanovic (2009), *Equivalence Partitioning* (EP) merupakan salah satu teknik pengujian *input* pada sistem. Teknik ini mengklasifikasikan *input* menjadi kelas-kelas dengan batasan tertentu untuk kemudian diuji kevalidan hasilnya. Kelas-kelas *input* dapat ditentukan dengan cara berikut :

1. Minimal satu kelas valid dan dua kelas yang tidak valid untuk kondisi dimana sistem menerima *input* dalam rentang nilai spesifik.
2. Minimal satu kelas valid dan dua kelas yang tidak valid untuk kondisi dimana sistem menerima *input* hanya pada suatu nilai spesifik.
3. Minimal satu kelas valid dan satu kelas tidak valid untuk kondisi dimana sistem menerima *input* dalam serangkaian nilai tertentu.
4. Minimal satu kelas valid dan satu kelas tidak valid untuk kondisi dimana sistem menerima *input Boolean* (benar atau salah).

BAB III

METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

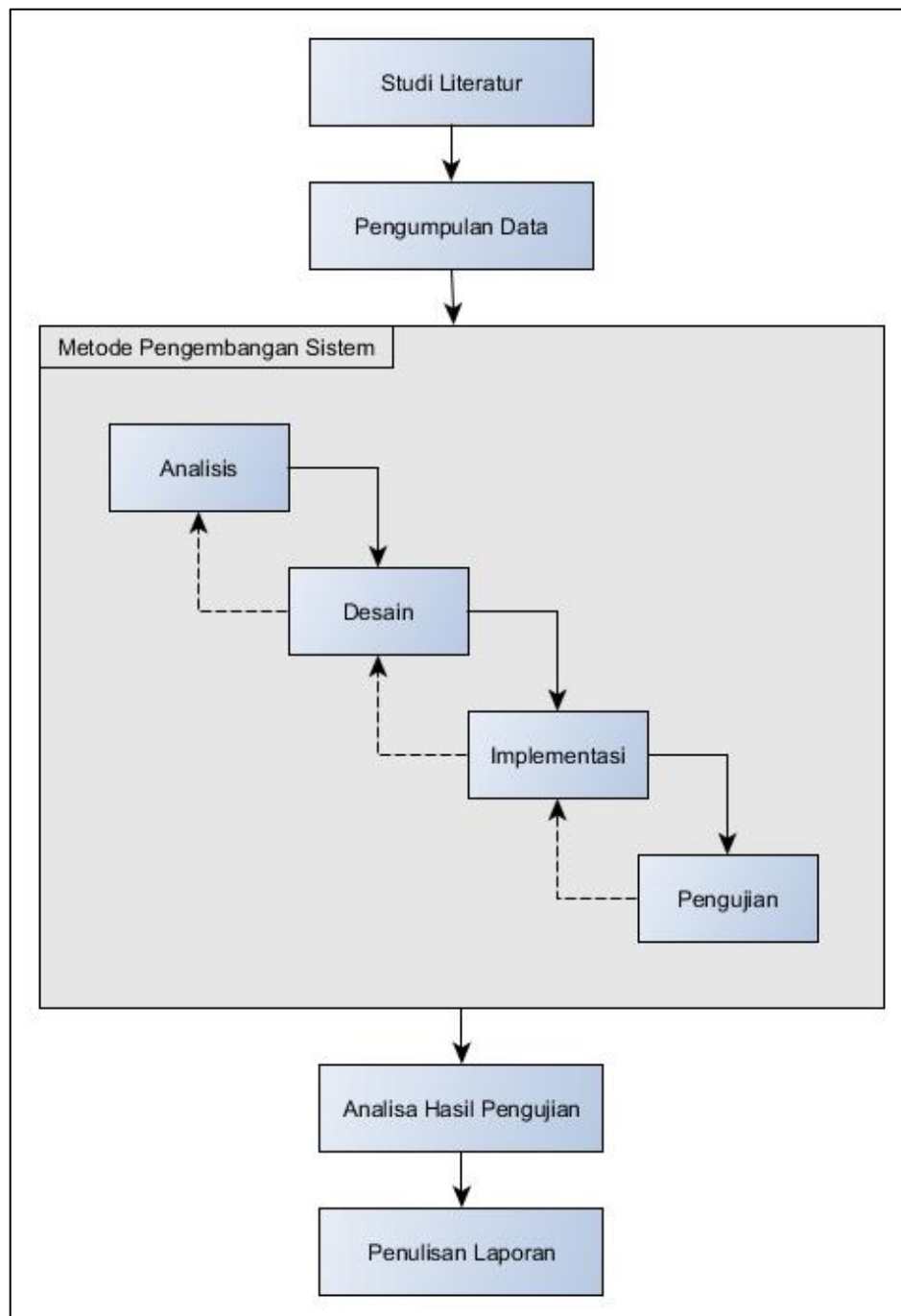
Penelitian ini dilakukan di kecamatan Pringsewu provinsi Lampung dan Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung. Waktu penelitian dilaksanakan dari bulan Januari 2017 sampai bulan Juni 2017.

3.2 Metode Penelitian

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).

Metode SDLC yang digunakan dalam penelitian ini adalah metode *Waterfall* dengan tahapan seperti yang dapat dilihat pada gambar 3.1. Metode ini 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*” (Pranata, 2014) dan “*Token-Based Authentication Using JSON Web Token on SIKASIR RESTful Web Service*” (Haekal, 2016).



Gambar 3.1 Tahap penelitian metode *Waterfall*

Implementasi metode *waterfall* pada penelitian pengembangan *back-end* sistem administrasi pasien klinik perawatan kulit kesehatan dilakukan dalam lima tahap, yakni studi literatur, analisis, desain, implementasi, dan pengujian.

3.2.1 Studi Literatur

Pada tahap ini peneliti mengumpulkan informasi dengan melakukan studi terhadap penelitian-penelitian yang mengangkat topik seputar pengembangan sistem informasi dengan konsep API dan gaya arsitektur REST. Peneliti juga melakukan komunikasi langsung dan berdiskusi dengan karyawan bagian administrasi pada salah satu penyelenggara pelayanan kesehatan. Hal ini dilakukan untuk menghindari kesalahan-kesalahan yang mungkin dilakukan dalam penelitian-penelitian sebelumnya serta memperkokoh konsep-konsep yang digunakan dalam penelitian dengan kasus serupa.

3.2.2 Pengumpulan Data

Pada tahap pengumpulan data, peneliti mengumpulkan beberapa dokumen pendukung yang pada umumnya digunakan dalam penyelenggaraan pelayanan kesehatan di klinik kulit dan kelamin. Dokumen-dokumen tersebut antara lain kartu kontrol pasien, bukti pembayaran dan contoh skema pencatatan konsumsi pasien terhadap barang dan jasa. Selanjutnya data yang telah dikumpulkan akan menjadi bahan pertimbangan dalam tahap analisis.

3.2.3 Analisis

Pada tahap analisis, peneliti melakukan analisa kebutuhan yang menjadi dasar persiapan dalam pengembangan sistem. Hal ini dilakukan supaya pengembangan

sistem benar-benar sesuai dengan studi kasus pada pelayanan kesehatan, khususnya administrasi pasien klinik perawatan kulit. Analisis diklasifikasikan menjadi empat bagian, yakni analisis proses bisnis rekam medis, analisis diagram relasi entitas, analisis kandidat *database* dan analisis kandidat API.

