BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Dalam pembuatan aplikasi ini dilakukan beberapa tinjauan studi yang nantinya mendukung pembuatan aplikasi yang akan dilakukan, dimana tinjauan studi yang dilakukan adalah :

1. Rancang Bangun Aplikasi Mobile untuk Notifikasi Jadwal Kuliah Berbasis Android (Studi Kasus : STMIK Provisi Semarang) oleh Taufik Ramadhan dan Victor G Utomo (2014). Pada aplikasi ini diangkat masalah mahasiswa yang dapat mengakses jadwal kuliah dari kampung halaman dengan handphone.
2. Sistem Informasi Penjadwalan Kegiatan Belajar Mengajar Berbasis Web (Studi Kasus : Yayasan Ganesha Operation Semarang) oleh Rudi Hidayat dan Victor Gayuh Utomo (2016). Pada aplikasi tersebut mengangkat bagaimana penjadwalan dibuat dalam kurun waktu satu hari dengan menggunakan Bahasa Pemgrograman PHP dan database MySQL.
3. Sistem Reservasi Pesawat Terbang Berbasis Web oleh Anton Setiawan Honggowibowo dan Titien Desiarti (2005). Pada aplikasi tersebut diangkat masalah pemrograman dari *client-server* side untuk pembuatan program.

Dari ketiga tinjauan studi yang telah dilakukan, terdapat perbedaan antara tinjauan dengan aplikasi penjadwalan *driver* yang dibuat. Tinjauan studi pertama menggunakan XAML sebagai metode pertukaran data. Tinjauan studi kedua merupakan jadwal per hari dengan MySQL sebagai database yang digunakan. Sedangkan Tinjauan studi ketiga *client-side coding* untuk merender aplikasi. Sementara itu aplikasi karya penulis yakni aplikasi penjadwalan yang dibuat penulis menggunakan JSON sebagai pertukaran data, MongoDB sebagai database, dan serta pengoptimalan *server-side coding*.

2.2 Dasar Teori

2.2.1 Visual Studio Code

Merupakan sebuah perangkat lunak dibawah naungan Microsoft Corporation yang dapat beroperasi di dalam Sistem Operasi Windows, Linux, dan MacOS. Berfungsi sebagai alat yang digunakan untuk mengembangkan aplikasi website maupun console (Studio, 2016) .Visual Studio mendukung proses *debugging*, perintah operasi gabungan Git dengan GitHub, penyorotan sintaksis, otomatisasi dan saran pada *code*, *snippets*, dan *code refactor*. Sumber data yang digunakan bersifat *open source* dan berada dibawah lisensi MIT. Lisensi MIT merupakan lisensi perangkat lunak bebas permisif yang berasal dari Massachusetts Institute of Technology. Sebagai lisensi permisif, Lisensi MIT memberikan batasan yang sangat longgar tentang penggunaan kembali dan memiliki kompatibilitas lisensi yang sangat baik (MIT, 2017). Hasil akhir *binary* merupakan hak bebas pengguna yang bisa digunakan untuk keperluan pribadi maupun komersial.

2.2.2 Arsitektur Aplikasi

Arsitektur aplikasi adalah ilmu dan seni yang digunakan untuk mendeskripsikan perilaku sebuah aplikasi yang digunakan di dalam suatu bisnis. Pada praktiknya, fokus dari ilmu ini terletak pada bagaimana aplikasi dapat berkomunikasi dengan pengguna melalui fungsi yang dibutuhkan. Agar fungsi ini dapat berjalan dengan baik, dibutuhkan komunikasi antar *package* aplikasi, *database*, dan *middleware*. Ilmu ini bertujuan untuk memastikan aplikasi yang digunakan memiliki keterluasan, dapat diandalkan, mampu dan mudah dikelola.

2.2.3 Arsitektur Aplikasi Web

Terdapat dua komponen dasar di dalam arsitektur aplikasi web, yakni tampilan pengguna dan struktur. Tampilan pengguna adalah tempat interaksi antara aplikasi dan pengguna melalui halaman web yang dipicu oleh fungsi seperti *log in*, *searching* dan *upload* foto. Halaman ini dapat dibuat dengan memadukan beberapa bahasa pemrograman seperti *HTML*, *CSS* dan *Javascript* (Osetskyi, 2019). Sementara itu pada struktur terdapat koneksi ke internet *browser* atau pengguna, *server* aplikasi, dan *server database.* Untuk membangun koneksi antar server dan memastikan fungsi yang ditampilkan berjalan dengan baik, digunakan berbagai jenis bahasa pemrograman seperti *Ruby*, *Node.js*, *Pyhton*, *.Net*, *PHP*.

