BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu ini menjadi salah satu acuan penulis dalam membuat laporan akhir sehingga dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang dilakukan. Berikut merupakan penelitian terdahulu berupa beberapa jurnal yang terkait dengan judul laporan akhir penulis.

Tabel 2.1 Penelitian Terdahulu

Nama Peneliti	Judul Penelitian	Hasil Penelitian
Gita, Aminudin, dan Yuda. 2010	Implementasi Web Service Untuk Mendukung Interoperabilitas Pada Aplikasi E-Commerce	Penelitian yang dilakukan Gita, Aminudin dan Yuda menghasilkan sebuah sistem *e-commerce* yang fleksible karena tidak lagi tergantung pada *platform* komponen yang terlibat didalamnya serta mampu mengelola dan mendukung interoperabilitas antar aplikasi. Teknologi yang digunakan untuk membangun *web service* ini adalah bahasa pemrograman PHP dengan arsitektur SOAP dan format data XML.

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Nama Peneliti	Judul Penelitian	Hasil Penelitian
Deviana, Hartati. 2011	Penerapan *XML Web Service* Pada Sistem Distribusi Barang	Penelitian ini menghasilkan sebuah sistem client dan server pada aplikasi apotek pusat cabang/outlet dapat berkomunikasi dan bertukar data dengan dukungan teknologi *web service* yang dapat memberikan informasi transaksi data pada proses pelayanan pemesanan maupun pengiriman barang secara lengkap. Teknologi *web service* ini menggunakan arsitektur SOAP dengan format data XML dan bahasa pemrograman PHP.
Hidayat dan Ashari. 2013	Penerapan Teknologi Web Service Untuk Integrasi Layanan Puskesmas dan Rumah Sakit	Penelitian yang dilakukan oleh Hidayat dan Ashari menghasilkan sebuah prototipe sistem yang berupa integrasi sistem layanan informasi kesehatan berbasis *web service* dengan arsitektur SOAP dan format data

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Nama Peneliti	Judul Penelitian	Hasil Penelitian
		XML yang dapat menyatukan beberapa sistem yang berbeda dari segi bahasa pemrograman (ASP.NET, PHP) dan databasenya (SQL Server, MYSQL).
Purnawansyah dan Amaliah. 2014	Implementasi Web Service Pada Aplikasi Sistem Informasi Akademik Dengan Platform Mobile	Penelitian yang dilakukan Purnawansyah dan Amalia menghasilkan sistem KRS Online Fakultas Ilmu Komputer UMI Makassar yang lebih efisien dan efektif. Sistem dengan teknologi *web service* ini dapat mengelolah dan memberikan informasi berupa data mahasiswa, dosen, mata kuliah, nilai serta data akademik lainnya. Teknologi yang digunakan untuk membangun *web service* iniadalah SOAP, XML dan bahasa pemrograman PHP.

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Nama Peneliti	Judul Penelitian	Hasil Penelitian
Bramwell, Rizal, dan Oktavian. 2014	Rancang Bangun Web Service Perpustakaan Universitas Sam Ratulangi	Penelitian ini memperoleh hasil yaitu *service* dengan fungsi-fungsi yang mendukung proses administratif perpustakaan dikembangkan dengan dua buah aplikasi *end-user* dalam bentuk *desktop application* dan web yang sudah terintegrasi dengan *web service* yang dibangun menggunakan arsitektur SOAP dengan format data JSON dan bahasa pemrograman Java, sehingga fungsi-fungsi *service* yang sudah dibuat dapat dikembangkan melalui aplikasi *end-user*.

Dari 5 penelitian terdahulu tersebut telah diperoleh beberapa perbedaan teknis maupun non-teknis yang dilakukan oleh peneliti, untuk perbedaan teknisnya yaitu dari arsitektur protokol dan format data *web service* yang digunakan. Alasan penulis menggunakan arsitektur protokol REST dikarenakan kesederhanaan arsitekturnya dan juga menggunakan format data JSON dikarenakan JSON lebih minim ukuran jika dibandingkan dengan XML serta kemudahan untuk dibaca manusia.

2.2 Pengertian Web Service

W3C mendefinisikan *web service* sebagai sebuah sistem perangkat lunak yang dirancang untuk mendukung komunikasi dan interaksi antar mesin ke mesin (*Machine to Machine*) melalui sebuah *network* (jaringan). *Web Service* juga termasuk *WebAPIs* yang dapat diakses melalui jaringan seperti misalnya internet, dan dieksekusi melalui sebuah sistem jarak jauh sesusai dengan layanan yang diminta. Definisi *Web Service* menurut W3C juga meliputi banyak sistem berbeda, tetapi pada umumnya lebih menyangkut pada client dan server yang berkomunikasi menggunakan XML yang memenuhi standar SOAP (*Simple Object Access Protocol*). Asumsi secara umum adalah pada terminologi terdapat deskripsi dari mesin yang layanannya disediakan oleh server, atau sama seperti konsep dari WSDL. WSDL bukan termasuk standard dari SOAP tetapi merupakan syarat mutlak untuk *client-side* otomatis pada *framework* Java dan .NET SOAP. Beberapa organisasi industri seperti WS-I mengklaim baik SOAP dan WSDL sebagai definisi sari *Web Service*.

