BAB II
TINJAUAN PUSTAKA

2.1 *Cloud Computing*

Sistem teknologi *cloud computing* atau bisa disebut juga dengan komputasi awan saat ini sering digunakan dalam melakukan pengembangan aplikasi. *cloud computing* dalam era teknologi yang pesat dan sudah banyak dikenal dan dilakukan perkembangan yang sangat cepat. Karena secara *virtual* menjadikan Perusahaan besar di berbagai bidang kini sudah mulai fokus dan tidak asing lagi dalam penggunaan sistem *cloud computing*. Penggunaan *cloud computing*, pengguna diperlukan untuk melakukan penyewaan berberapa layanan penyedia komponen kerja dan server penyimpanan data hingga data *center* dalam melakukan pengembangan aplikasi.

Cloud computing memberikan banyak layanan dalam bentuk bisnis melalui Internet. penggunaan *platform cloud* dalam melakukan proses *cloud computing* dapat membuat tidak perlu melakukan kegiatan seperti tanggung jawab untuk melakukan pemeliharaan layanan, hanya perlu fokus pada masalah teknis yang lebih penting. Layanan *cloud computing* dapat membuat infrastruktur yang lebih efisien dan mampu membuat layanan yang dapat di atur sesuai kebutuhan pengguna. *Cloud computing* menurut Google merupakan sebuah *central* data yang ditempatkan pada luar komputer atau secara *virtual* dari aplikasi pengguna. Google memberikan fitur kontribusinya dalam sistem *cloud computing* yang memiliki fungsi utama agar dapat memanfaatkan sumber daya komputer dalam jumlah yang sangat besar dan terdiri dari banyaknya komputer tersebar di seluruh dunia, dapat digunakan untuk mempercepat operasi yang dilakukan oleh aplikasi modern [20].

Terdapat beberapa sifat yang dimiliki dalam sistem *cloud computing*, sifat tersebut diantaranya adalah sesuai permintaan dari penggunan layanan, elastis yang berarti dapat diatur sesuai kebutuhan, terukur yang membuat *developer* dapat mengetahui rancangan berkelanjutannya dan sepenuhnya dikelola oleh penyedia atau *provider* yang memudahkan dalam menggunakan layanan *cloud*. Layanan yang bersifat sesuai permintaan menjelaskan bahwa penggunaan layanan

cloud dapat bebas memilih dan memanfaatkan layanan-layanan yang telah diberikan oleh penyedia layanan *cloud*. Karakteristik elastis dan terukur menjelaskan mengenai bagaimana fleksibilitas dan kemampuan ruang yang dapat diminta pengguna ketika menggunakan satuan kecepatan data tertentu, sementara karakter layanan terkahir yaitu untuk dapat dikendalikan sepenuhnya oleh penyedia layanan, yang dapat memfasilitasi kemampuan pengguna untuk meringankan dalam memproses dan berbagi data dan dokumen dengan perangkat lain [21].

Cloud computing terdapat tiga jenis dasar penyusunnya layanan dari *cloud computing*, yaitu diantaranya *Software as a Service* (SaaS), *Platform as a Service* (PaaS), dan *Infrastructure as a Service* (IaaS). *Software as a Service* (SaaS) merupakan beberapa layanan *cloud computing* yang pertama kali dikenal oleh pengguna. Jenis SaaS ini merupakan inovasi selanjutnya dari konsep dari ASP (*Application Service Provider*). SaaS dapat mempermudah penggunanya dalam berlangganan, oleh karena itu pengguna SaaS tanpa harus mengeluarkan dana dalam pengembangan internal atau membeli lisensi. *Platform as a Service* (PaaS) adalah layanan penyediaan modul-modul yang secara langsung siap pakai dalam membangun sebuah aplikasi yang hanya mampu beroperasi di atas basis aplikasi. Layanan PaaS pada sistem *cloud* memberikan tempat untuk membuat dan mendistribusikan aplikasi tanpa harus mengetahui banyaknya *processor* atau memori yang diperlukan aplikasi. *Infrastructure as a Service* (IaaS) memiliki tingkat di bawah PaaS. IaaS adalah layanan yang berikan fitur sewa sumber daya teknologi informasi dasar seperti sistem operasi, daya komputasi, media penyimpanan, memori, kapasitas jaringan, dan layanan lainnya yang dapat digunakan oleh pengguna dalam menjalankan aplikasi [22].