Membangun halaman web yang baik dan tidak membingungkan pengguna berada dalam ranah kerja Frontend sementara memastikan struktur web berjalan dengan benar termasuk dalam lingkup kerja Backend.

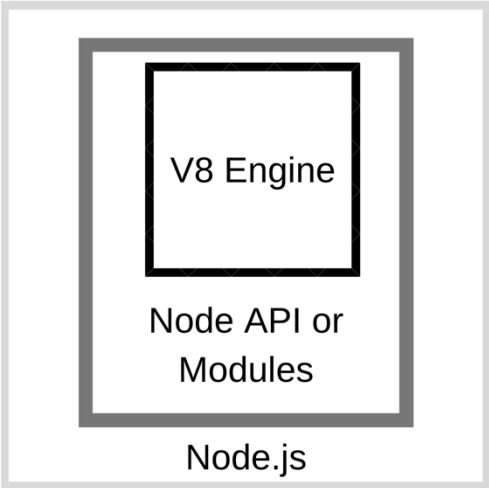
2.2.4 Backend

Merupakan istilah yang sering digunakan dalam aplikasi web bertipe CSM (*Content Management System*) (Medium, 2017). Dalam sebuah halaman web, kita dapat melihat laman tampilannya dan mengakses fungsi yang ada. Backend bekerja dengan cara masuk menggunakan akses administrator untuk memastikan fungsi-fungsi ini berjalan dengan baik, seperti membangun koneksi antar halaman web dan *server database* untuk fungsi *log in.*

2.2.5 Node.js

Node.js adalah sebuah *runtime environment* atau ruang lingkup dimana aplikasi berbasis Javascript berjalan (Org, 2017) .Di dalam

ruang lingkup ini, telah disediakan semua kebutuhan yang diperlukan agar sebuah aplikasi berbasis javascript berjalan dengan baik.



Gambar 2. 1 Ruang Lingkup Node.js

Baik javascript maupun Node.js berjalan di Mesin V8. Mesin inilah yang akan mengubah kode Javascript menjadi kode mesin yang cepat. Kode mesin sendiri adalah kode yang dapat dimengerti oleh komputer tanpa perlu diterjemahkan menjadi lebih sederhana.

Node.js merupakan sistem yang bekerja secara *asyncronous*, yakni mampu menjalankan beberapa proses secara bersamaan. Proses *runtime* yang berjalan secara *asyncronous* ini sengaja dibuat agar mampu mendukung pengembangan aplikasi secara berskala.

2.2.6 Express.js

Express.js adalah sebuah framework aplikasi berbasis web yang menggunakan core pemrograman Node.js dengan komponen modul *Http* dan *Connect* (Express.js, 2016). Framework ini dibuat untuk berjalan dalam mesin *V8 Chrome*, membuatnya berjalan beriringan dengan Node sehingga dapat melakukan kerja dengan sangat cepat.

Framework Express.js menggunakan proses paradigma kerja yang disebut dengan *event-driven programming*. Paradigma ini berpengaruh pada flow sistem yang telah dibuat dengan membuat

semua fungsi berjalan berdasarkan *event*. Semua *event* akan terus diawasi oleh *code* yang disebut dengan *event handler* yang bekerja dengan cara mendeteksi event aktif kemudian memanggil *handler*. Kebanyakan *handler* memiliki tipe sebagai *function* atau *method*. Dalam *handler* ini terjadi proses fungsi yang tengah dipicu *user* (Vivah, 2017).

Seluruh proses tersebut tidak terjadi secara otomatis. Terdapat sebuah proses yang terjadi secara bersamaan sehingga browser akan diharuskan untuk menunggu. Multi proses yang terjadi di server ini dihandle oleh Node dengan menggunakan *Node Event Loop*.

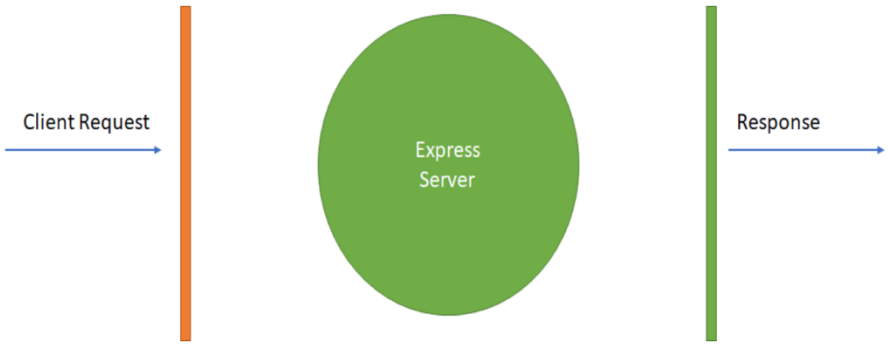
Node menyediakan *Node Event Loop* sebagai bagian dari bahasa pemrograman miliknya. Ketika Node dipanggil, *programmer* tidak perlu membuat *code* yang berfungsi memanggil fungsi lain untuk memulai sebuah *loop*. Express.js akan langsung memulai *loop* otomatis begitu Node berjalan. Proses *loop* ini akan terus menerus berjalan hingga programmer membuat *code* berisi kondisi untuk program agar berhenti. Berikut adalah kategori komponen yang dapat digunakan untuk menjalankan *loop* dalam Express.js:

