2. LANDASAN TEORI

2.1. Web Application

Sebuah *web application* adalah aplikasi apapun yang menggunakan *browser* sebagai klien-nya. Dalam *web application*, yang disebut sebagai klien adalah aplikasi yang dipakai untuk memasukkan informasi, dan server adalah aplikasi yang dipakai untuk menyimpan informasi. Membangun aplikasi dengan konsep *web application* menghilangkan kekhawatiran pengembang akan aplikasi yang hanya bisa berjalan pada perangkat atau sistem operasi tertentu (Nations, n.d.).

Web application yang juga bisa disebut sebagai *rich internet application*, memiliki fungsi yang lebih dari sekedar menampilkan konten kepada pengguna. *Web application* dikembangkan dengan maksud untuk memungkinkan interaksi pengguna, dan bukan hanya menampilkan konten (Sharp, 2013).

Dalam halnya dengan pengembangan aplikasi *mobile*, menggunakan *web application* untuk membuat aplikasi *cross-platform* memiliki berbagai manfaat, seperti yang dijelaskan oleh Heitkotter, Hanschke, dan Majchrzak (2013) :

- Tidak ada biaya yang harus dikeluarkan, karena tersedianya banyak *framework web application* yang dapat di-*download* dengan gratis.
- Semua sistem operasi *mobile* dapat membuka *web application*, hanya dengan syarat adanya *browser* yang dapat dipakai untuk membuka internet.
- Tidak perlu melakukan instalasi aplikasi pada perangkat yang ingin membuka *web application*, sehingga tidak perlu menggunakan ruang yang ada pada perangkat untuk menampung aplikasi yang bersangkutan.
- Perkembangan teknologi bahasa pemrograman web mengalami perkembangan terus-menerus dan internet akan selalu menjadi landasan bagi perkembangan teknologi masa depan, sehingga melakukan pengembangan aplikasi dengan *web application* bersifat *future-proof*.

- Pengguna hanya perlu mengetahui alamat dari *web application* dan pengguna akan selalu dapat melihat perkembangan terbaru dari *web application* tersebut. Tidak perlu dilakukan *update* aplikasi secara rutin dari pusat aplikasi seperti Google Play Store, Apple App Store, dan sebagainya.
- Ada banyak tool yang dapat digunakan untuk membangun dan mengubah *web application*. Dari yang sederhana seperti Notepad hingga yang menawarkan kemampuan lebih seperti Adobe Dreamweaver. Juga tersedia tool yang dapat digunakan untuk menguji *web application*. Secara garis besar, *tool* untuk melakukan pengembangan *web application* sudah sangat memadai.
- Dokumentasi-dokumentasi yang diperlukan untuk membangun *web application* tersedia banyak di internet dengan kualitas yang baik, sehingga memudahkan pengembang untuk melakukan pengembangan *web application*.
- Pengembangan lanjutan terbuka lebar bagi pengembangan *web application*. Sebuah proyek yang dimulai dengan *web application* dapat di-*porting* dengan mudah dengan memakai PhoneGap bila akses API *native* dari sistem operasi tertentu diperlukan. Pengembangan yang lebih jauh lagi dapat dilakukan dengan Appcelerator Titanium, bahkan membuat sebuah aplikasi *native*, yang semuanya dimulai dengan *web application*.

2.2. Framework Javascript

Framework Javascript adalah sebuah cara yang lebih baik untuk dengan cepat membangun *web application* yang interaktif, khususnya bila dibandingkan dengan penggunaan Javascript biasa. Javascript biasa bersama dengan jQuery telah dipakai selama bertahun-tahun untuk membangun antarmuka web yang kompleks, tetapi dengan jauh lebih banyak upaya dan kompleksitas dalam pembangunan dan pemeliharaannya. Pemakaian *framework* Javascript memberikan ruang bagi pengembang untuk fokus pada pembangunan elemen-elemen interaktif pada

antarmuka web tanpa terlalu mengkhawatirkan struktur kode dan pemeliharaan kode (Arora, 2016).

