BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian ini menggunakan beberapa sumber pustaka yang berhubungan dengan kasus atau metode yang akan diteliti, Diantaranya yaitu :

Heri Purnama (2016), telah melakukan penelitian tentang Aplikasi Pengolahan Skripsi Di STMIK AKAKOM YOGYAKARTA Menggunakan Arsitektur *Microservice* Dengan Node.js. Aplikasi dibuat bertujuan untuk mengecek duplikasi skripsi dengan berdasarkan judul dari skripsi. Dibuat dengan Bahasa pemrograman JavaScript pada Node.js, menggunakan database MongoDB yang mengimplementasikan arsitektur *Microservice*.

Setiyono Suryo Asmoro (2017), telah melakukan penelitian tentang Aplikasi Pencarian Barang Hilang Di KOTA SOLO Berbasis Web, Aplikasi ini bertujuan untuk memudahkan masyarakat di Kota Solo yang kehilangan barang atau mencari informasi tentang barang hilang ataupun barang temuan, serta dapat mengurangi resiko pengakuan barang temuan. Dibuat dengan bahasa PHP dan database menggunakan MySQL dengan metode *system development life cycle* pendekatan *waterfall.*

Riko (2019), telah melakukan penelitian tentang Implementasi Rest Api Untuk Sistem Penjadwalan Pendadaran dan Seminar Proposal Skripsi Mahasiswa (Studi Kasus Program Studi Sistem Informasi STMIK AKAKOM). Aplikasi bertujuan untuk penjadwalan ujian pendadaran dan seminar proposal mahasiswa

program studi Sistem Informasi STMIK AKAKOM yang berintegrasi dengan Whatsapp dengan menerapkan REST API. Dibuat dengan bahasa pemrograman PHP dan menggunakan database MySQL.

Andre Putra (2020), telah melakukan penelitian tentang RESTful API Untuk Menampilkan List Berita Menggunakan Arsitektur *Microservice*s untuk ditampilkan di *frontend*. Aplikasi bertujuan untuk membangun sebuah sistem yang lebih dinamis dan berkembang dengan memanfaatkan teknologi Arsitektur *Microservice*. Adapun *tools* yang diguanakan yaitu Bahasa pemrograman Python, menggunakan database PostgreSQL.

Pembahasan yang dibuat kali ini tentang "Implementasi Arsitektur *Microservice* Untuk Sistem Pencarian Barang Hilang Menggunakan Restful API". Penelitian ini bertujuan untuk menyelesaikan masalah melalui aplikasi yang penulis buat, berupa informasi berita kehilangan pada seluruh warga STMIK Akakom. Dibuat dengan Bahasa pemrograman PHP menggunakan database MySQL.

Tabel tinjauan pustaka merupakan tabel yang dibuat untuk mendefinisikan penelitian yang sebelumnya hampir sama dilakukan dengan penelitian yang diajukan saat ini, adapun perbandingan yang menjadi tabel tinjauan pustaka penelitian yakni :

Table 2.1 Tinjauan Pustaka

Penulis	Judul	Masalah	Teknologi	Hasil
Heri Purnama (2016).	Aplikasi Pengelolaan Skripsi Di STMIK AKAKOM Yogyakarta Menggunakan Arsitektur *Microservice* Dengan Node.js .	Membangun Aplikasi Pengelolaan Skripsi Menggunakan Arsitektur *Microservice* Dan Perangkat Lunak Node.js .	Docker, *Microservice* , MongoDB, Node.js .	Aplikasi Untuk Pengelolaan Skripsi Menggunakan Node.js .
Setiyono Suryo Asmoro (2017).	Aplikasi Pencarian Barang Hilang Di Kota Solo Berbasis Web.	Membangun Aplikasi Pencarian Barang Hilang Di Kota Solo Dengan System Development Life Cycle Pendekatan Waterfall.	PHP, MySQL	Mengembangkan Aplikasi Pencarian Barang Hilang Di Kota Solo Berbasis Web.
Riko (2019)	Implementasi Rest Api Untuk Sistem Penjadwalan Pendadaran dan Seminar Proposal Skripsi Mahasiswa (Studi Kasus Program Studi Sistem Informasi STMIK AKAKOM)	Membangun Aplikasi Untuk Penjadwalan Ujian Pendadaran Dan Seminar Proposal Skripsi Mahasiswa Menggunakan Implementasi Rest Api.	PHP, MySQL, *Microservice* , RESTful API	Aplikasi Untuk Sistem Penjadwalan Pendadaran Dan Seminar Proposal Skripsi Mahasiswa.
Andre Putra (2020)	RESTfull API Untuk	Bagaimana caranya	Javascript, Vuejs,	Telah Dibuat REST API