1. *Controllers*. Mendefinisikan *route handler* aplikasi dan logika bisnis.
2. *Middleware*. Digunakan untuk menginterpretasikan semua permintaan yang datang sebelum diproses oleh *route handler*.
3. *Models*. Merupakan perantara antara *controller* dan database. Kita dapat mendefinisikan sebuah skema database dan melakukan validasi data pada database.
4. *Public*. Tempat menyimpan gambar, javascript dan css file.
5. *Routes*. Mendefinisikan rute aplikasi dengan menggunakan metode http.

6. *Util*. Merupakan fungsi tambahan yang dapat membantu jalannya aplikasi serta dapat digunakan oleh semua *controller*.
7. *Views*. Di dalamnya terdapat templat yang akan diproses oleh server.
8. *app.js*. Merupakan file utama yang akan dijalankan pertama kali ketika aplikasi dinyalakan.
9. *package.json*. Mengelola package yang diinstal lewat NPM, perintah untuk menjalankan aplikasi, serta versi proyek.

2.2.7 Express.js Middleware

Seperti namanya, middleware (alat penengah) bekerja diantara proses *request* dan *response* serta memiliki hak akses untuk kedua proses tersebut. Merupakan salah satu bagian dalam *life-cycle* Express.js. Selain memiliki kemampuan untuk melanjutkan proses program menuju proses yang lain dengan menggunakan next, Middleware juga mampu menghentikan perputaran *request-response*.

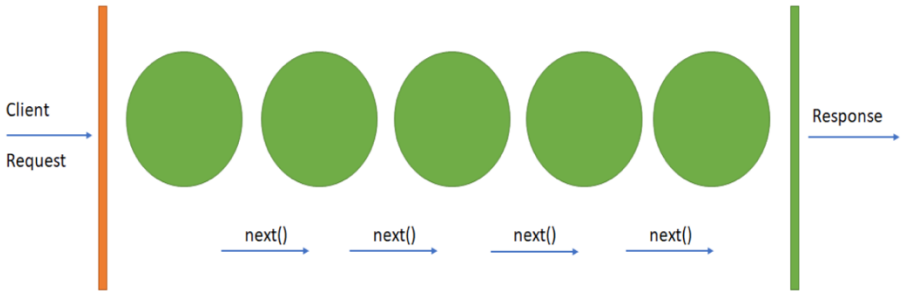


Gambar 2. 2. Express.js Life Cycle

(Sumber : Selvaganesh, 2018)

Middleware bekerja dengan cara menerima obyek *request* dan *response* sebagai *third handler* (tangan ketiga). Mirip seperti yang dapat dilakukan oleh *route handler*. Sebagai argumen ketiga,

middleware memiliki fungsi lain yang akan dipanggil setelah *code* middleware dijalankan. Ini berarti program dapat dijalankan dengan menunggu proses sinkron database atau operasi jaringan selesai sebelum melanjutkan ke langkah berikutnya. Proses ini terjadi seperti berikut :



Gambar 2. 3. Express.js Middleware

(Sumber : Selvaganesh, 2018)

Apabila terdapat beberapa middleware yang harus dilalui, maka middleware yang awal dijalankan harus memanggil fungsi next (*next()*). Jika tidak demikian, request yang tengah dijalankan akan terus berputar tanpa bisa selesai (Selvaganesh, 2018).

Berikut beberapa jenis middleware yang disediakan oleh framework express.js:

- Middleware berlevel aplikasi: *app.use*
- Middleware berlevel router: *app route*
- *Build-in* Middleware: *express.static, express.urlcoded, express.json*
- *Error-handling* Middleware: *app.use (err, req, res, next)*
- Middleware pihak ketiga: *bodyparser, cookieparser*

2.2.8 NPM

NPM (*Node Package Manager*) adalah ekosistem berisi registrasi *package* yang memiliki bentuk banyak package Node.js. NPM dapat diakses secara gratis dan merupakan salah satu jenis ekosistem paling populer di dunia (Patel, 2018). NPM terdiri dari 3 komponen:

- *Website*. Merupakan tempat untuk mencari dan menemukan package, mengatur profil, serta membagikan package yang kita buat.
- *CLI* (*Command Line Interface*). Merupakan cara developer berinteraksi dengan NPM lewat *terminal* atau *command prompt*.
- *Registry*. Merupakan database publik yang berisi Javascript software beserta meta informasi tentangnya.