2.2.1. AngularJS

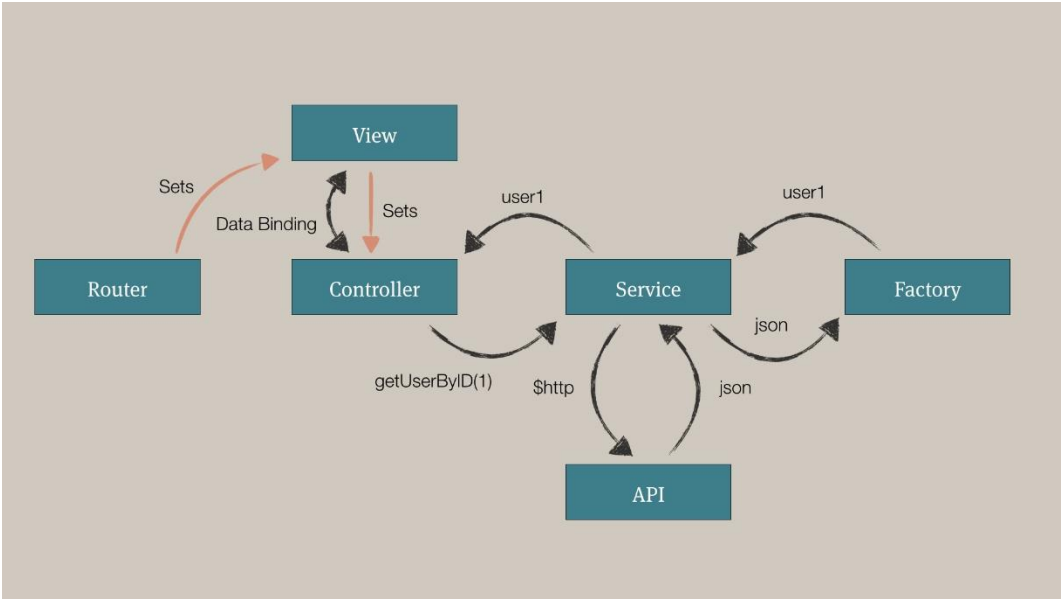
AngularJS adalah sebuah *framework* struktural yang ditujukan untuk pengembangan *web application* yang dinamis. AngularJS memungkinkan penggunaan *HTML* sebagai bahasa *template* dan memperluas *syntax HTML* untuk mengekspresikan komponen aplikasi yang dibangun dengan jelas dan ringkas. AngularJS berupaya untuk meminimalisir ketidakcocokan antara *HTML* yang dokumen sentris dengan kebutuhan dalam pengembangan aplikasi dengan membuat susunan *HTML* yang baru. AngularJS menangani semua hubungan antara *DOM* dan *AJAX* dengan menyusunnya sebagai struktur yang didefinisikan dengan baik. AngularJS juga menyederhanakan pengembangan aplikasi dengan menyediakan tingkat abstraksi yang lebih tinggi bagi pengembang (Google, 2015).

Ada beberapa konsep dasar yang dipakai dalam pengembangan aplikasi AngularJS seperti yang ditulis oleh Green dan Seshadri (2013), antara lain adalah :

- *Client-Side Templates* : *template* dan data dikirimkan dari server menuju *browser* untuk kemudian disatukan pada *browser* sebelum ditunjukkan pada pengguna.
- *Model View Controller* : adanya pembedaan antara pengelolaan data (*model*), logika aplikasi (*controller*), dan presentasi data untuk pengguna (*view*). *View* mendapat data dari *model* untuk ditampilkan pada pengguna. Ketika pengguna berinteraksi dengan aplikasi, *controller* meresponi dengan mengubah data dalam *model*. *Model* memberitahu *view* bahwa terjadi perubahan data sehingga *view* memperbarui apa yang ditampilkan.
- *Data Binding* : sinkronisasi otomatis antara *model* dan *view*.
- *Dependency Injection* : mengarahkan pengembang untuk mengikuti gaya pengembangan di mana kelas-kelas yang terdapat pada aplikasi hanya meminta apa yang mereka perlukan, sehingga membantu pengembang untuk membuat, mengerti, dan menguji aplikasi dengan mudah.
- *Directives* : membantu menentukan *view* pada aplikasi dan dapat dipakai untuk membuat *HTML custom tag* yang baru.

Buku “AngularJS: Web Application Framework” yang diterbitkan oleh Tutorials Point pada tahun 2014, mencatat beberapa keunggulan dari AngularJS, yaitu :

- AngularJS memberi kemampuan untuk membuat *single page application* dengan cara yang bersih dan mudah dipelihara
- AngularJS memberi kemampuan *data binding* bagi *HTML*, sehingga memberi pengalaman yang kaya dan responsif bagi pengguna
- Kode AngularJS dapat digunakan untuk *unit testing*
- AngularJS memberi komponen-komponen yang *reusable*
- Pengembang dapat mencapai fungsionalitas yang lebih baik dengan kode yang lebih sedikit
- *View* pada AngularJS adalah murni *HTML*, sementara *controller* yang ada pada JavaScript melakukan proses kerja yang diperlukan
- AngularJS didukung dan dikembangkan oleh Google
- Besarnya komunitas yang mendukung AngularJS, dengan banyaknya *plugin* dan *tutorial* yang siap mendukung pengembang ketika membuat *web application*