2.2 *GCP (Google Cloud Platform)*

GCP merupakan layanan *cloud computing* yang di sediakan oleh Google, dan itu adalah kumpulan administrasi komputasi awan. GCP memberikan landasan tahap manajemen dan kondisi pendaftaran tanpa *server*. Google memperkenalkan *App Engine* pada bulan April 2008, yang berfungsi sebagai tahapan untuk membuat dan memfasilitasi aplikasi dalam *server* yang diawasi

Google. Ini terutama berhubungan dengan administrasi komputasi terdistribusi. GCP merupakan unit Google yang menggabungkan GCP kerangka *cloud* terbuka, hanya menggunakan *G Suite*, industri adaptasi Android dan Chrome OS, plus aplikasi antarmuka pemrograman (API) yang ditujukan untuk *Artificial Inteligent* (AI) dan melakukan administrasi pemetaan [23].

Layanan produk yang GCP mampu digunakan untuk melakukan perancangan dalam membangun infrastruktur server ialah *Compute Engine, Cloud SQL, Cloud Storage, dan Container Registry*. Produk Google *cloud* memiliki fitur dan teknik yang dapat diterapkan untuk membangun infrastruktur *server* dengan ketersediaan tinggi seperti fitur *autohealing, multiple zones, load balancing, autoscaling, automatic updating, dan failover*. Infrastruktur *server* mampu dibuat dengan menggunakan *service product* yang ada pada teknologi GCP, sehingga mampu menghasilkan sistem yang andal dan memiliki ketersediaan tinggi [24].

2.3 *App Engine*

Google *App Engine* adalah salah satu jenis layanan yang dijelaskan pada pembahasan google *cloud platform* yaitu *Platform as a Service* (PaaS). Google *App Engine* dapat membuat pengguna mampu menjalankan aplikasi web pada infrastruktur Google. Menggunakan *App Engine* tidak perlu lagi sebuah *server*, aplikasi tersebut sudah dapat bisa digunakan oleh pengguna sebagai bagian dari *server* [25]. Sebagian besar dari program modern saat ini dapat dengan mudah dilakukan dengan ditulis oleh pemprogram yang ahli dalam bahasa pemprograman seperti java dan python. Komputasi tanpa menggunakan *server* dapat dijelaskan sebagai model komputasi awan yang menyediakan berbagai layanan *backend* dengan sesuai permintaan. Penyedia layanan *cloud* saat ini cenderung memberikan penyediaan, mendistribusikan, dan menagih pengguna dari sumber daya dan penyimpanan yang mereka gunakan. Oleh karena itu, pengguna tidak perlu melakukan pembayaran dengan tarif yang tetap untuk banyaknya *server* dan *bandwidth* yang digunakan atau dipesan, hanya mencakup *server* tempat *database* dan *file* dari aplikasi pengguna. Meskipun proses ini disebut *serverless computing*, *server* berbentuk fisik masih menjadi bagian dari

persamaan, namun dengan perangkat lunak seperti ini, pengguna tidak perlu khawatir terkait pemeliharaan dan penyediaan *server* [20].

Google App Engine mengelola sepenuhnya untuk mengembangkan dan *hosting* aplikasi *web* dalam skala besar. Pengguna dapat menentukan penggunaan dan rancangan bahasa, pustaka, dan kerangka kerja untuk melakukan pengembangkan aplikasi, kemudian *App Engine* dapat menangani penyediaan *server* dan pengaturan penggunaan *instance* aplikasi berdasarkan permintaan. *App engine* adalah PaaS untuk membangun aplikasi yang skalabel. *App engine* merupakan lingkungan standar dengan lingkungan terbatas dan dukungan untuk bahasa seperti Python, Go, node.js dan yang lainnya merupakan lingkungan yang memiliki keunggulan fleksibel di mana pengguna dapat memiliki lebih banyak kebebasan seperti menjalankan *runtime* menggunakan docker, waktu tunggu permintaan respons yang lebih lama, kemampuan yang dapat mengunduh dependensi atau perangkat lunak khusus, dan SSH ke dalam mesin *virtual* [20].