NPM dapat mengelola ketergantungan paket dari suatu proyek, serta pemasangan program *Javascript* secara global. Ketika digunakan sebagai *manager* paket untuk proyek lokal, NPM dapat menginstal semua paket dengan satu perintah melalui file *package.json*. Pada file *package.json*, dapat ditentukan versi pada setiap paket yang akan dipasang dengan menggunakan skema *semantic versioning*. Hal ini memungkinkan *developer* untuk melakukan pembaharuan paket otomatis dan juga menghindari benturan perubahan yang tidak diinginkan. NPM juga menyediakan alat *version-bumping* untuk memilih paket dengan versi tertentu. Selain *package.json*, NPM juga menyediakan file *package-lock.json* yang bertugas mengunci versi paket yang terpasang pada proyek (NPM, 2018).

2.2.9 Path

Path merupakan salah satu paket berisi tool yang dapat diperoleh dari NPM. Path sendiri berfungsi dalam memudahkan proses

routing dalam pemrograman berbasis Javascript. *Route* atau *routing* dalam *backend* diartikan sebagai sebuah jalur yang ditempuh antar paket data agar dapat diproses oleh service yang sesuai.

Semua direktori NPM tersimpan di dalam *node_modules/.bin* kemudian dibagi kedalam *folder-folder* utama paket. Hal ini didesain dengan tujuan untuk memungkinan programmer untuk memanggil fungsi yang dapat dieksekusi dari setiap paket. Misalkan dalam *folder* tersebut terdapat paket Moment, Path akan memungkinkan program untuk mengakses fungsi Moment setelah mendaftarkan direktori Moment ke dalam npm-path melalui $PATH.

Ketika semua direktori paket telah didaftarkan, Path memiliki kemampuan untuk mengeksekusi perintah *script* dengan isi *echo $PATH*. Akan tetapi semua bobot kerja Path tetap ditanggung dan bergantung pada operasi NPM.

2.2.10 Moment

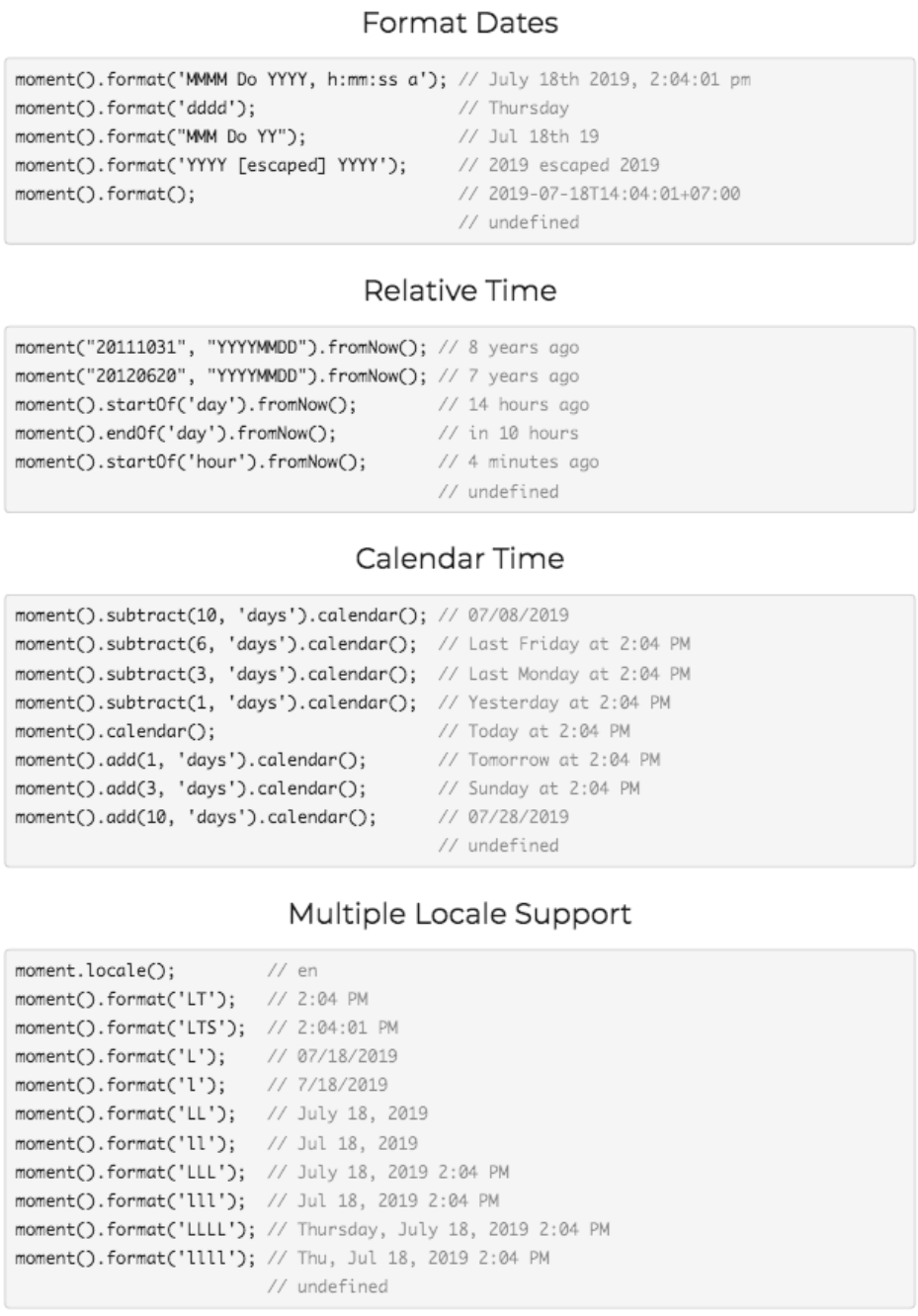
Moment digunakan ketika backend membutuhkan informasi tentang waktu dan tanggal. Apabila kita menggunakan fungsi untuk mendapatkan waktu saat ini menggunakan Vanilla Javascript, hasil yang keluar akan seperti gambar dibawah.