Gambar 2.1. Cara Kerja AngularJS

(Sumber: https://www.mutuallyhuman.com/blog/2014/05/08/angularjs-services-and-factories-done-right/)

2.3. MySQL

MySQL adalah sistem manajemen *database* SQL bersifat *open source* yang paling terkenal saat ini. MySQL dikembangkan, didistribusikan, dan didukung oleh Oracle Corporation. MySQL adalah sebuah sistem manajemen *database*, yang berfungsi untuk menambah, mengakses, dan memproses data yang tersimpan dalam *database* (*database* adalah koleksi data yang terstruktur). *Database* dalam MySQL bersifat relasional. *Database* relasional menyimpan data dalam tabel-tabel yang terpisah, dan model logikal dari *database* menawarkan lingkungan pemrograman yang fleksibel, di mana pengguna dapat menentukan relasi antar tabel. MySQL bersifat *open source*, sehingga memungkinkan siapa saja untuk memakai dan memodifikasi MySQL. MySQL Server dapat berjalan dalam sistem *client/server* atau sistem tertanam, yang berarti MySQL mendukung berbagai *back-end,* berbagai program dan *library*, berbagai *tool* administrasi, dan berbagai *application programming interfaces* (Oracle, 2015).

MySQL banyak digunakan pada berbagai server web. MySQL adalah sebuah sistem manajemen *database* yang kuat dan luar biasa cepat yang menggunakan perintah-perintah berbahasa Inggris. Tingkat tertinggi dari struktur MySQL adalah *database*, yang di dalamnya pengguna dapat memiliki satu atau lebih tabel yang menyimpan data pengguna. Perintah-perintah dasar yang dapat dieksekusi dalam MySQL adalah SELECT untuk mencari data dan INSERT untuk memasukkan data, akan tetapi ada beberapa fungsi lain yang dapat dipakai pengguna dibanding SELECT dan INSERT yang sederhana, antara lain adalah penggabungan beberapa tabel menurut berbagai kriteria, meminta hasil dengan pengurutan yang beragam, dan lain-lain. Penggunaan PHP memungkinkan pengguna untuk memanggil data secara langsung tanpa perlu menjalankan MySQL secara manual atau memakai *command line interface* (Nixon, 2014).

Berikut ini adalah tipe-tipe data yang didukung oleh MySQL, sesuai dengan yang dituliskan oleh Oracle (2015) dalam "MySQL 5.6 Reference Manual" :

- Numerik
    - Integer: tipe data Integer menyimpan angka *integer* dalam nilai yang eksak. Tipe data Integer masih dibagi menjadi Tinyint, Smallint, Mediumint, Int/Integer, dan Bigint, dimana setiap tipe data ini

memiliki cakupan angka yang berbeda-beda, dari yang terkecil (Tinyint) hingga yang terbesar (Bigint).

- Decimal, Numeric: dalam MySQL, Numeric diimplementasikan sebagai Decimal, sehingga sebenarnya tidak ada perbedaan antara kedua tipe data ini. Decimal (maupun Numeric) menyimpan angka *fixed-point* dalam nilai yang eksak. Tipe data ini penting dalam penggunaannya untuk menyimpan angka dengan presisi yang tepat, seperti data keuangan. Tipe data ini ditulis dalam format "DECIMAL (x,y)" di mana 'x' menandakan jumlah *digit* yang dapat disimpan dan 'y' berarti jumlah *digit* desimal yang dapat disimpan.
- Float, Double: Float dan Double mewakili nilai data numerik yang berjumlah kira-kira (*floating-point*). Float dipakai untuk menyimpan nilai dengan presisi tunggal, sementara Double dipakai untuk menyimpan nilai dengan presisi ganda. Double berkapasitas dua kali lipat dibanding Float, sehingga memungkinkan akurasi yang lebih tinggi.
- Bit: tipe data Bit dipakai untuk menyimpan nilai bit.

- Tanggal dan waktu
  - Date, Datetime, Timestamp: Date, Datetime, dan Timestamp adalah tipe-tipe data yang saling berhubungan. Walau begitu, ada perbedaan di antara ketiga tipe data ini. Date dipakai untuk menyimpan nilai yang berupa tanggal, namun tidak menyimpan waktu. Date ditulis dalam format 'YYYY-MM-DD'. Datetime dipakai untuk menyimpan nilai yang mengandung tanggal dan waktu. Datetime dapat ditulis dalam format 'YYYY-MM-DD HH:MM:SS'. Timestamp juga dipakai untuk menyimpan nilai yang mengandung tanggal dan waktu dan ditulis dengan format yang sama dengan Datetime ditambah 'UTC'. Perbedaan Timestamp dengan Datetime adalah Timestamp menyimpan waktu dalam UTC.
  - Time: Time menyimpan nilai yang mengandung waktu dan ditulis dalam format 'HH:MM:SS'.