Membangun aplikasi *Serverless* dengan *backend* atau API dapat mendukung beberapa bahasa pengembangan tanpa perlu khawatir tentang dukungan infrastruktur. Saat pengguna memiliki aplikasi yang memerlukan komunikasi dengan beberapa layanan seperti aplikasi atau API, *App Engine* merupakan solusi yang sesuai untuk aplikasi *serverless* dengan *backend* dan API tanpa perlu khawatir dukungan infrastruktur. Hubungan antara layanan tersebut memungkinkan aplikasi diperlakukan sebagai entitas yang dikelola [20].

2.4 *Cloud SQL*

Basis data SQL dikenal memiliki sifat yang disingkat ACID (*Atomic, Consistent, Isolation dan Durability*). Sifat *Atomic* pada data SQL apabila ada sebuah transaksi yang memiliki dua atau lebih bagian data informasi, semua bagiannya harus disimpan atau semua komponennya tidak disimpan. Tidak ada sebagian saja komponen yang disimpan atau tidak disimpan. *Consistent* adalah data yang disimpan tidak boleh melanggar integritas basis data. Inkonsisten data dapat mengalami gangguan dan dibatalkan untuk memastikan basis data berada dalam kondisi sama seperti sebelum ada perubahan. *Isolation* adalah sebuah transaksi yang tidak dipengaruhi oleh transaksi lain saat sedang berjalan. Hal ini

dapat bertujuan untuk mencegah terjadinya pertukaran yang gagal antara data dan transaksi. *Durability* adalah apabila transaksi sudah disimpan di basis data secara permanen, untuk seterusnya transaksi tersebut ada di basis data meskipun terjadi kegagalan sistem [26].

Penyedia layanan *cloud* GCP mempunyai layanan yang disebut Google *Cloud* SQL yang merupakan *database* MySQL yang disediakan oleh Google *Cloud*. Layanan yang ditawarkan GCP ini memiliki semua kemampuan dan fungsi dari MySQL, dengan beberapa fitur tambahan yaitu dilakukan secara *cloud*. Google *Cloud* SQL dapat mudah digunakan, dan tidak memerlukan instalasi perangkat lunak yang sulit atau pemeliharaan, oleh karena itu sangat ideal pengembangan aplikasi kecil hingga menengah [25].

2.5 *Cloud Storage*

Cloud Storage atau penyimpanan awan merupakan suatu layanan penyimpanan *file* berbasis internet. Sistem penyimpanan *cloud* melakukan proses penyimpanan *file* yang dapat langsung dan diakses mana saja dan kapan saja selama penggunanya terhubung internet yang dapat mengakses langsung ke *Cloud Storage*. Konsep dari *Cloud Storage* seperti konsep *file server* yang dimiliki kantor perusahaan, namun yang membedakan adalah infrastrukturnya oleh media penyimpanan langsung dikelola oleh *provider Cloud* dan pemanfaatannya dapat dijadikan layanan penyimpanan *file* yang dapat diakses menggunakan internet. Sebelum awal kemunculan *Cloud Computing* seperti saat ini, layanan dari *Cloud Storage* lebih dikenal dengan istilah *virtual drive* yaitu hanya sebagai penyimpanan secara virtual, namun saat ini dengan adanya istilah *Cloud Computing* lebih dikenal dengan sebutan *Cloud Storage*. Pengguna *Cloud Storage* tidak memerlukan membawa atau merawat media penyimpanan seperti *hard drive* untuk *file* yang telah tersimpan di dalam penyimpanan *cloud* [27].

Salah satu layanan dari *cloud storage* yang baik untuk digunakan adalah Google *Cloud Storage*. Google *Cloud Storage* merupakan layanan *cloud* yang termasuk dari banyaknya layanan yang ditawarkan oleh penyedia layanan *cloud* yaitu GCP. Penyimpanan *cloud* ini dapat membantu pengembang skala bawah hingga atas seperti di perusahaan untuk melakukan menyimpan dan mengakses

data di Google *Cloud*. Fitur ini dapat membantu para *developer* agar dapat mengakses langsung Google *scalable storage* dan infrastuktur jaringan yang sama baiknya dengan metode autentifikasi dan mekanisme dengan data *sharing* yang kuat [25].