Gambar 2. 4. Hasil Pemanggilan Hari Menggunakan Javascript

Namun terkadang, hasil yang ingin didapatkan dari fungsi ini berbeda-beda. Untuk mendapatkan hasil yang spesifik, *engineer* perlu melakukan manipulasi pada data yang diberikan. Proses panjang dalam

manipulasi data berupa tanggal ini dapat dilakukan dengan lebih cepat dan sederhana menggunakan Moment.

Dengan menggunakan Moment, kita bisa memilih format output yang diinginkan. Beberapa contoh format yang bisa didapatkan dengan Moment adalah seperti gambar berikut :



Gambar 2. 5. Hasil Pemanggilan Hari Menggunakan Moment

(Sumber : Moment.js, 2016)

2.2.11 MongoDB

MongoDB adalah sebuah aplikasi terbuka *NoSQL-Database* yang dikembangkan oleh Korporasi MongoDB dengan menggunakan bahasa pemrograman *C++* (B, 2018). Merupakan jenis baru dari *database* yang tidak terpaku pada bentuk *schema* atau juga yang disebut dengan *non-relational database*. Karena sifat fleksibel yang dimilikinya, setiap data yang ada di dalam koleksi dapat mempunyai tipe objek yang berbeda dari satu sama lain. Setiap objek atau yang juga disebut sebagai *document* selalu diwakili oleh struktur JSON.

Gambar 2. 6. Contoh JSON

(Sumber : JSON, 2009)

Dalam menggunakan MongoDB terdapat beberapa terminologi yang digunakan. Berikut adalah beberapa terminologi yang paling umum digunakan (Karnik, 2018):

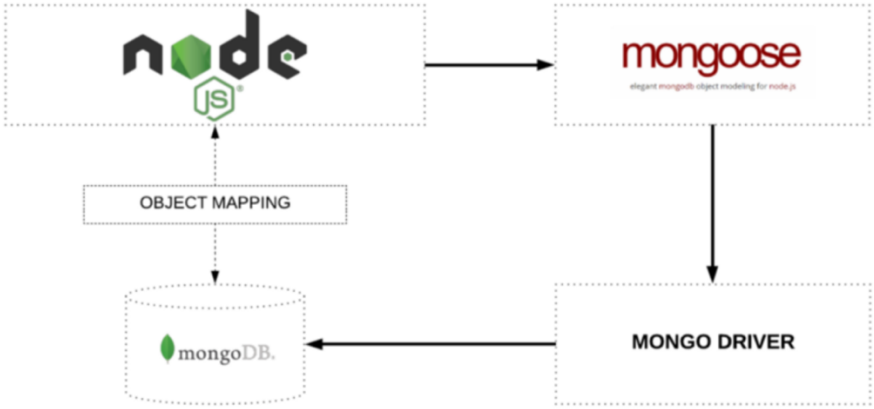
- *Collections*. Pada MongoDB koleksi memiliki posisi yang serupa dengan tabel pada *relational database* seperti SQL. Collection berisikan banyak data JSON.