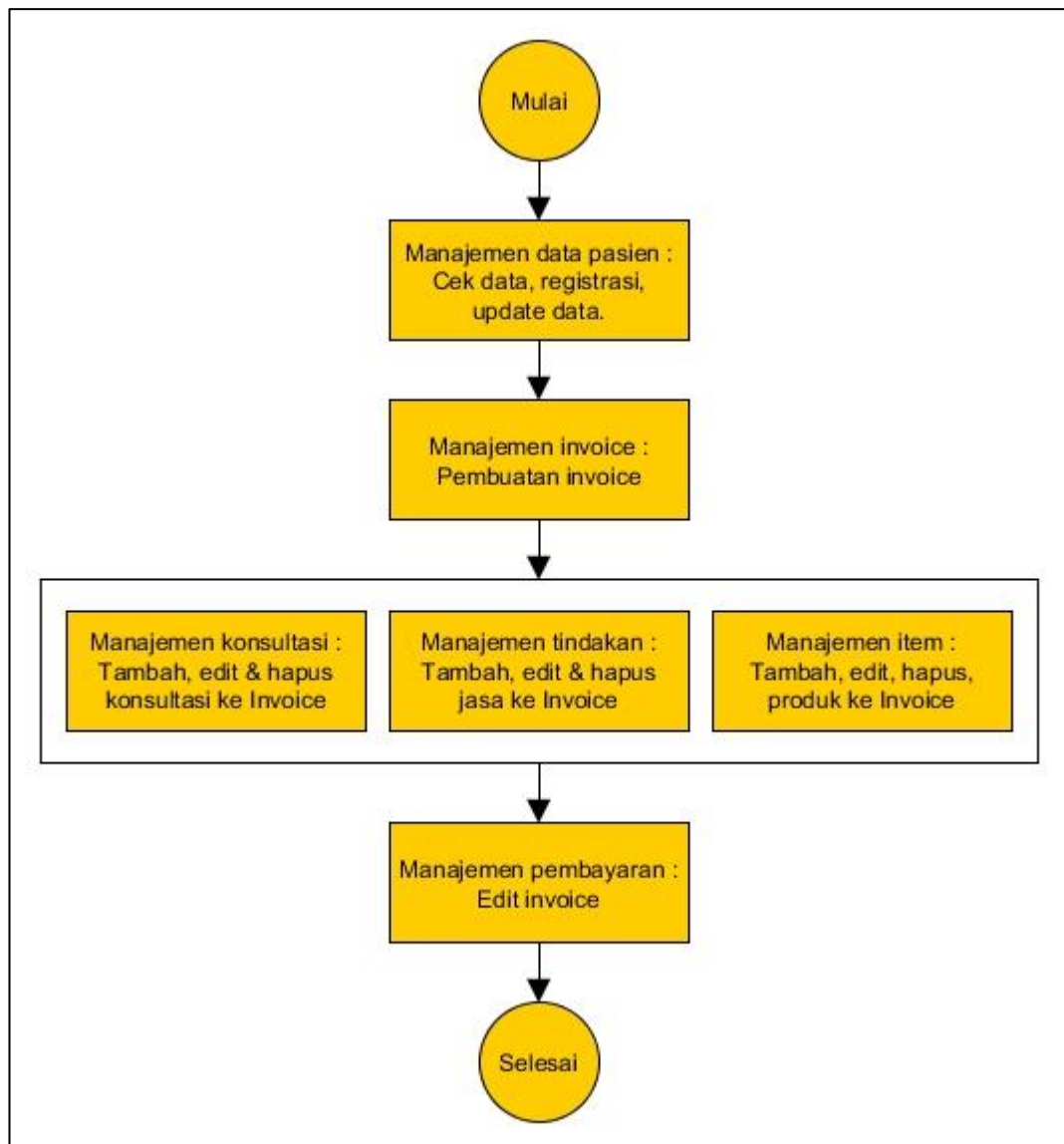
3.2.3.1 Analisis Proses Bisnis Rekam Medis

Pada tahap awal proses analisis proses bisnis rekam medis, peneliti mengumpulkan informasi dengan mewawancarai seorang sumber yang berprofesi sebagai *administrator* salah satu klinik kulit dan kelamin. Kemudian berdasarkan hasil wawancara dilakukan penentuan prosedur-prosedur yang berhubungan langsung dengan fungsi-fungsi dari API yang dikembangkan. Hasil analisis ini kemudian difokuskan kepada lima entitas utama dalam administrasi pasien klinik perawatan kulit, antara lain :

1. Pasien yang mengatur prosedur-prosedur dalam pembuatan, pembaruan (pengubahan) dan penghapusan data pasien.
2. *Invoice* yang mengatur prosedur-prosedur dalam pembuatan, pembaruan (pengubahan) dan penghapusan data *invoice* beserta produk-produk klinik perawatan kulit yang terdaftar dalam *invoice* itu sendiri.
3. Konsultasi yang mengatur prosedur-prosedur pelaksanaan jasa konsultasi oleh seorang dokter serta administrasinya dalam sebuah *invoice*.

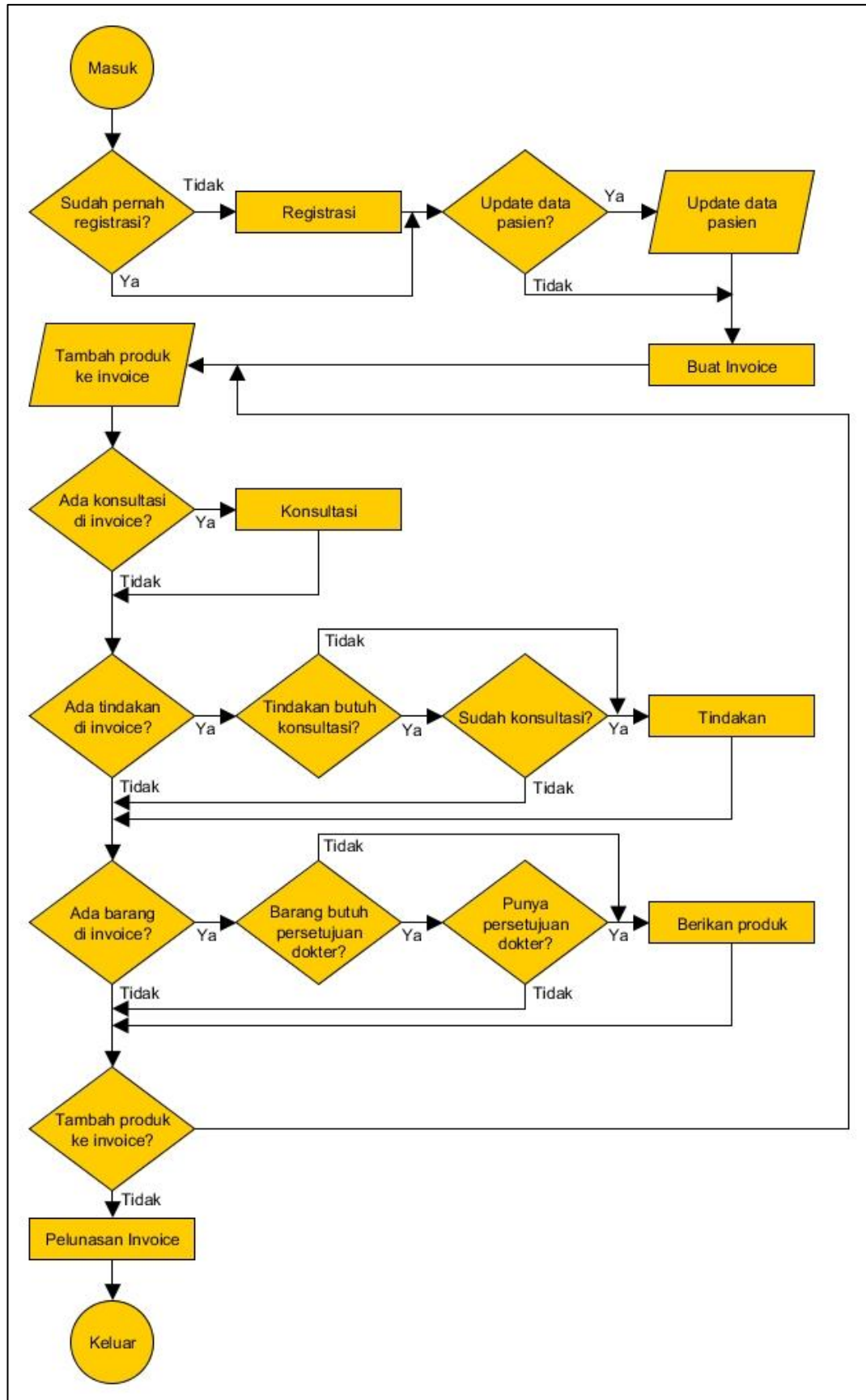
4. Tindakan yang mengatur prosedur-prosedur pelaksanaan jasa penanganan oleh satu orang atau lebih dokter atau perawat terhadap pasien beserta administrasinya dalam sebuah *invoice*.
5. Produk yang mengatur prosedur-prosedur penyerahan produk-produk klinik perawatan kulit beserta administrasinya dalam sebuah *invoice*.