2.6 *Backend Development*

Backend merupakan bagian dari suatu proses yang terjadi dibalik sebuah sebuah *website* atau aplikasi. Bahasa pemprograman untuk pengembangan atau pembuatan dari *backend development* secara umum digunakan diantaranya adalah javascript, PHP, Ruby, Python, dan banyak lainnya. *Backend* merupakan sebuah mesin yang dapat melakukan pekerjaan di balik layar, segala sesuatu yang pengguna akhir atau interaksi langsung tidak lihat dalam prosesnya, namun dengan itu dapat memberikan pada apa yang terjadi. Pengembangan *backend* berfokus pada *database*, *scripting*, dan arsitektur sesuai dengan kebutuhan aplikasi yang dapat memudahkan terciptanya fitur penggunaan aplikasi. Kode yang ditulis oleh pengembang dalam membangun *backend* dari aplikasi akan membantu transfer informasi basis data terhadap *browser* [28].

Backend merupakan fungsi atau bagian yang bertanggung jawab untuk melakukan proses data, menjalankan logika bisnis, dan berkomunikasi dengan basis data. *Backend* berfungsi untuk penghubung antara pengguna dengan penyimpanan data menggunakan antarmuka pengguna atau API. Proses membuatan *backend* dalam menggunakan *serverless platform* seperti Google *Cloud Platform* memungkinkan untuk melakukan *run code* tanpa harus mengelola infrastruktur server.

2.7 *API*

API adalah antarmuka yang sering digunakan dalam mengakses suatu aplikasi atau layanan dari program yang dibuat. API dapat membuat pengembang untuk menggunakan fitur atau fungsi yang sudah ada dari aplikasi lain, sehingga pengembang tidak perlu melakukan membuat ulang dari awal. Penggunan dalam konteks *website*, API merupakan pemanggilan fungsi melalui *Hyper Text Transfer Protocol* (HTTP) yang akan menerima sebuah respon dalam format berbentuk

Extensible Markup Language (XML) atau *JavaScript Object Notation* (JSON) [29].

Tujuan penggunaan dari API dalam pengembangan aplikasi berfungsi agar dapat saling berbagi data antar aplikasi yang berbeda, selain dapat saling berbagi data antar aplikasi API memiliki agar mempercepat proses pengembangan aplikasi dengan cara menyediakan sebuah *function* yang terpisah sehingga pengembang tidak perlu lagi merancang fitur yang sama. API yang bekerja pada tingkat sistem operasi dapat membantu aplikasi untuk bisa berkomunikasi dengan lapisan dasar dan satu sama lainnya dapat mengikuti serangkaian protokol dengan spesifikasi yang telah disesuaikan [29].

2.8 *REST API*

REST API adalah seperangkat prinsip arsitektur yang dapat melakukan transfer data melalui antarmuka yang terstandarisasi seperti HTTP. REST dapat berfungsi seperti layaknya aplikasi *web*. Penggunaan REST API *Client* mampu mengirimkan sebuah permintaan kepada server melalui protokol HTTP yang kemudian *server* memberikan kembali respon kepada klien.

REST dibuat dan dikembangkan oleh Roy Fielding yang merupakan salah satu pendiri dari Apache HTTP *Server Project*. Dalam arsitektur REST *server* mampu menyediakan dan mengakses sumber daya, data REST klien mengakses dan menunjukan sumber daya tersebut untuk penggunaan yang lain. Setiap sumber daya diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. *Resource* tersebut direpresentasikan dalam bentuk format teks, JSON atau XML [29].

2.9 *NodeJS*

Node.js merupakan *platform* yang dibuat untuk sebuah *webserver*. Node.js ini diulis dalam bentuk bahasa pemprograman javascript yang berbasis pada *event driven*. Berbeda dengan bahasa javascript yang dijalankan di klien, Node.js ini dapat berjalan sebagai aplikasi *server*. Node.js mempunyai kinerja dari *memory* yang terbilang efisien dan lebih baik ketika bekerja dalam kondisi beban yang berat. Pengguna yang menggunakan platform tidak perlu khawatir dengan

terjadinya kondisi *deadlock* pada saat menggunkannya karena tidak ada *lock* dalam Node.js. Aplikasi Nodejs ini terdiri dari V8. Mesin dari javascrpit yang dikebangkan oleh google dan beberapa komponen modul bawaan yang terhubung dengan Nodejs.org [30].