- *Documents*. Serupa dengan apa yang disebut dengan rekaman atau baris (*row*) pada database SQL. Sementara baris pada SQL dapat mereferensikan data dalam tabel lain, dokumen pada Mongo biasanya menggabungkan data menjadi satu dokumen.
- *Fields*. Atribut fields ini serupa dengan kolom (*column*) pada SQL.
- *Schema*. Sekalipun MongoDB merupakan database yang tidak memiliki skema, SQL mampu mendefinisikan sebuah skema berdasarkan definisi dari tabel. Skema dalam MongoDB adalah struktur data dokumen atau bentuk dokumen yang ditegakkan menggunakan *application layer*.

Models. Model adalah konstruktor tingkat tinggi yang mengambil skema dan membuat *instance* dokumen yang setara dengan catatan dalam database relasional (*relational database*, SQL).

2.2.12 Mongoose

Mongoose merupakan Object Data Modeling (ODM) library yang digunakan untuk MongoDB dan Node.js. Mongoose mengatur hubungan antar data, menyediakan *schema* untuk *validation*, juga digunakan untuk penerjemah antar objek pada code dan representasinya dalam MongoDB.



Gambar 2. 7. Pemetaan Objek Mongo Menggunakan Mongoose

(Sumber : Nick Karnik, 2018)

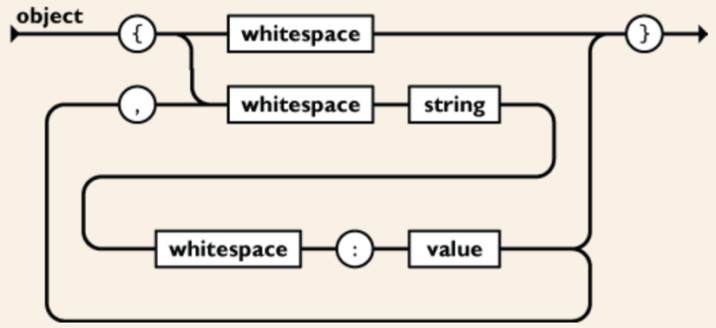
2.2.13 JSON

Ketika server berkomunikasi dengan browser, tipe data yang dapat dikirim hanya berupa teks. *JavaScript Object Notation* atau JSON adalah sebuah format pertukaran data yang ringan (JSON, n.d.). Dibuat berdasarkan bagian dari Bahasa Pemrograman *Javascript*, dengan standar *ECMA-262* edisi ke 3 pada bulan Desember 1999. JSON adalah format teks dan juga bahasa yang sepenuhnya independen. JSON dibuat dengan sintaksis konversi yang mudah dipahami bagi *programmer C-family*, C++, C#, Java, JavaScript, Perl, Python, dan lain sebagainya serta memiliki dua struktur:

- Kumpulan nama atau nilai. Dalam berbagai bahasa, hal ini diwujudkan sebagai objek, catatan, *struck*, kamus, tabel *hash*, daftar kunci, atau *array* asosiatif.
- Daftar urutan nilai atau *value*. Pada sebagian besar bahasa, hal ini diwujudkan sebagai *array*, *vector*, *list*, atau urutan.

Yang demikian adalah struktur data universal. Hampir semua bahasa pemrograman mendukung bentuk tersebut dengan jenis dan cara yang beragam. Karena itu masuk akal apabila pertukaran data dengan bahasa pemrograman juga berdasarkan pada struktur ini.

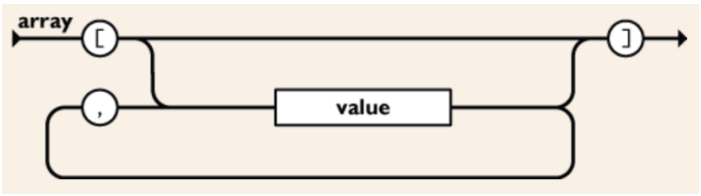
Sebuah objek dalam JSON merupakan kumpulan nama atau nilai yang tidak beraturan. Sebuah objek diawali dengan tanda buka kurung kurawal ({) dan berakhir dengan tanda tutup kurung kurawal (}). Setiap nama diikuti oleh tanda titik dua (:) dan sepasang nama atau nilai diakhiri dengan tanda koma (,).