Kelima pokok bahasan diatas saling berinteraksi dalam sebuah alur proses bisnis seperti yang terlihat pada gambar 3.2.



Gambar 3.2 Alur umum proses bisnis klinik perawatan kulit dan kelamin

Kemudian, analisis alur umum proses bisnis dikembangkan kedalam bentuk *flowchart* seperti yang dapat dilihat pada gambar 3.3.

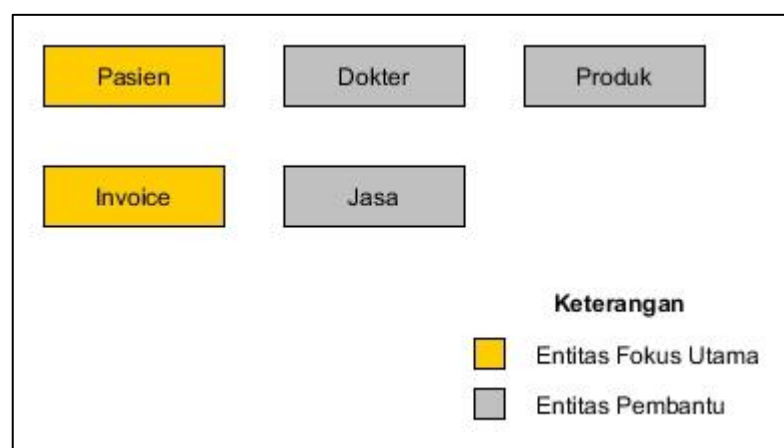


Gambar 3.3 *Flowchart* proses bisnis administrasi rekam medis

3.2.3.2 Analisis Diagram Relasi Entitas

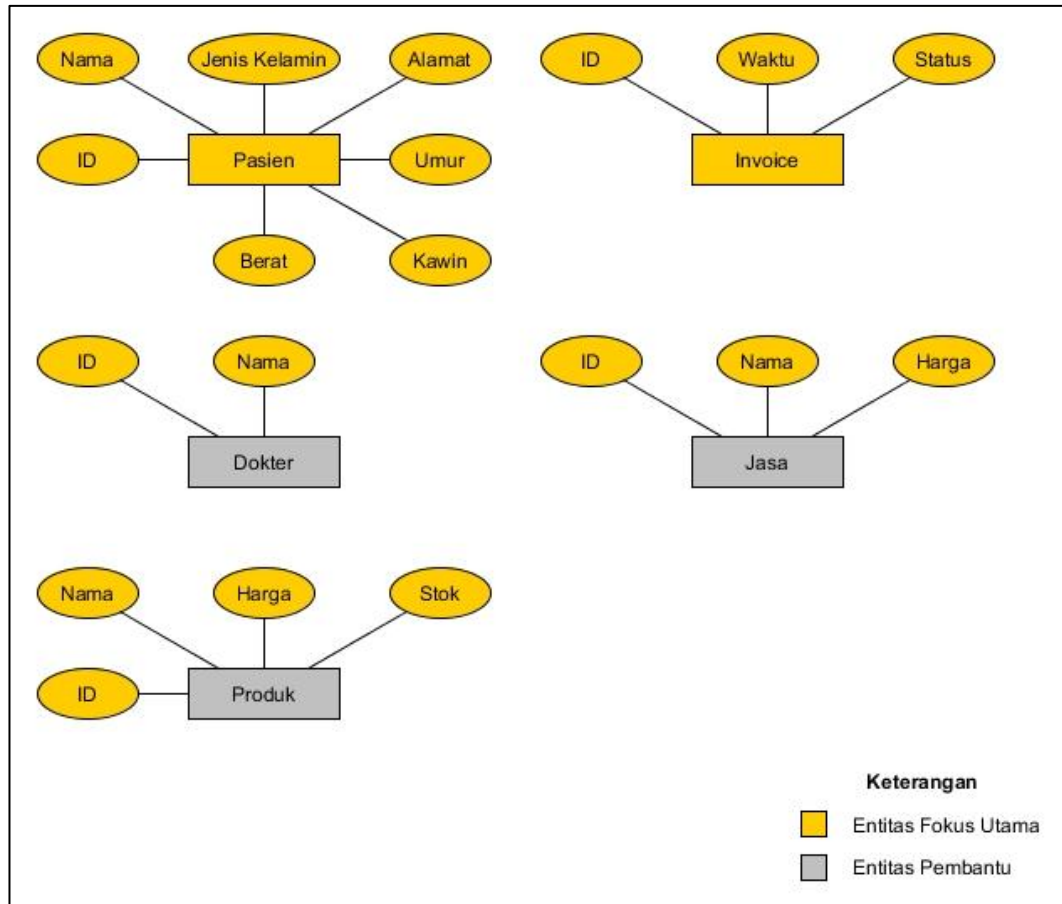
Analisis diagram relasi entitas dilakukan untuk menentukan hubungan antar data dalam basis data berdasarkan objek-objek dasar data pada penelitian. Analisis ini dilakukan dalam empat tahap, yakni penentuan entitas, penentuan atribut, penentuan relasi dan penentuan diagram relasi entitas itu sendiri. Analisis ini bertujuan untuk mempermudah peneliti dalam menentukan basis data dan perencanaan pengembangan API.

Tahap pertama dalam analisis diagram relasi entitas merupakan penentuan objek-objek yang akan menjadi basis utama dalam diagram relasi entitas, objek-objek tersebut dapat dilihat pada gambar 3.4. Pada tahap ini didapatkan dua entitas utama yang menjadi fokus penelitian, yakni pasien dan *invoice*. Sementara didapatkan dua entitas yang akan mendukung penelitian ini, yakni produk dan karyawan.



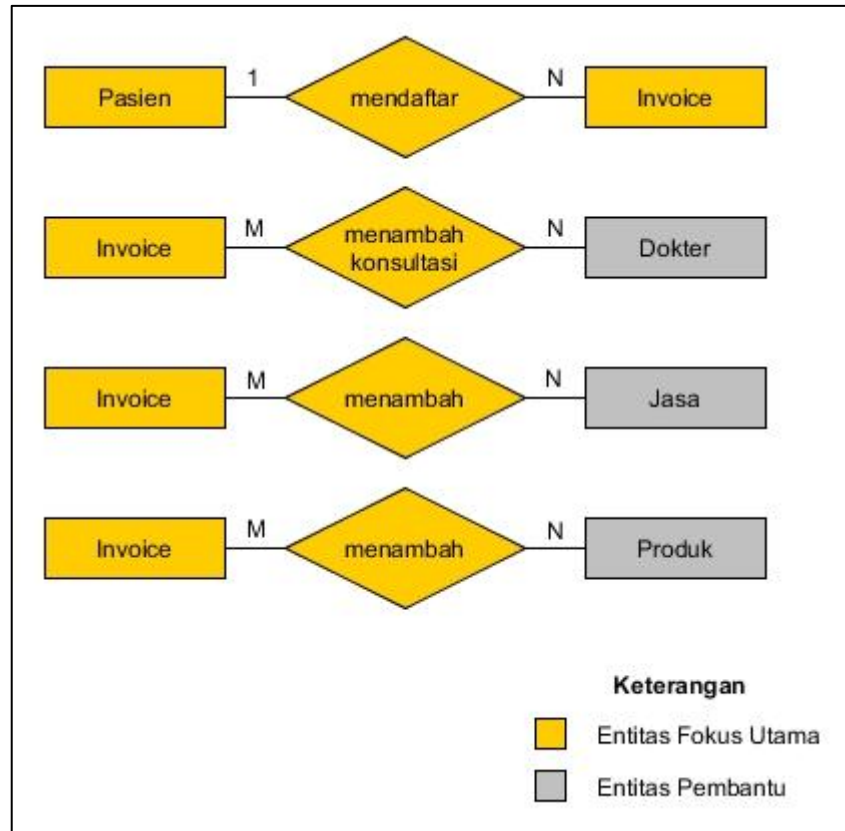
Gambar 3.4 Entitas

Tahap kedua dalam analisis diagram relasi entitas adalah menentukan atribut-atribut pada basis entitas. Hasil dari tahap kedua dapat dilihat pada gambar 3.5.