Pada platform Node.js terdapat operasi atas yang bersifat *synchronous* dan *asynchronous,* namun Node.js ini memiliki ke unggulannya yang terdapat pada *asynchronousnya* atau bisa juga disebut dengan *non-blocking* I/O yaitu dapat melakukan permintaan bersamaan dalam proses tunggal. Node.js menggunakan pendekatan *event-driven* berbasis *infinite event loop* dalam satu *thread* untuk mengurangi jumlah memori yang digunakan.

2.10 *Prisma ORM*

Object Relational Mapping (ORM) adalah teknik yang bertujuan untuk menghubungkan perbedaan antara paradigma pengembangan aplikasi berorientasi objek dengan pengembangan aplikasi berorientasi dengan paradigma basis data. Prisma merupakan *Object-Relational Mapper* (ORM) yang dirancang dan dibuat untuk dapat mengoptimalkan koneksi antar basis data dengan basis data lainnya [31]. Fitur yang didapatkan dari prisma yaitu untuk menyelaraskan model basis data proyek dengan membuat integrasi data yang ada pada basis data dengan API tidak terlalu rumit. Skema basis data dengan Prisma ditunjukan pada Gambar 2.1.

```
datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

model User {
  id         Int         @id @unique @default(autoincrement())
  email      String      @unique
  username   String      @unique
  password   String
  created_at DateTime    @default(now())
  updated_at DateTime    @updatedAt
  AuthUsers  AuthUsers[]
  Outfit Outfit[]

  @@map("users")
}
```

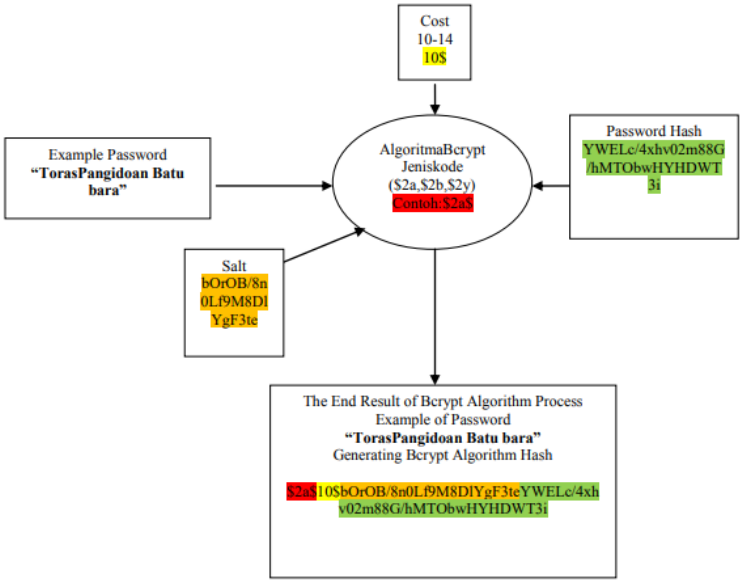
Gambar 2.1 Contoh Skema Tabel Basis Data pada Prisma

Pada Gambar 2.1 merupakan contoh skema pada penggunakan Prisma. Prisma menyediakan skema data yang dapat dibaca dan diubah menjadi definisi yang memudahkan bagi pengembang. Prisma menghasilkan kueri yang menghemat waku karena tanpa perlu menuliskan SQL secara manual dan prisma dapat menangani skema migrasi ketika tidak sesuai dan perlu diubah dalam *database*. Untuk menghubungkan basis data dengan aplikasi, prisma dipasang pada sisi *Backend* yang umumnya dalam menampilkan data para pengembang perlu melakukan penulisan SQL terlebih dahulu, namun dengan Prisma membuat prosesnya lebih mudah karena dapat memungkinkan untuk menunjukan data dalam kode berorientasi objek.