Penulis	Judul	Masalah	Teknologi	Hasil
	Menampilkan List Berita Menggunakan Arsitekrut *Microservice* (Studi Kasus : Portal Berita)	membuat REST API Untuk Menampilkan List Berita Menggunakan Konsep *Microservice* Dan Dapat Ditampilkan Di Frontend	Axios, Python, Postman, MySQL.	Untuk Menampilkan List Berita Menggunakan Konsep *Microservice* Dan Dapat Ditampilkan Di Frontend.
Wisnu Andrian (2021).	Implementasi Arsitektur *Microservice* Untuk Sistem Pencarian Barang Hilang Menggunakan Restfull API (Studi Kasus Papan Informasi Kehilangan STMIK AKAKOM)	Membangun Sistem Pencarian Barang Hilang Dengan Memanfaatkan Arsitektur *Microservice* Menggunakan *RESTful API.*	PHP, Javascript, *Microservice* , *Codeigniter* , MySQL, RESTful API.	Akan Dibuat Sistem Pencarian Barang Hilang Dengan Memanfaatkan Arsitektur *Microservice* Menggunakan *RESTful API.*

2.2 Dasar Teori

2.2.1 *Microservice*

Microservices adalah sebuah pendekatan untuk mengembangkan aplikasi dengan rangkaian *service-service* yang kecil, yang mana setiap *service* berjalan pada prosesnya sendiri-sendiri. Aplikasi dipecah tidak sekedar hanya memisahkan antara *backend* dan *frontend*, melainkan dipecah menjadi *service-service* yang berfungsi secara spesifik. Apabila *client* menjalankan layanan

website blog, maka dengan pendekatan *Microservices* aplikasi bisa di *breakdown* menjadi layanan khusus konten, statistik, ad, *reporting*, photo/video, *messaging*, *searching*, komentar dan lainnya, maka setiap *service* dapat berkomunikasi dengan mekanisme yang ringan.

Setiap *service* yang dibuat harus mengenkapsulasi data dengan logika bisnis yang beroperasi pada data itu sendiri, dan hanya dapat diakses melalui *published service interface*. Tidak ada database yang dapat diakses secara langsung dari luar *service* dan tidak ada data yang dibagikan (*sharing*) antara setiap *service*.

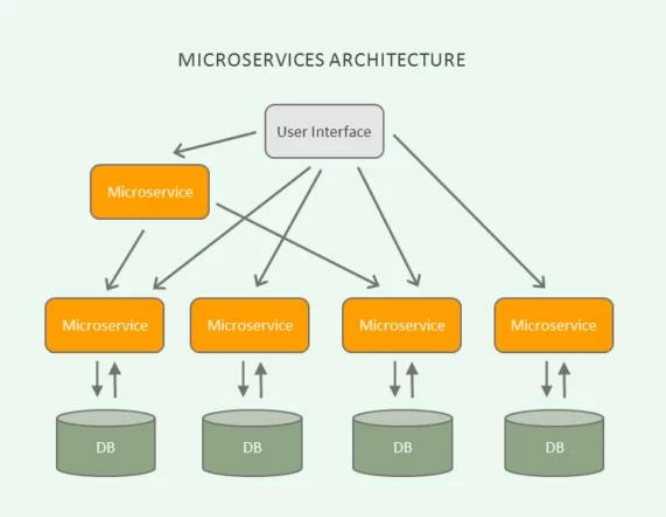
Oleh karena itu, setiap *service* yang dibangun harus memiliki landasan domainnya sendiri-sendiri dan process *sharing* data domain yang satu dengan yang lainnya hanya dapat dilakukan melalui *published service interface*.

Microservices memperkuat struktur modular yang sangat penting bagi tim yang sangat besar. Menurut Martin, Fowler ini adalah *key benefit* yang juga aneh jika dikatakan kelebihan, karena tidak ada alasan apapun mengapa *Microservice*s memiliki struktur modular yang lebih kuat daripada monolithic.

Dalam *arsitektur monolithic* pada umumnya, sangatlah mudah bagi *developer* untuk melewati batas. Umumnya digunakan untuk mencari jalan pintas dalam mengimplementasikan fitur dengan lebih cepat. Akan tetapi berakhir pada merusak struktur modular yang berimplikasi pada penurunan produktivitas Tim.

Service yang sederhana lebih mudah di *deploy* dan digunakan, karena berdiri sendiri, kecil kemungkinan kegagalan sistem terjadi saat salah satu *service* mengalami kesalahan.