Selain SOAP dengan XML nya terdapat jenis *engine web service* lainnya yang banyak diimplementasikan pada aplikasi web, yaitu REST. REST *web service* atau yang kadang disebut RESTful *web service* atau RESTful API adalah web service yang mengimplementasikan arsitektur REST. Pada arsitektur REST, setiap *service* atau layanan dipandang sebagai sebuah *Resources* yang diidentifikasikan melalui URL. *Web service* REST memiliki karakteristik sebagai berikut :

1. Menggunakan *method* HTTP secara eksplisit
2. Memiliki struktur direktori URI
3. Pesan yang ditransfer dalam format XML, JSON atau keduanya

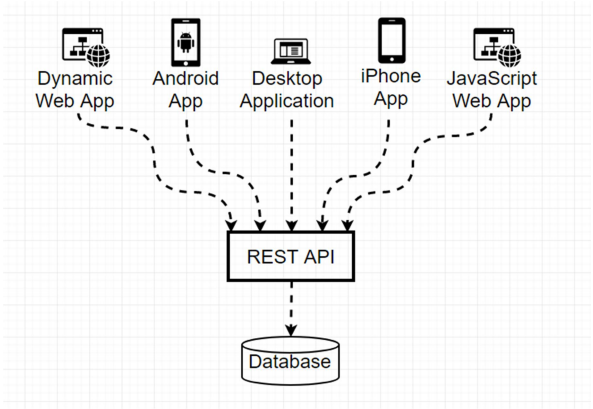
(Abdul dkk, 2013:2).

2.3 Pengertian API (Application Programming Interface)

Gambar 2.1 Logo API (*Application Programming Interface*)

API merupakan *software interface* yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk *library* dan menjelaskan bagaimana agar suatu *software* dapat berinteraksi dengan *software* lain. Penjelasan ini dapat dicontohkan dengan analogi apabila akan dibangun suatu rumah. Dengan menyewa kontraktor yang dapat menangani bagian yang berbeda, pemilik rumah dapat memberikan tugas yang perlu dilakukan oleh kontraktor tanpa harus mengetahui bagaimana cara kontraktor menyelesaikan pekerjaan tersebut. Dari analogi tersebut, rumah merupakan *software* yang akan dibuat, dan kontraktor merupakan API yang mengerjakan bagian tertentu dari *software* tersebut tanpa harus diketahui bagaimana prosedur dalam melakukan pekerjaan tersebut (Webber dkk, 2010:3).

2.4 Pengertian REST (REpresentational State Transfer)



Gambar 2.2 Desain REST API

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

Kekhasan REST terletak pada interaksi antara klien dan *server* yang difasilitasi oleh sejumlah tipe operasional (verba) dan *Universal Resource Identifiers* (URIs) yang unik bagi tiap-tiap sumberdaya. Masing-masing verba *GET*, *POST*, *PUT* dan *DELETE* memiliki makna operasional khusus untuk menghindari ambiguitas. REST kerap dipergunakan dalam *mobile application*, situs web jejaring sosial, *mashup tools*, dan *automated business processes*.

Arsitektur REST yang *decoupled* (terpisah) serta beban komunikasi yang ringan antara produsen dan konsumen membuatnya populer di dunia *cloud-based API*, seperti yang disajikan oleh Amazon, Microsoft, dan Google.Layanan berbasis web yang menggunakan arsitektur REST semacam itu dinamakan RESTful APIs (*Application Programming Interfaces*) atau REST APIs (Sumber: ekajogja.com).

2.5 Pengertian JavaScript

Gambar 2.3 Logo *JavaScript*

JavaScript merupakan salah satu bahasa *script website* yang paling banyak digunakan untuk menambah manipulasi *script* HTML dan CSS pada sisi *client/browser*. *JavaScript* mampu memberikan fungsionalitas lebih pada *website*, seperti validasi form, berkomunikasi dengan *server* serta membuat *website* lebih interaktif dan animatif.

JavaScript digunakan pada banyak *browser* seperti *Internet Explorer, Firefox, Chrome, Opera, Safari* dan lain sebagainya. Hampir seluruh *browser* mendukung *JavaScript* sehingga tidak perlu khawatir kode *JavaScript* yang digunakan pada *website* tidak berfungsi (Wahana Komputer, 2012:2).