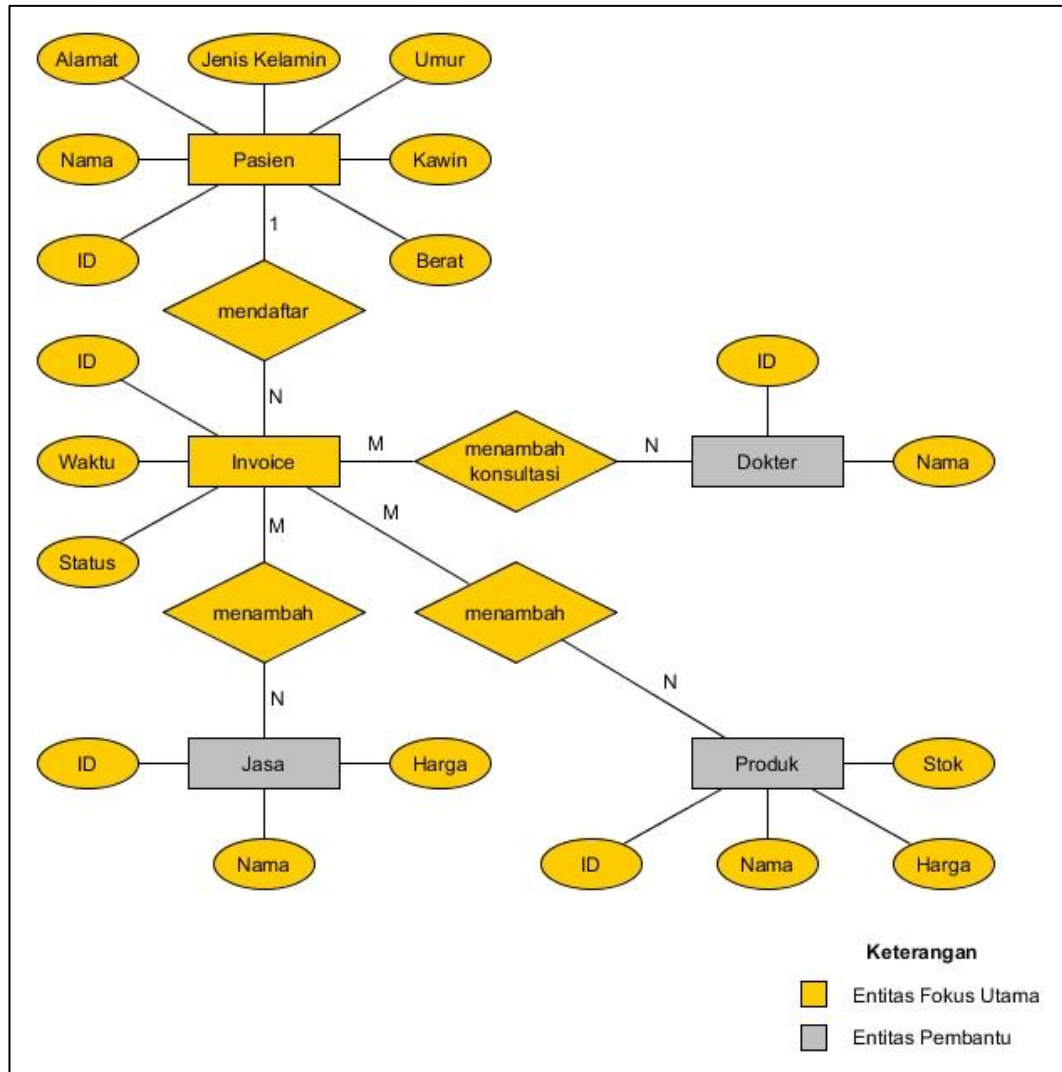
Gambar 3.5 Atribut entitas

Kemudian pada tahap ketiga analisis diagram relasi dilakukan penentuan kardinalitas entitas. Tahap ini menganalisa kemungkinan-kemungkinan relasi serta himpunan antara sebuah entitas terhadap entitas lainnya, dan hasilnya dapat dilihat pada gambar 3.6.



Gambar 3.6 Kardinalitas entitas

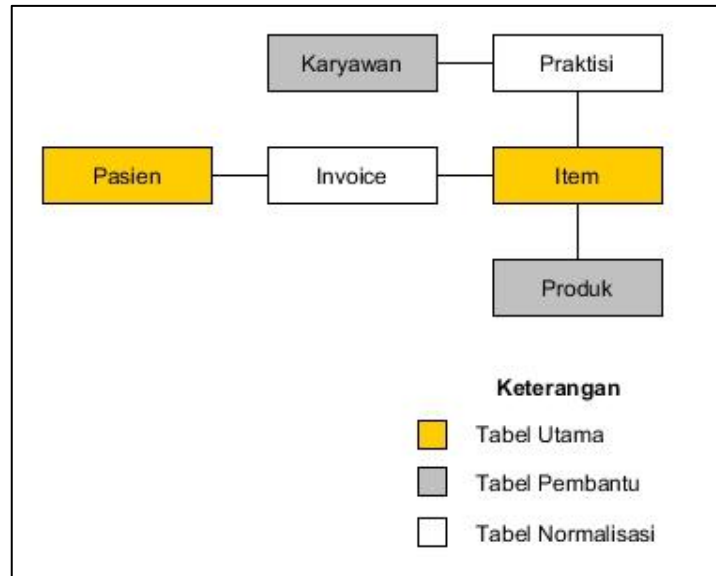
Pada tahap akhir analisis diagram relasi entitas, peneliti menghubungkan entitas-entitas berdasarkan seluruh tahap sebelumnya. Hasil dari tahap ini dapat dilihat pada gambar 3.7.



Gambar 3.7 Diagram relasi entitas

3.2.2.3 Analisis Kandidat Database

Analisis kandidat *database* dilakukan berdasarkan diagram relasi entitas yang telah ditentukan. Pada tahap ini peneliti juga melakukan normalisasi terhadap entitas yang memiliki relasi himpunan *many-to-many*. Hasil dari analisis ini dapat dilihat pada gambar 3.8.