Penggunaan *Microservice*s, dapat mencampur dan menggunakan beragam bahasa pemrograman, framework, dan teknologi penyimpanan database yang digunakan.(Refractory, 2017)



Gambar 2.1 *Microservice*s *Architecture*

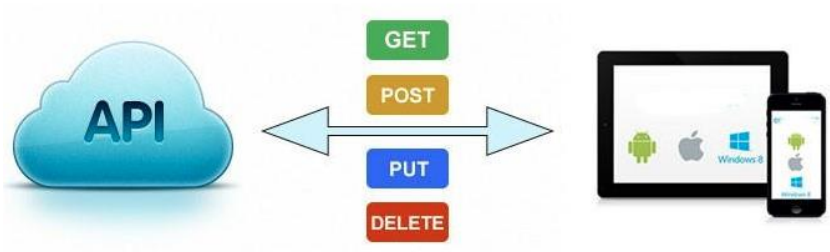
2.2.2 RESTful API

RESTful API didasarkan pada teknologi *state transfer* (*representational state transfer / REST*), gaya arsitektur dan pendekatan komunikasi yang sering digunakan dalam pengembangan layanan *web*.

Meskipun REST dapat digunakan di hampir semua protokol, tapi biasanya memanfaatkan HTTP ketika digunakan untuk Web API. Hal ini membantu pengembang *web* tidak perlu menginstal *library* atau perangkat lunak tambahan untuk memanfaatkan desain REST API. *Design* REST API pertama kali diperkenalkan oleh Dr. Roy Fielding dalam disertasi doktor tahun 2000-nya. REST API terkenal karena fleksibilitasnya yang luar biasa. Data tidak terikat dengan metode dan sumber daya, REST memiliki kemampuan untuk menangani

beberapa jenis panggilan, mengembalikan format data yang berbeda dan bahkan mengubah secara struktural tentunya dengan implementasi yang benar.

REST yang digunakan oleh *browser* dapat dianggap sebagai bahasa internet. Dengan meningkatnya penggunaan *cloud*, API muncul untuk mengekspos layanan *web.* REST adalah pilihan logis untuk membangun API yang memungkinkan pengguna untuk terhubung dan berinteraksi dengan layanan *cloud*. API telah banyak digunakan oleh situs-situs seperti Amazon, Google, LinkedIn dan Twitter. (Yudana, 2019)



Gambar 2.2 Arsitektur RESTful API

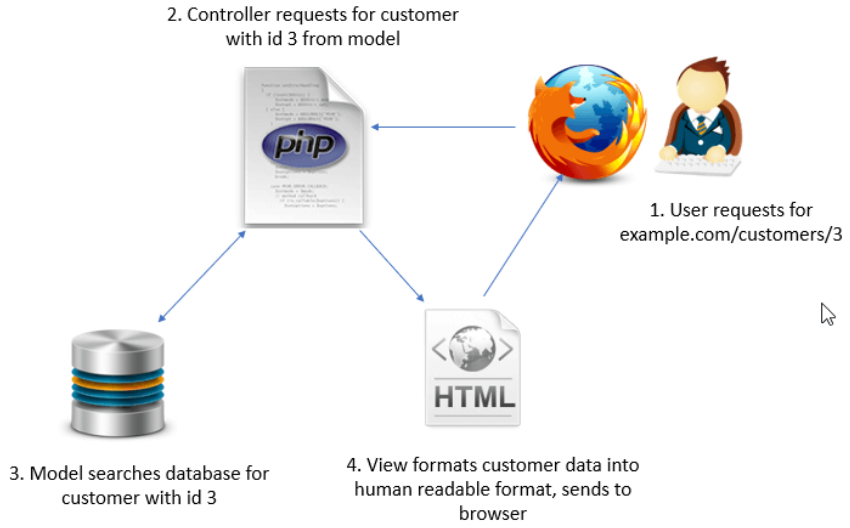
2.2.3 *Codeigniter*

Codeigniter adalah *framework* PHP yang digunakan untuk mengembangkan aplikasi, keseluruhan *source code* pada *Codeigniter* hanya sebesar 2 MB, yang mana membuatnya menjadi mudah dipelajari dan cara kerjanya, fitur *built-in* pada *Codeigniter* didesain agar bekerja secara independen tanpa harus bergantung pada komponen lainnya.

Framework Codeigniter menggunakan desain arsitektur MVC, *Codeigniter* didokumentasikan dengan baik, dan memiliki *library* yang mirip dengan bahasa program PHP yang lain, dalam *Codeigniter user* meminta

informasi, *controller* merespon permintaan tersebut, dan menyajikan informasi sesuai yang diminta.

Controller menerima permintaan dari pengguna lalu berinteraksi dengan model database jika perlu kemudian mengembalikan hasilnya kembali ke *browser* dalam bentuk kode HTML yang ditafsirkan oleh *browser* menjadi format yang dapat dibaca manusia dan ditampilkan kepada pengguna. (Techfor.id, 2020)



Gambar 2.3 Konsep MVC

2.2.4 MySQL

Menurut Arief (2011d:152) **“**MySQL adalah salah satu jenis database server yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan database sebagai sumber dan pengolahan datanya”.