2.6 Pengertian JSON (JavaScript Object Notation)

Gambar 2.4 Logo JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan

dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa pemrograman *JavaScript*. JSON terdiri dari dua struktur, yaitu:

1. Pasangan nama dengan nilai. Pada beberapa bahasa hal ini dinyatakan sebagai *object*, *record*, *struct*, *dictionary*, *hashtable*, *keyedlist* atau *associative array*.
2. Daftar nilai terurutkan (*anorderedlist of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai *array*, *vector*, *list* atau *sequence* (Abdul dkk, 2013:2).

2.7 Pengertian Node.js (JavaScript Runtime)

Gambar 2.5 Logo *Node.js*

Node.js adalah perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web dan ditulis dalam sintaks bahasa pemrograman *JavaScript*. Bila selama ini kita mengenal *JavaScript* sebagai bahasa pemrograman yang berjalan di sisi client / *browser* saja, maka *Node.js* ada untuk melengkapi peran *JavaScript* sehingga bisa juga berlaku sebagai bahasa pemrograman yang berjalan di sisi server, seperti halnya PHP, Ruby, Perl, dan sebagainya. *Node.js* dapat berjalan di sistem operasi Windows, Mac OS X dan Linux tanpa perlu ada perubahan kode program.

Berbeda dengan bahasa pemrograman sisi server pada umumnya yang bersifat sinkronis atau *blocking*, *Node.js* bersifat asinkronis atau *non-blocking* sebagaimana halnya *JavaScript* bekerja. *Node.js* berjalan dengan basis *event* (*event-driven*). Maksud dari *blocking* secara sederhana adalah bahwa suatu kode

program akan dijalankan hingga selesai, baru kemudian beralih ke kode program selanjutnya (Sheba, 2017:8).

2.8 Pengertian Socket.io (JavaScript Library)

*Socket.io*merupakan sebuah *library JavaScript* yang membantu dalam pembuatan aplikasi web yang *real-time* lebih mudah, dengan menggunakan *socket.io* kita dapat menghubungkan antara *client* dan *server* dapat terjadi secara *bidirectional* (dua arah). Maksudnya yaitu kita dapat menghubungkan *client* dan *server* sehingga dapat berperan sebagai pengirim dan sekaligus penerima data, komponen yang terdapat pada *socket.io* terdiri dari dua bagian yang pertama *client-side* yaitu yang berjalan pada *browser*, dan *server-side* yang dapat digunakan sebagai modul untuk *Node.js* (Sumber: www.kursuswebsite.org).

2.9 Pengertian Express.js (Web Framework)

Gambar 2.6 Logo *Express.js*

Express.js adalah satu *web framework* paling populer di dunia *Node.js*. Dokumentasinya yang lengkap dan penggunaannya yang cukup mudah, dapat membuat kita mengembangkan berbagai produk seperti aplikasi web ataupun *RESTful* API. *Express.js* pun dapat digunakan menjadi pijakan untuk membangun *web framework* yang lebih kompleks seperti, *Sails.js*, MEAN (*MongoDB*, *Express.js*, *Angular.js*, *Node.js*) dan MERN (*MongoDB*, *Express.js*, *React.js*, *Node.js*). *Express.js* dibuat oleh TJ Holowaychuk dan sekarang dikelola oleh komunitas (Sumber: arfianhidayat.com).

2.10 Pengertian MongoDB

Gambar 2.7 Logo *MongoDB*

MongoDB merupakan sebuah sistem basis data yang berbasis dokumen (*Document Oriented Database*) dan termasuk sistem basis data yang menganut paham NoSQL. NoSQL singkatan dari *Not Only SQL*, artinya sebuah sistem basis data tidak hanya harus menggunakan perintah SQL untuk melakukan proses manipulasi data.

MongoDB tidak memiliki tabel, kolom, dan baris. Pada *MongoDB* yang ada hanyalah koleksi dan dokumen. Dokumen yang terdapat dalam MongoDB dapat memiliki atribut yang berbeda dengan dokumen lain walaupun berada dalam satu koleksi. Hal ini tidak dapat dilakukan dalam RDBMS, dimana sebuah baris dalam tabel tidak mungkin memiliki kolom yang berbeda dengan baris yang lain jika berada dalam satu tabel.

MongoDB merupakan sistem basis data yang menggunakan konsep *key-value*, artinya setiap dokumen dalam *MongoDB* pasti memiliki *key*. Hal ini berbeda dalam RDBMS yang bisa tidak menggunakan *primary key* ketika membuat sebuah tabel. Sehingga walaupun kita membuat sebuah dokumen tanpa menggunakan *primary key*, tapi secara otomatis *MongoDB* memberinya sebuah *key*. Penggunaan konsep *key-value* sangat berperan penting, karena hal ini membuat *MongoDB* menjadi sistem basis data yang sangat cepat jika dibandingkan dengan *non key-value* seperti RDBMS (Eko dan Fitry, 2016:69).