2.11 *Bcrypt*

Bcrypt adalah sebuah fungsi untuk melakukan *hash* yang dikembangkan oleh Niels Provos dan David Mazières berdasarkan enkripsi *Blowfish*. Nama *Bcrypt* terdiri dari B untuk *Blowfish* dan *Crypt*, nama fungsi *hash* yang digunakan dalam sistem kata sandi di UNIX [32]. Bcrypt merupakan sebuah fungsi untuk melakukan *hashing* suatu kata sandi dengan jumlah ilustrasi yang dilakukan lebih banyak untuk membuatnya menjadi lambat dan bertahan lama. Fungsi *hash* yang dilakukan bcrypt ini mampu bertahan terhadap serangan *brute force* dan meningkatkan daya komputasi dengan menggabungkan proses *salt* untuk melindungi dari serangan *rainbow table attack* [33].

Bcrypt ini mengenkripsi 192-bit *hash* dengan menggunakan 128-bit *salt*. Pada prosesnya bcrypt memiliki tahapan inisialisasi dengan secara *default* menggunakan 128-bit pada proses *hashing*. Proses awal bcrypt akan melakukan penurunan kunci, sekumpulan subkunci akan diturunkan dari sebuah kunci utama. Ketika kata sandi yang dimasukkan cukup pendek pada proses awal ini akan dibuat menjadi panjang yang dapat dikatakan melakukan penguatan kunci. Lalu selanjutnya dilakukan enkripsi dengan 192-bit *hash* menggunakan kunci yang telah dibuat dengan dilakukan sebanyak 64 kali. Skema proses *hashing* dengan bcrypt dapat dilihat pada Gambar 2.2.



Gambar 2.2 Proses Keamanan Kata Sandi dengan Bcrypt [33]

Gambar 2.2 merupakan langkah proses melakukan *hashing* pada suatu kata sandi yang dimasukkan, diawali dengan memilih jumlah *cost* yang akan di lakukan proses *hash*. *Default cost* untuk melakukan *hash* sebesar 10. Setelah dilakukan pemilihan nilai *cost* proses *salt* akan melakukan proses acaknya yang akan menghasilkan kode unik pada Gambar 2.2 hasil kode *salt* di *highlight* dengan warna jingga. Kode unik yang melewati proses *hashing* pada Gambar 2.2 diberikan *highlight* dengan warna hijau. Proses *hashing* ini akan menggabungkan kode untuk *salt* dengan kata kunci yang telah dilakukan proses hashing. Hasil akhir dari kode unik yang telah dilakukan proses *hashing* menghasilkan kombinasi karakter yang beragam dan membuat tinggi tingkat keamanan.

2.12 *Jmeter*

Jmeter merupakan alat perangkat lunak untuk melakukan pengujian dari apache berbasis *desktop*, yang ditulis dalam bahasa pemprogramn java. Meskipun JMeter dibuat menggunakan bahasa Java, namun alat ini dapat digunakan untuk menguji aplikasi yang ditulis dengan berbagai bahasa pemprograman. Alat pengujian ini dapat digunakan untuk melakukan pengujian halaman web, aplikasi, dan sumber statis atau dinamis seperti *database*, java object, FTP *server*, dan lain-lain. Jmeter dapat melakukan pengujian beban *load testing* dengan menghasilkan

beberapa parameter seperti dari parameter *quality of service* yaitu metrik yang dapat diukur dengan JMeter adalah TPS (*Transaction Per Second*) atau *throughput*. *Throughput* ini merupakan kecepatan yang di ukur sebagai jumlah permintaan respon dalam waktu tertentu. Latensi rata-rata yang diukur selama pengujian bebas dalam waktu responnya. *Error rate* yang merupakan presentasi permintaan yang gagal dibandingkan dengan total permintaan [34].

JMeter juga dapat mensimulasikan beban *server* yang tinggi dengan membuat beberapa pengguna virtual secara bersamaan di *server web*. Pengujian dapat dilakukan dengan atau tanpa skrip di aplikasi JMeter. Pengujian tanpa skrip dapat dilakukan dengan membuat *Test Plan* yang berisi satu atau lebih *thread*, yang masing-masing bisa terdiri dari beberapa *thread*. Sebuah *thread* adalah simulasi keadaan pengguna yang mengakses langsung aplikasi. Penggunaan Jmeter digunakan untuk melakukan pengujian performa, karena ringan dan mudah dalam penginstalan [35].