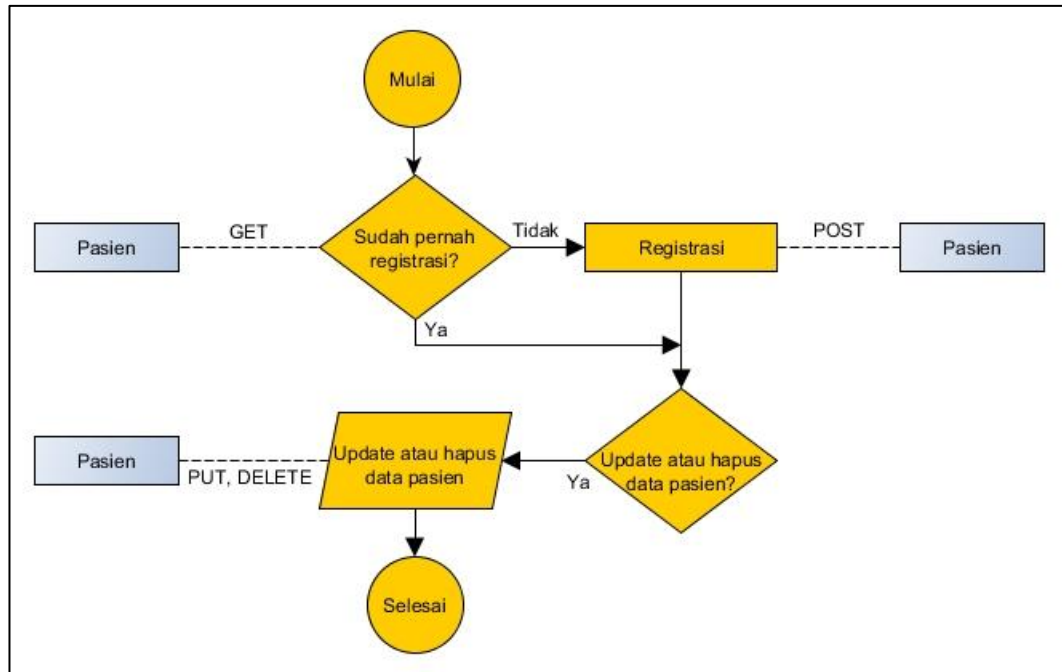


Gambar 3.8 Analisis kandidat *database*

3.2.2.4 Analisis Kandidat API

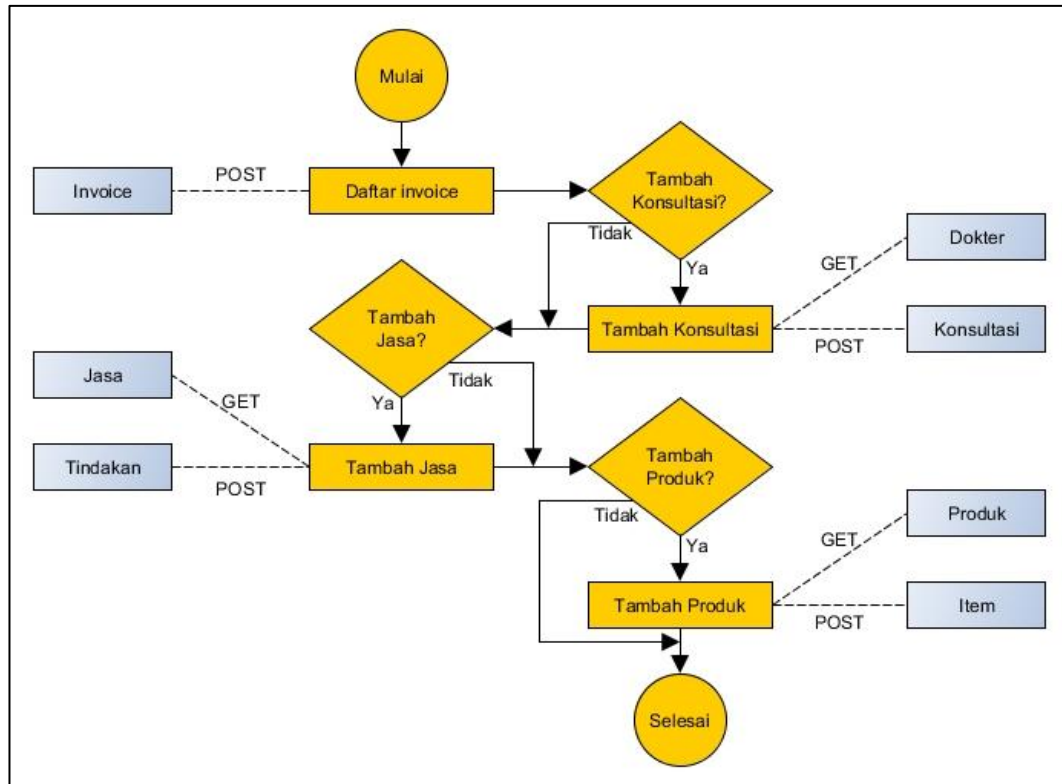
Pada analisis kandidat API telah ditentukan fungsi-fungsi yang akan menjadi basis penerapan basis API dengan metode REST berdasarkan hasil analisis proses bisnis, analisis diagram relasi entitas dan analisis kandidat *database*. Analisis ini bertujuan untuk memfokuskan pengembangan sistem supaya tidak keluar dari lingkup pengembangan API yang telah direncanakan.

Untuk mempermudah analisis kandidat API, peneliti membagi analisis menjadi empat tahap. Pada tahap pertama analisis kandidat API, telah ditentukan kandidat API berdasarkan hal-hal yang berhubungan dengan basis data pasien. Hal ini mencakup seluruh kemungkinan interaksi pasien dengan *administrator* pada saat pertama kali pasien datang ke klinik dalam kondisinya nyata yang dipetakan berdasarkan alur informasinya. Hasil dari analisis ini dapat dilihat pada gambar 3.9.



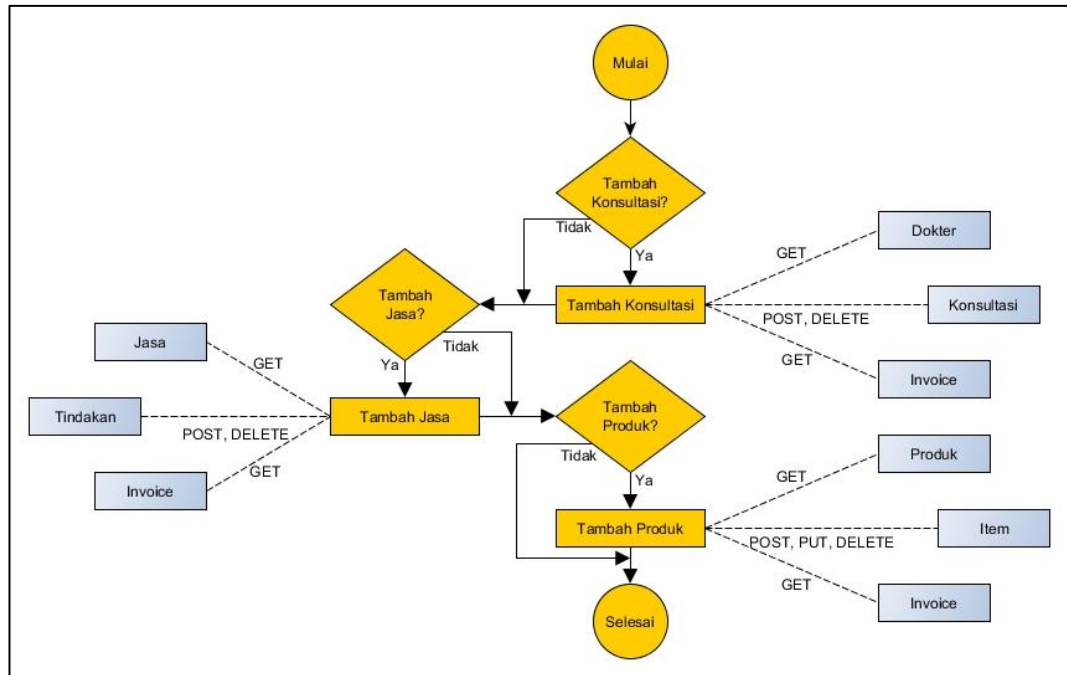
Gambar 3.9 Analisis kandidat API tahap pertama

Pada tahap kedua analisis kandidat API, telah ditentukan kandidat API berdasarkan hal-hal yang berhubungan dengan *invoice* serta produk dan jasa. Hal ini mencakup seluruh kemungkinan interaksi pasien dengan *administrator* pada saat pencatatan awal, penambahan dan perubahan produk dan jasa untuk pasien sebelum konsultasi kondisinya nyata yang dipetakan berdasarkan alur informasinya. Hasil dari analisis ini dapat dilihat pada gambar 3.10.