2.11 Pengertian Black Box Testing

Menurut Simarmata (2009) menyatakan pengujian adalah sebuah proses terhadap aplikasi atau program untuk menemukan segala kesalahan dan segala kemungkinan yang akan menimbulkan kesalahan sesuai dengan spesifikasi perangkat lunak yang telah ditentukan sebelum aplikasi tersebut diserahkan kepada pengguna. Salah satu module untuk melakukan pengujian pada perangkat lunak yaitu *black box testing.*

Black Box Testing yang terfokus pada apakah unit program memenuhi kebutuhan (*requirement*) yang disebutkan dalam spesifikasi. Pada *black box testing,* cara pengujian hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses yang diinginkan (Emilda, 2018:14).

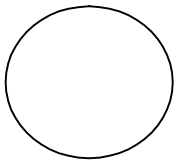
2.12 Pengertian Data Flow Diagram (DFD)

Data Flow Diagram (DFD) disebut juga dengan Diagram Arus Data (DAD). DFD adalah: suatu model logika data atau proses yang dibuat untuk menggambarkan: darimana asal data, dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut, dan interaksi antara data yang tersimpan, dan proses yang dikenakan pada data tersebut (Kristanto, 2008).

Data Flow Diagram atau dalam bahasa Indonesia menjadi Diagram Alir Data (DAD) adalah refresentasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengatur dari masukan (input) dan keluaran (output). DFD tidak sesuai untuk memodelkan sistem yang menggunakan pemograman berorientasi objek (Sukamto, 2014).

Notasi-notasi pada DFD (Edward Yourdon dan Tom DeMarco) adalah sebagai berikut:

Tabel 2.2 Simbol-simbol Data *Flow* Diagram (DFD)

NOTASI	KETERANGAN
	Proses atau fungsi atau prosedur; pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemograman terstruktur, maka pemodelan notasi inilah yang harusnya menjadi fungsi atau prosedur di dalam kode program.
	*File* atau basis data atau penyimpanan (*storage*) pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemograman terstruktur, maka pemodelan notasi inilah yang harusnya dibuat menjadi tabel-tabel basis data yang dibutuhkan, tabel-tabel ini juga harus sesuai dengan perancangan tabel-tabel basis data yang dibutuhkan (*Entity Relationship Diagram* (ERD), *Conceptual Data Model* (CMD), *Physical Data Model* (PDM).
	Entitas luar (*external entit*y) atau masukan (*input*) atau keluaran (*output*) atau orang yang memakai atau berinteraksi dengan perangkat lunak yang dimodelkan atau sistem lain yang terkait dengan aliran data dari sistem yang dimodelkan.
	Aliran data merupakan data yang di kirim antar proses, dari penyimpanan ke proses, atau dari proses ke masukan (*input*) atau keluaran (*output*).

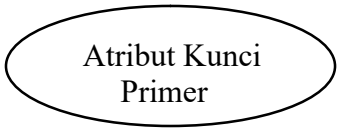
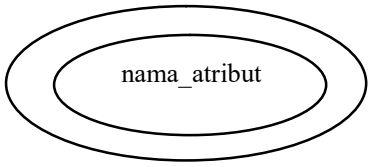
(Sumber :Sukamto, 2014).

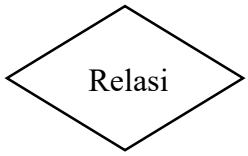
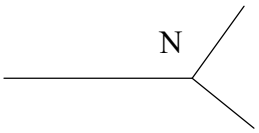
2.13 Pengertian Entity Relationship Diagram (ERD)

Entitiy Relationship Diagram (ERD) adalah pemodelan awal basis data yang akan dikembangkan berdasarkan teori himpunan dalam bidang matematika untuk pemodelan basis data relasional. ERD memiliki beberapa aliran notasi seperti notasi *Chen* (dikembangkan oleh Peter Chen). (Sukamto, 2014).

Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi Chen:

Tabel 2.3 Simbol-simbol *Entity Relationship* Diagram (ERD)

SIMBOL	DESKRIPSI
	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data; benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel.
	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa id; kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).
Atribut multinilai/*multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.

	Relasi yang menghubungkan antar entitas, biasanya diawali dengan kata kerja.
Asosiasi/*association* 	Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian, keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan *one to many* manghubungkan entitas A dan entitas B.

(Sumber :Sukamto, 2014).