2.13 *Kajian Pustaka*

Pada kajian pustaka berisikan rangkuman dari penelitian terdahulu yang berkaitan dengan penelitian yang saat ini dilakukan yaitu mengenai implementasi pembuatan API dengan sistem *cloud storage* untuk aplikasi. Berikut merupakan beberapa penelitian yang menjadi landasan untuk penelitian saat ini.

Penelitian mengenai rancang bangun REST API untuk aplikasi menggunakan perancangan *System Development Life Cycle* (SDLC), dengan melakukan analisis, perancangan, implementasi dan pengujian. hasil dari penelitian ini menjelaskan API dapat membuat proses pengembangan aplikasi menjadi lebih cepat dengan cara membuat sebuah *function* yang terpisah sehingga *developer* perlu merancang fitur yang serupa. Penelitian ini menggunakan sistem *database* berupa ORM (*Object Relation Mapping*), sehingga proses data dapat berhubungan antar tabel [36].

Penelitian mengenai perancangan dan implementasi dari sistem informasi manajemen ini menggunakan metode identifikasi masalah dan melakukan perancangan RESTful API dengan menggunakan bahasa pemprograman java dan basisdata MySQL. Pengujian yang dilakukan pada penelitian ini dilakukan

menggunakan *software* Postman dan melakukan implementasi RESTful API yang di rancang. hasil dari penelitan menunjukan bahwa setiap metode HTTP yang digunakan mampu berjalan dan mengasilkan status berhasil [37].

Penelitian ini membahas terkait penerapan penyimpanan berbasis *cloud* untuk perusahaan hasil dari penelitian ini dengan menerapkan sistem penyimpanan *cloud* dapat memberikan respon positif bagi pengguna terkait efektifitas dan produktifitas pengiriman dan penyimpanan data dibandingkan dengan pertukaran data secara manual dengan media *hardware* [38].

Penelitian mengenai penerapan dan perancangan *backend* API dengan menggunakan *framework* express JS untuk sebuah aplikasi dengan menggunakan metode *waterfall* yaitu dengan melakukan analisa kebutuhan dari aplikasi, desain sistem, implementasi, pengujian, dan penerapan dari *back-end* yang telah dirancang. Penelitian ini menggunakan platform Nodejs dengan framework *Express* yang telah menyediakan fitur *routing*, *handling*, dan *response*. Pada penelitian ini juga memanfaatkan layanan GCP untuk sistem *cloud*. Hasil dari penelitian ini menjelaskan penerapan yang dilakukan membuat keseluruhan API berfungsi dengan baik dan didokumentasi untuk memudahkan dalam mengakses API [39].

Penelitian mengenai perancangan *back-end server* dengan menggunakan arsitektur REST dan platform Nodejs. pada penelitian ini menjelaskan peggunaan Nodejs memberikan keunggulan pada teknik *non-blocking* yang memungkinkan banyak *request* yang dapat diselesaikan secara parallel dengan kinerja dan fungsionalitas yang baik. Penelitian ini menjelaskan layanan REST API yang berjalan dengan Nodejs lebih sesuai untuk mengatasi respon langsung terhadap permintaan yang masuk tanpa melibatkan banyak perhitungan [40].

Penelitian ini melakukan perancangan dan implementasi RESTful API untuk aplikasi *mobile* pembelajaran flora dan fauna dengan menggunakan metode *waterfall* dengan Nodejs dan *framework* Express JS yang dilakukan *deployment* pada layanan *cloud computing Google Cloud Platform* (GCP) yang menghasilkan rancangan berhasil dibangun dan dengan menggunakan layanan dari GCP dapat mendukung performa RESTful API dari aplikasi EksFlorasi [41].

Oleh karena itu, dari hasil penelitian sebelumnya yang menjadi rujukan kajian pustaka untuk penelitian ini, maka pada penelitian ini melakukan implementasi pembuatan API dengan sistem *cloud storage* untuk komponen *backend* pada aplikasi rekomendasi pakaian dengan menggunakan metode analisis kebutuhan aplikasi, perancangan, implementasi, dan pengujian API yang telah dibuat, dengan begitu hasil dari API yang telah dibuat akan berfungsi dengan baik sebagai *backend* untuk aplikasi rekomendasi pakaian.