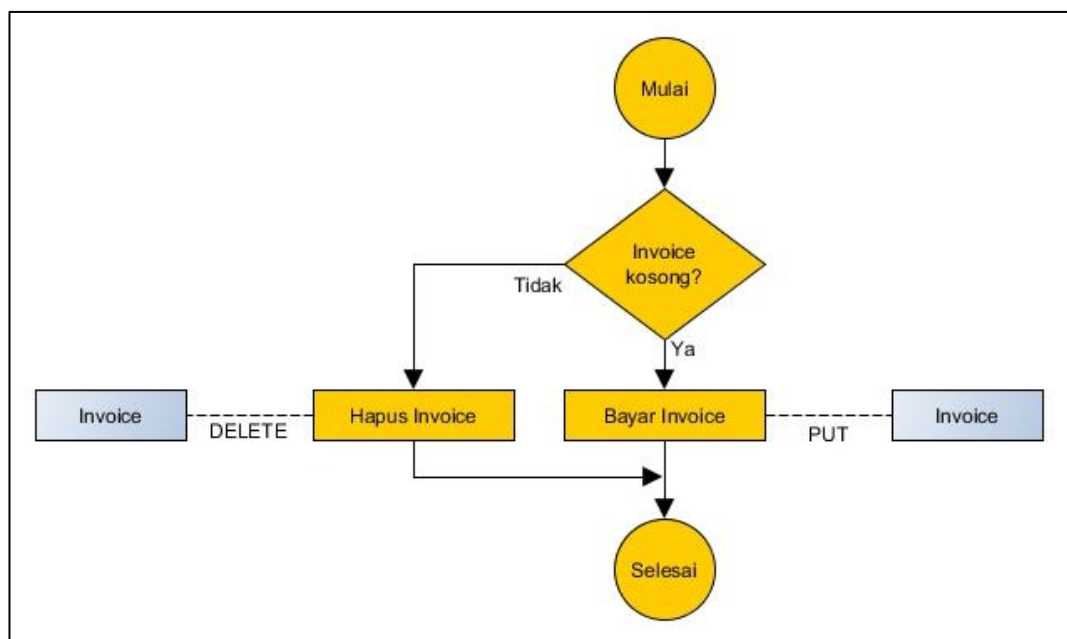
Gambar 3.10 Analisis kandidat API tahap kedua

Pada tahap ketiga analisis kandidat API, telah ditentukan kandidat API berdasarkan hal-hal yang berhubungan perubahan data pada *invoice*. Hal ini mencakup seluruh kemungkinan interaksi pasien dengan *administrator* tentang perubahan data pada *invoice* setelah pasien berkonsultasi dengan dokter. Hasil dari analisis tahap ketiga dapat dilihat pada gambar 3.11.



Gambar 3.11 Analisis kandidat API tahap ketiga

Kemudian pada tahap keempat analisis kandidat API, telah ditentukan kandidat API berdasarkan hal-hal yang berhubungan dengan pembatalan dan pelunasan *invoice*. Hasil dari tahap keempat dapat dilihat pada gambar 3.12.



Gambar 3.12 Analisis kandidat API tahap keempat

Berdasarkan analisis kandidat API, dapat disimpulkan keseluruhan kandidat API yang menjadi fokus pengembangan sistem *back-end* pada penelitian ini kedalam tabel 3.1.

Tabel 3.1 Kandidat API

No	Proses	Kandidat API		Entitas
		Method	URI	
1	Menampilkan daftar pasien	GET	/pasien	Pasien
2	Menampilkan data pasien	GET	/pasien/<pasien_id>	Pasien
3	Mendaftarkan pasien	POST	/pasien	Pasien
4	Mengedit pasien	PUT	/pasien/<pasien_id>	Pasien
5	Menghapus pasien	DELETE	/pasien/<pasien_id>	Pasien
6	Membuat invoice	POST	/pasien/<pasien_id>/invoice	Invoice
7	Menampilkan daftar dokter	GET	/dokter	Dokter
8	Menambahkan konsultasi ke invoice	POST	/invoice/<invoice_id>/konsultasi	Konsultasi
9	Menampilkan daftar jasa	GET	/jasa	Jasa
10	Menambahkan jasa ke invoice	POST	/invoice/<invoice_id>/tindakan	Tindakan
11	Menampilkan daftar produk	GET	/produk	Produk
12	Menambahkan produk ke invoice	POST	/invoice/<invoice_id>/item	Item

No	Proses	Kandidat API		Entitas
		Method	URI	
13	Menampilkan data invoice	GET	/invoice/<invoice_id>	Invoice
14	Menghapus konsultasi dari invoice	DELETE	/konsultasi/<konsultasi_id>	Konsultasi
15	Menghapus jasa dari invoice	DELETE	/tindakan/<tindakan_id>	Tindakan
16	Mengedit produk pada invoice	PUT	/item/<item_id>	Item
17	Menghapus produk dari invoice	DELETE	/item/<item_id>	Item
18	Mengedit invoice	PUT	/invoice/<invoice_id>	Invoice
19	Menghapus invoice	DELETE	/invoice/<invoice_id>	Invoice

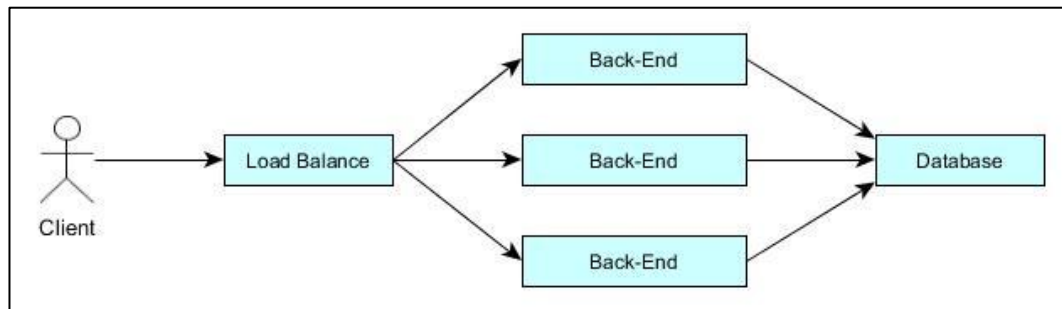
3.2.4 Desain

Pada penelitian ini, tahap desain secara umum dibagi menjadi empat bagian utama. Empat bagian tersebut adalah desain *server*, desain *database*, desain kode program dan desain *web server*.

3.2.4.1 Desain Server

Sistem pada penelitian ini direncanakan mampu diimplementasi pada *back-end server* berjumlah lebih dari satu dengan harapan *traffic* yang masuk dapat bergantian ditangani oleh *server* yang berbeda. Untuk mencapai hal itu, jenis otentikasi yang digunakan pada sistem tidak boleh terikat pada satu *server* saja melainkan harus dapat diterima oleh seluruh *back-end server*. Lingkungan yang

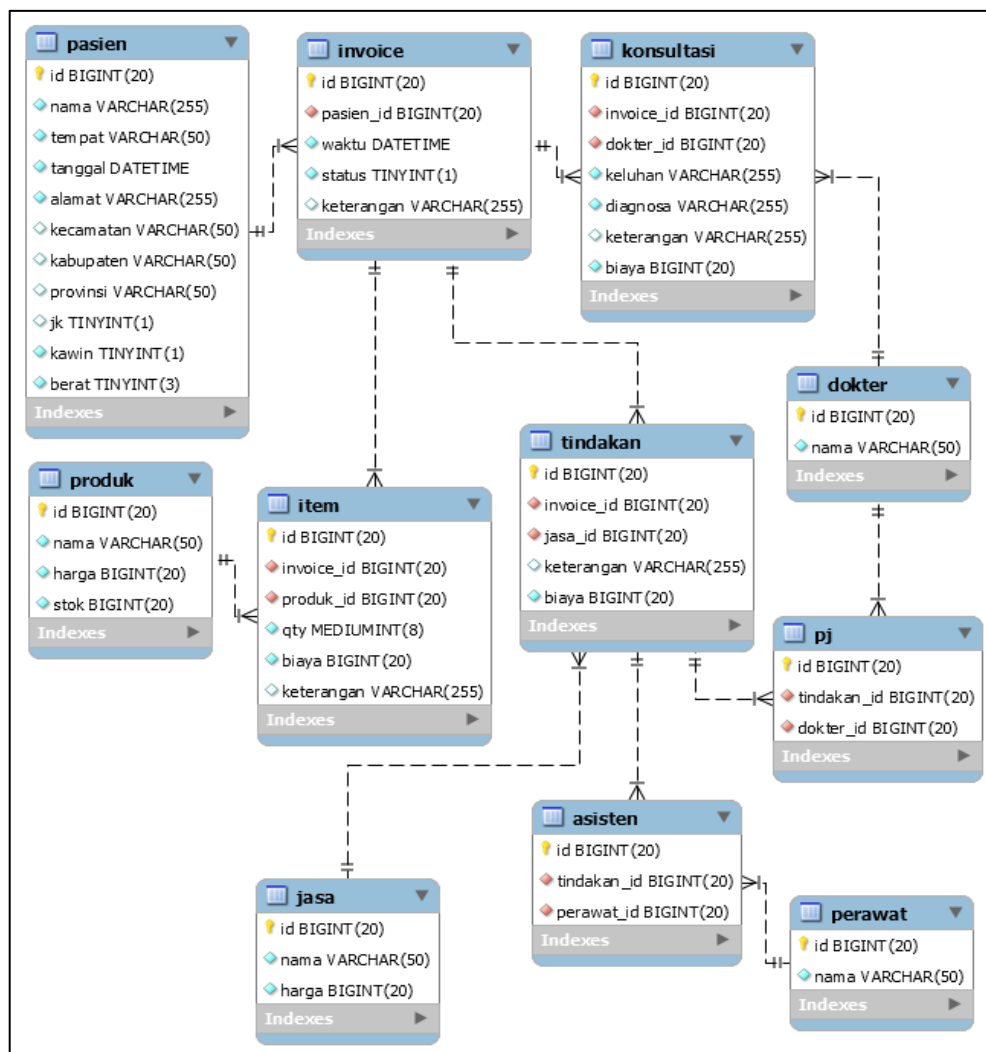
digunakan dalam penelitian ini adalah satu *host* sebagai *load balance server*, tiga *host* sebagai *back-end server* dan satu *database server*. Lingkungan yang dibangun ditujukan untuk proses pengujian dan bukan implementasi sebenarnya.