Menurut Adi Nugroho (2011) MySQL (*My Structured Query Language*) adalah: “ Suatu sistem basis data *relation* atau *Relational Database management System* (RDBMS) yang mampu bekerja secara cepat dan mudah digunakan MySQL juga merupakan program pengakses database yang bersifat jaringan,

sehingga dapat digunakan untuk aplikasi multi *user* (banyak pengguna). MySQL didistribusikan gratis dibawah lisensi GPL (*General Public License*). Dimana setiap program bebas menggunakan MySQL namun tidak bisa dijadikan produk turunan yang dijadikan *closed source* atau komersial".

MySQL merupakan database yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pembangun aplikasi *web* yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman script PHP.

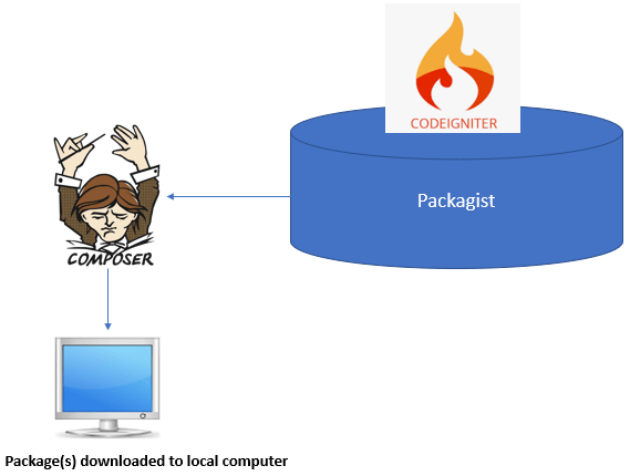
2.2.5 Composer

Composer adalah *Dependecy Manager* yang ada di dalam pemrogramman PHP. *Dependency* adalah bergantung, atau bisa di sebut banyak file yang bergantung dalam file-file yang lain, dimana satu program bergantung pada program yang lainnya.

Dengan menggunakan *Composer developer* dapat dengan mudah menambahkan kode *library* dan *plugin* untuk aplikasi yang dibuat. Di *framework Codeigniter* biasanya akan menambahkan kode *library* dan *plugin* secara manual di dalam direktori "*application/libraries*" atau "*application/third_party*". Hal ini tentu akan menyulitkan ketika akan melakukan *update* atau ada pembaruan dari pembuat kode *library* yang di gunakan. Karena setiap *developer* harus

memperbaharui secara manual. Dengan Composer proses instalasi, *update*, dan hapus *plugin* atau *library* bisa dilakukan secara otomatis.

Composer memiliki sistem *autoload* yang sudah menggunakan standar penulisan PSR-4 dll. Ini membuat projek *web* dengan konsep MVC akan lebih terstruktur. Autoload composer membuat pemanggilan *class* dari file lain semakin mudah. Dengan adanya standar ini, membuat pengembangan *web* secara tim semakin mudah (Gede Arya S.G, 2020)



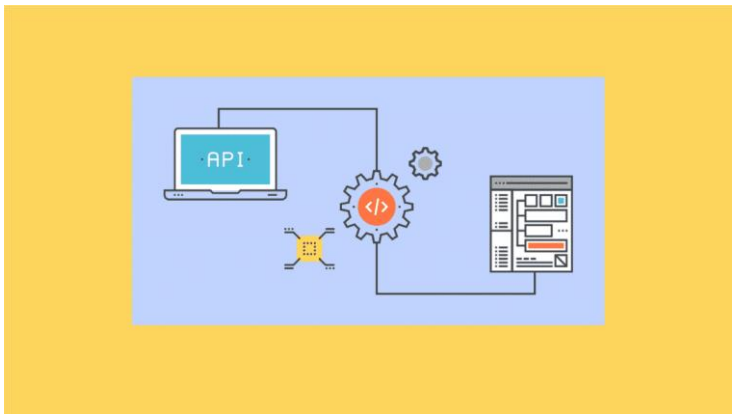
Gambar 2.4 Cara Kerja Composer

2.2.6 Guzzle

Guzzle merupakan salah satu pilihan *library* PHP yang memberikan fungsionalitasi *client* http lebih mantap dibanding dari bawaan PHP. Guzzle popular digunakan oleh banyak orang sebagai *tool* untuk melakukan scrapping data (Wiryosukiro Wagiman, 2020).

Guzzle merupakan independen HTTP *client* untuk PHP. Library PHP ini sangat berguna karena memungkinkan pengguna untuk mengirim permintaan HTTP dengan mudah. Guzzle juga memiliki *simple interface* yang dapat

digunakan untuk membangun *query strings, POST requests*, menggunakan HTTP *cookies*, mengupload JSON data, dan lain-lain (Wdi, 2020)



Gambar 2.5 Guzzle