Gambar 2. 8. Objek dalam JSON

(Sumber : JSON, 2009)

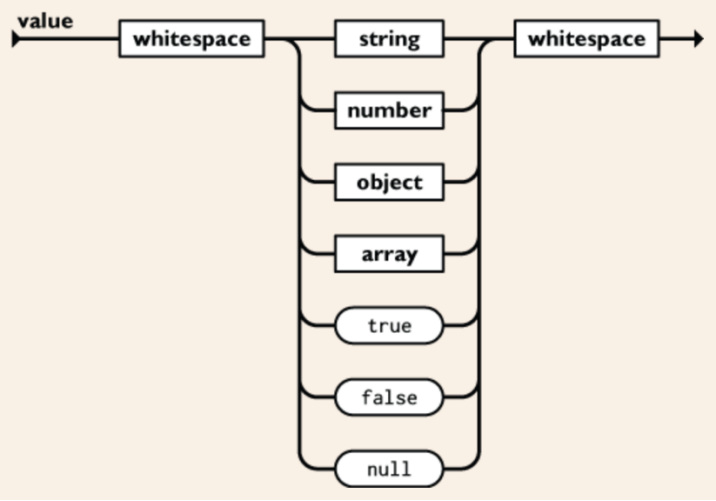
Array dalam JSON merupakan kumpulan dari banyak nilai. Sebuah *array* dimulai dengan tanda buka kurung ([ ) dan diakhiri dengan tanda tutup kurung ( ]). Sementara itu untuk setiap nilai dipisahkan dengan tanda koma (,).



Gambar 2. 9. Array dalam JSON

(Sumber : JSON, 2009)

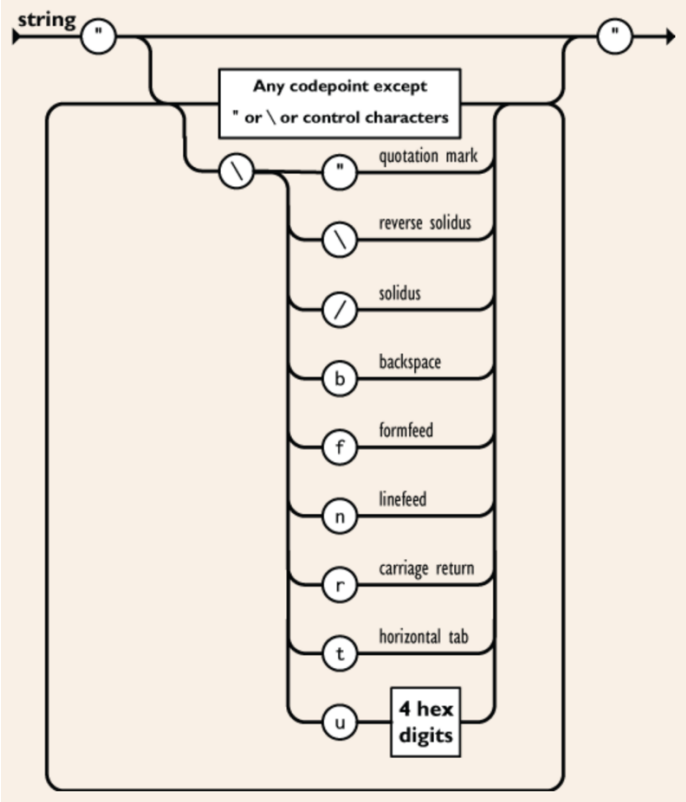
Sebuah nilai atau *value* dapat bertipe *string* yang diapit tanda kutip ganda ("…"), atau angka, atau true/false/null, atau sebuah objek atau *array*. Struktur ini dapat dibuat dengan komponen-komponen yang bercabang atau disebut *nested*.



Gambar 2. 10. Value dalam JSON

(Sumber : JSON, 2009)

String atau urutan dari nol atau lebih karakter *Unicode*, diapit tanda kutip ganda ("…") dan menggunakan *backslash escape*. Sebuah *string* singkat dapat mewakili string yang panjang. Konsep *string* ini sangat mirip dengan konsep *string* pada bahasa pemrograman C dan Java.



Gambar 2. 11. String Dalam JSON

(Sumber : JSON, 2009)

2.2.14 Git

Git adalah *version control system* ciptaan Linus Torvalds yang digunakan para developer untuk mengembangkan software secara bersama-bersama (Git, 2017). Pada awalnya Git hanya dirancang sebagai mesin tingkat rendah yang dapat digunakan oleh tampilan muka (*front end*) lain seperti Cogito atau StGit namun selanjutnya proyek inti Git telah berkembang menjadi pengontrol revisi lengkap yang dapat digunakan langsung. Fungsi utama git yaitu mengatur versi dari source code program dengan mengasih tanda baris dan code mana yang ditambah atau diganti.