Gambar 3.13 Desain *server*

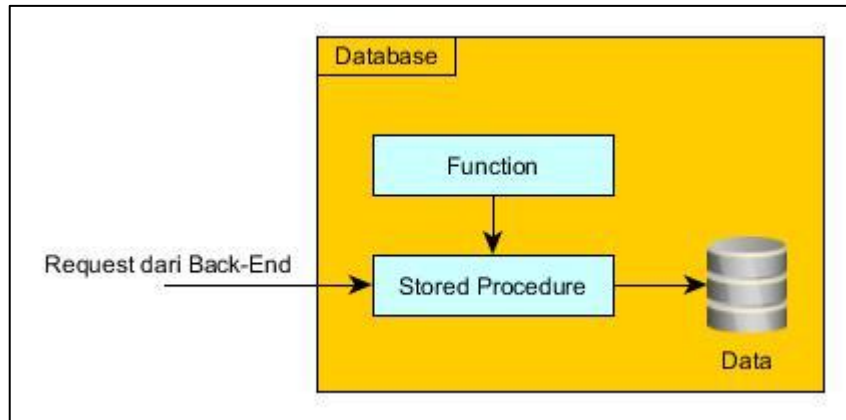
3.2.4.2 Desain *Database*

Pada penelitian ini dirancang sebuah desain tabel *database* yang diharapkan mampu menampung seluruh proses bisnis yang akan difasilitasi. Desain *database* setidaknya mencakup entitas-entitas utama dalam klinik perawatan kulit seperti pasien, *invoice*, dokter, produk dan jasa yang beberapa diantaranya membutuhkan normalisasi dalam relasi tabelnya.



Gambar 3.14 Struktur tabel *database*

Pada penelitian ini juga dirancang *stored procedure* yang merepresentasikan hal-hal yang dapat dilakukan pada *database*. *Stored procedure* digunakan untuk membatasi hak-hak manipulasi *database*. Hal ini akan berguna untuk meningkatkan keamanan data pada *database* sekaligus memberikan gambaran mengenai logika-logika pemrograman yang harus dibuat dalam memanipulasi data.



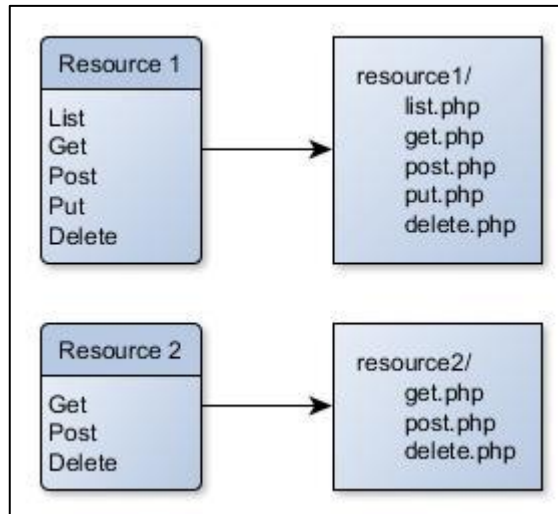
Gambar 3.15 Alur *request database*

3.2.4.3 Desain Kode Program

Peneliti menggunakan PHP sebagai bahasa pemrograman untuk membangun *back-end*. Penulisan kode program dilakukan dengan konsep pemrograman berorientasi *object*. Jenis operasi yang tersedia pada setiap *resource* adalah :

1. *List* untuk merequest daftar *resource* yang tersedia.
2. *Post* untuk menambah *resource*.
3. *Get* untuk meminta data lengkap dari satu *resource*.
4. *Put* untuk mengubah detail *resource*.
5. *Delete* untuk menghapus *resource* yang tersedia.

Dalam penulisan kode program diberlakukan aturan satu file kode untuk setiap jenis operasi pada setiap *resource*. Hal ini dilakukan untuk mempermudah pengembangan *logic* pada *resource* secara swadaya antara yang satu dengan lainnya.



Gambar 3.16 Struktur folder kode program

3.2.4.4 Desain Web Server

Peneliti menggunakan modul *rewrite* pada *software* Nginx sebagai *web server* untuk membantu proses pemetaan struktur REST kepada kode program. *Request* yang masuk pada *web server* dialihkan kepada kode program yang sesuai.

3.2.5 Implementasi

Tahap implementasi pada penelitian ini dilakukan dalam beberapa sub proses, antara lain :

1. Membangun lingkungan pengembangan sistem.
2. Mendesain struktur *table*, *function* dan *stored procedure* pada *database*.
3. Mengembangkan sistem *back-end* (*coding*) dan menyesuaikan dengan *database*.
4. Mendesain struktur *rewrite* pada *web server*.

3.2.6 Pengujian

Untuk memastikan bahwa sistem berjalan sesuai dengan rencana pengembangan sistem dan proses bisnis yang difasilitasi, diperlukan sebuah skenario pengujian sistem. Pada penelitian ini, pengujian akan dilakukan dengan menggunakan *software Postman* dan dibagi menjadi tiga skenario pengujian, antara lain :

1. Pengujian otentikasi *token* untuk memastikan prinsip REST terpenuhi.

Pengujian ini akan dilakukan pada *load-balanced server* dengan jumlah *back-end server* sebanyak tiga *virtual server* yang diklasifikasi berdasarkan *port* jaringan.

2. Pengujian dengan metode *equivalent partitioning* untuk memastikan

nilai-nilai masukan sesuai dengan rencana pengembangan sistem.

Pengujian ini akan mengelompokkan nilai masukan ke dalam kelas-kelas batasan nilai untuk kemudian diuji dalam kasus-kasus tertentu.

Daftar pengujian pada skenario ini dapat dilihat pada tabel 3.2.

Tabel 3.2 Rencana pengujian *equivalence partitioning*

No	Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan
1	Path dan request method	Validasi path	Request method GET untuk /pasien	Sukses mendapatkan list pasien
			Request method GET untuk /pasien/1	Sukses mendapatkan detail pasien Id 1
			Request method GET untuk /pasien/1/invoice	Sukses mendapatkan list invoice untuk

No	Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan
				pasien Id 1
			Request method GET untuk /pasien/1/invoice/1	Error dengan kode 404 - not found
			Request method GET untuk /salah	Error dengan kode 404 - not found
		Validasi request method	Request method POST untuk /pasien	Sukses menginput record pasien baru
			Request method GET untuk /pasien	Sukses mendapatkan list pasien
			Request method GET untuk /pasien/1	Sukses mendapatkan detail pasien Id 1
			Request method PUT untuk /pasien/1	Sukses mengedit data pasien dengan Id 1
			Request method DELETE untuk /pasien/1	Sukses menghapus data pasien dengan Id 1
			Request method PUT untuk /pasien	Sukses mengedit data pasien dengan Id 1
			Request method DELETE untuk /pasien	Error dengan kode 404 - not found
			Request method SALAH untuk /pasien	Error dengan kode 404 - not found
2	Otentik	Validasi	Merequest GET /pasien dengan	Sukses mendapatkan

No	Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan
	asi	token	menggunakan session sebelum 5 menit	list pasien
			Merequest GET /pasien dengan menggunakan session setelah 5 menit	Gagal mendapatkan list pasien dengan pesan error "Access token not valid."
			Merequest GET /pasien menggunakan token permanen dari alamat Ip yang sesuai sebelum 5 menit	Sukses mendapatkan list pasien
			Merequest GET /pasien menggunakan token permanen dari alamat Ip yang sesuai setelah 5 menit	Sukses mendapatkan list pasien
			Merequest GET /pasien menggunakan token permanen dari alamat Ip yang tidak sesuai	Gagal mendapatkan list pasien dengan pesan error "Access token not valid."
			Merequest GET /pasien tanpa token	Gagal mendapatkan list pasien dengan pesan error "Access token not valid."
			Merequest GET / pasien dengan token yang salah	Gagal mendapatkan list pasien dengan pesan error "Access

No	Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan
				token not valid."
3	Parameter	Kesesuaian tipe data	Merequest POST /pasien dengan parameter nama "qwerty"	Sukses menginput record pasien baru
			Merequest POST /pasien dengan parameter nama "12345"	Sukses menginput record pasien baru
			Merequest POST /pasien dengan parameter berat 50	Sukses menginput record pasien baru
			Merequest POST /pasien dengan parameter berat "ab"	Gagal menginput record pasien baru dengan pesan error "Parameter berat is not set properly."
			Merequest POST /pasien dengan parameter tanggal 19920711	Sukses menginput record pasien baru
			Merequest POST /pasien dengan parameter tanggal "199abc11"	Gagal menginput record pasien baru dengan pesan error "Parameter tanggal is not set properly."
		Validasi nilai	Merequest POST /pasien dengan parameter nama "ab" (2 karakter)	Gagal menginput record pasien baru dengan pesan error "Size of parameter nama does not meet the minimum value."

No	Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan
			tuvwxyzabcdefghijklmnopqrstuv wxyzabcdefghijklmnopqrstuv" (256 karakter)	
			Merequest POST /pasien dengan parameter berat -50	Sukses menginput record pasien baru dengan berat disesuaikan pada nilai minimal parameter yakni 0
			Merequest POST /pasien dengan parameter berat 0	Sukses menginput record pasien baru
			Merequest POST /pasien dengan parameter berat 50	Sukses menginput record pasien baru
			Merequest POST /pasien dengan parameter berat 500	Sukses menginput record pasien baru dengan berat disesuaikan pada nilai maksimal parameter yakni 255
			Merequest POST /pasien dengan parameter tanggal 19920711	Sukses menginput record pasien baru
			Merequest POST /pasien dengan parameter tanggal 19910229 (29 Februari pada tahun bukan kabisat)	Gagal menginput record pasien baru dengan pesan error "Parameter tanggal is

No	Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan
				not set properly."
			Merequest POST /pasien dengan parameter tanggal 19920229 (29 Februari pada tahun kabisat)	Sukses menginput record pasien baru

3. Pengujian fungsional untuk memastikan setiap titik akses API memberikan respon yang sesuai dengan rencana pengembangan. Pengujian ini akan dilakukan dengan mencoba secara langsung setiap titik akses sistem API yang telah dibuat.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dapat disimpulkan beberapa hal sebagai berikut :

1. Prinsip REST pada API menggunakan *path*, *query* dan *request method* dalam mengklasifikasi *resource*.
2. Metode REST dapat diimplementasikan kedalam perancangan API sistem informasi. Dalam penelitian ini metode REST berhasil diimplementasikan pada sistem administrasi pasien klinik perawatan kulit.
3. Khusus dalam penelitian ini, basis data pengguna dan *token* merupakan kasus yang unik dimana kedua basis data tersebut membutuhkan pola akses yang berbeda dari sumber daya lainnya.
4. Pengujian sistem otentikasi pengguna dalam penelitian ini memperoleh keberhasilan pada lingkungan uji dengan jumlah *back-end server* satu atau lebih.

5. Pengujian masukan sistem dengan metode *equivalence partitioning* dalam penelitian ini memperoleh keberhasilan menyeluruh berdasarkan skenario pengujian yang direncanakan.
6. Pengujian fungsional sistem dalam penelitian ini memperoleh keberhasilan menyeluruh pada setiap titik akses API yang telah dibuat.

5.2 Saran

Adapun saran yang dapat peneliti berikan berdasarkan hasil penelitian ini antara lain :

1. Supaya mampu dimanfaatkan oleh awam, *back-end* dari sistem informasi berbasis REST API harus memiliki *front-end* sebagai antarmukanya. Untuk itu dibutuhkan pengembangan lebih lanjut terutama untuk antarmuka yang menjadi media interaksi antara pengguna dengan *back-end server*.
2. Dalam pengembangan REST API, struktur *path* dan *query string* dengan *request method* yang sama memiliki banyak kesamaan pola. Untuk mempermudah pengembangannya, akan sangat baik jika struktur kode program dibuat berdasarkan *class* dan fungsi.

DAFTAR PUSTAKA

- Adelman, Clifford. 2000. *A Parallel Post-secondary Universe: The Certification System in Information Technology*. Washington: US Department of Education.
- Darma. dkk. 2009. *Buku Pintar Menguasai Internet*. Jakarta: Media Kita.
- Fielding, Roy Thomas. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. University Of California, Irvine.
- Haryadi, Hendi. 2009. *Administrasi Perkantoran untuk Manajer dan Staf*. Jakarta: Transmedia Pustaka.
- Hidayatullah, Priyanto dan Jauhari Khairul Kawistara. 2015. *Pemrograman Web*. Bandung: Informatika Bandung.
- Jones, Michael B. dkk. 2015. *JSON Web Token (JWT)*. Internet Engineering Task Force (IETF), Mei 2015.
- Jovanoic, Irena. 2009. *Software Testing Methods and Techniques*. The IPSI BgD Transactions on Internet Research.
- Kleppmann, Martin dan Alastair R. Beresford. 2017. *A Conflict-Free Replicated JSON Datatype*. University of Cambridge Computer Laboratory, Cambridge.
- MADCOMS. 2016. *Pemrograman PHP dan MySQL untuk Pemula*. Yogyakarta: Penerbit Andi.

- Miller, Darrel. dkk. 2014. *Designing Evolvable Web APIs with ASP.NET*. O'Reilly Media.
- Riani, Istiasih Fajar. 2017. *Wawancara Administrasi Pelayanan Kesehatan*. Lampung.
- Postdot Technology. 2017. *Postman is the most complete API Development Environment*. [Online]. San Francisco. Tersedia: <https://www.getpostman.com/postman> [13 September 2017].
- Pranata, Beni Adi. 2014. *Sistem Perangkum Laporan Jaminan Pelayanan Kesehatan Medical Center PT. Central Pertiwi Bahari*. Lampung.
- Pratama, I Putu Agus Eka. 2014. *Handbook Jaringan Komputer*. Bandung: Informatika Bandung.
- _____. 2016. *Integrasi dan Migrasi Sistem*. Bandung: Informatika Bandung.
- Rama, Girish Maskeri dan Avinash Kak. 2013. *Software – Practice and Experience*. New Jersey: Wiley Online Library.
- Republik Indonesia. 2008. *Peraturan Menteri Kesehatan Republik Indonesia Nomor 269/MENKES/PER/III/2008 tentang Rekam Medis*. Jakarta: Kementrian Kesehatan.
- Republik Indonesia. 2009. *Undang-Undang No. 36 Tahun 2009 tentang Kesehatan*. Jakarta: Sekretariat Negara.
- Suryantara, I Gusti Ngurah. 2017. *Merancang Aplikasi dengan Metodologi Extreme Programmings*. Jakarta: Elex Media Komputindo.
- Sutarbi, Tata. 2016. *Sistem Informasi Manajemen (Edisi Revisi)*. Yogyakarta: Penerbit Andi.
- Scribner, Kenn dan Seely Scott. 2009. *Effective REST Services via .NET: For .NET Framework 3.5 (1st Edition)*. Addison-Wesley Professional.

LAMPIRAN