Git memudahkan developer untuk mengetahui perubahan source code yang terjadi. Selain itu, git menyediakan fitur branch sebagai workspace developer yang mencegah code bentrok. Fitur lain

yang diunggulkan adalah bagian komentar pada source code yang telah ditambah atau diubah. Hal ini mempermudah developer lain untuk tahu kendala apa yang dialami developer lain.

Untuk mengetahui bagaimana menggunakan git, berikut perintah-perintah dasar git:

- *Git init*. Digunakan untuk membuat *repository* pada file lokal yang nantinya ada *folder* .git
- *Git status*. Digunakan untuk mengetahui status dari *repository* lokal
- *Git add*. Digunakan menambahkan file baru pada *repository* yang dipilih
- *Git commit.* Digunakan untuk menyimpan perubahan yang dilakukan, tetapi tidak ada perubahan pada *remote repository.*
- *Git push*. Digunakan untuk mengirimkan perubahan file setelah di commit ke *remote repository.*
- *Git branch*. Digunakan melihat seluruh *branch* yang ada pada repository
- *Git checkout*. Digunakan menukar *branch* yang aktif dengan *branch* yang dipilih
- *Git merg*e. Digunakan untuk menggabungkan *branch* yang aktif dan *branch* yang dipilih
- *Git clone*. Digunakan membuat salinan *repository* lokal

Gambar 2. 12. Logo Git

(Sumber : Git, 2017)

2.2.15 Mercurial

Mercurial adalah revisi sistem terdistribusi, sebuah alat untuk pengembangan aplikasi. Alat ini dapat digunakan dengan mesin yang menggunakan Sistem Operasi Microsoft Windows serta Unix-like 22ystem seperti FreeBSD, macOS, dan Linux. Tujuan utama Mercurial adalah mendukung performa tinggi aplikasi, menjaga keutuhan teks dan file biner, melakukan *branching* dan *merging* namun tetap mempertahankan konsep kerja yang sederhana (Wikidata, 2017).

Dalam alat ini terdapat sistem integrasi web-interface yang memudahkan developer mengecek perubahan seperti version-control. Penggunaan Mercurial kebanyakan berada pada program yang bekerja di *command-line*. Semua operasi Mercurial dipanggil dalam bentuk argumen kepada program yang dituju seperti *hg* (nama yang diambil merupakan simbol kimia untuk elemen merkuri).

Gambar 2. 13. Logo Mercurial

(Sumber : Wikidata Mercurial, 2017)

2.2.16 Bitbucket

Dalam pembuatan aplikasi yang membutuhkan sebuah tim untuk berkolaborasi, solusi termudah yang paling sering diterapkan yakni menggunakan repositori penyimpanan *code online*. Bitbucket adalah sebuah layanan *hosting* yang berbasis web untuk berbagi *source code* dan pembangunan proyek yang menggunakan Mercurial (sejak peluncuran) ataupun Sistem Pengontrol Versi Git (sejak Oktober 2011) yang dimiliki oleh Atlassian. Akun yang disediakan terbagi menjadi akun gratis dan berbayar, dimana terdapat perbedaan dalam ketersediaan fitur (Bitbucket, 2019).

Bitbucket bekerja secara tradisional disesuaikan untuk membantu pengembang profesional dengan kode pribadi mereka. Aplikasi ini bekerja berdasarkan git dan mendukung penggunaan git sepenuhnya dengan tampilan yang lebih menarik dan integrasi yang kokoh.

Gambar 2. 14. Logo Bitbucket

(Sumber : Bitbucket, 2019)

2.2.17 Insomnia

Insomnia adalah aplikasi multi-platform yang digunakan untuk menata, menjalankan, serta melakukan *debugging* HTTP *request* (Domrongchai, 2017). Bekerja dengan memposisikan mesin lokal sebagai *client* yang mengakses *endpoint* dari *Service* untuk mengetes *server-side-program*. Insomnia menyediakan berbagai manfaat untuk

developer *backend* dan juga *frontend*. Fitur yang dapat digunakan di antara lain:

- Versi gratis untuk *users*. Versi ini dapat diupgrade menjadi berbayar untuk mendapatkan fitur yang lebih banyak.
- Memungkinkan untuk memiliki beberapa *workspace* sekaligus.
- Kemampuan untuk mengatur environment dan variabel lokal statis pada setiap panggilan endpoint (terkadang disebut sebagai *request chaining*).