- Year: Year dipakai untuk menyimpan nilai yang hanya berupa tahun saja.

- String
  - Char, Varchar: Char dan Varchar adalah dua tipe data yang mirip, tetapi memiliki perbedaan dalam proses penyimpanan dan pengambilan datanya. Panjang kolom Char bersifat tetap sejak pengguna memasukkan panjang kolom tersebut, dan dapat memiliki panjang bernilai 0 hingga 255. Sedangkan nilai dalam kolom Varchar disesuaikan dengan panjang variabel dan dapat memiliki panjang bernilai 0 hingga 65535. Ketika pengguna memasukkan panjang kolom Varchar, panjang kolom yang dimasukkan tersebut hanya dianggap sebagai nilai maksimal kolom itu, bukan nilai tetap kolom. Tabel 2.1. akan menjelaskan perbedaan Char dan Varchar.

Tabel 2.1 Perbedaan Char dan Varchar

Value	CHAR(4)	*Storage Required*	VARCHAR(4)	*Storage Required*
‘’	‘ ‘	4 bytes	‘’	1 byte
‘ab’	‘ab ‘	4 bytes	‘ab’	3 bytes
‘abcd’	‘abcd’	4 bytes	‘abcd’	5 bytes
‘abcdefgh’	‘abcd’	4 bytes	‘abcd’	5 bytes

Sumber: Oracle. (2015, p.1286)

  - Binary, Varbinary: Binary dan Varbinary mirip halnya dengan Char dan Varchar, kecuali fakta bahwa Binary dan Varbinary menyimpan *string* biner. Karena itu, kedua tipe data ini mengandung *string* dalam bentuk *byte* dibanding *string* karakter.
  - Blob, Text: Blob dan Text berbeda dalam hal bagaimana kedua tipe data ini ditangani. Blob ditangani sebagai *string* biner yang tidak memiliki set karakter. Text ditangani sebagai *string* nonbiner yang memiliki set karakter. Blob dapat dibagi menjadi Tinyblob, Blob, Mediumblob, dan Longblob dan hanya berbeda dalam jumlah maksimum nilai yang dapat ditampung. Text juga dapat dibagi menjadi Tinytext, Text, Mediumtext, dan Longtext.

- Enum: Enum adalah obyek string dengan nilai yang dapat dipilih dari sebuah daftar nilai-nilai yang diizinkan yang disebutkan secara eksplisit dalam spesifikasi kolom pada saat pembuatan tabel.
- Set: Set adalah obyek string yang dapat memiliki nilai nol atau lebih, di mana setiap dari nilai tersebut harus dipilih dari sebuah daftar nilai-nilai yang diizinkan yang disebutkan pada saat pembuatan tabel. Set memiliki nilai yang terdiri atas serangkaian anggota yang dipisahkan dengan koma.

2.4. Framework PHP

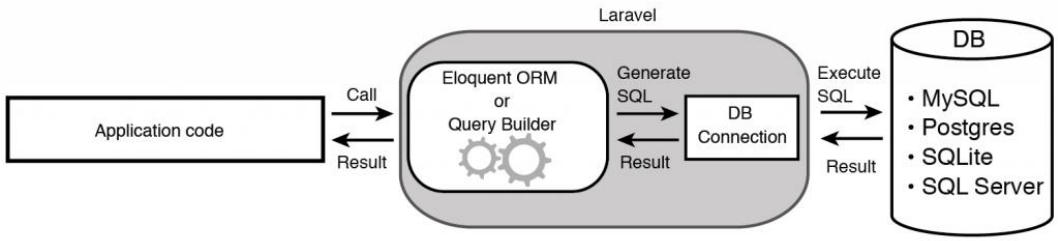
PHP adalah bahasa scripting skrip paling populer di dunia karena banyak alasan yang berbeda-beda, namun sering kali melakukan pemrograman dalam PHP dapat menjadi monoton dan berulang-ulang. *Framework* PHP mempersingkat pembangunan *web application* dengan menyediakan struktur dasar untuk membangun *web application*, yang dengan begitu dapat membantu pengembang dalam menghemat waktu, membangun aplikasi yang lebih stabil, dan mengurangi jumlah pemrograman yang berulang (Noupe Editorial Team, 2009).

Penggunaan *framework* PHP adalah sesuatu yang populer saat ini, karena *framework* sangat mengurangi jumlah pekerjaan yang perlu dilakukan pengembang, dengan menangani banyak keputusan-keputusan umum bagi pengembang, sebuah konsep yang dikenal sebagai '*convention over configuration*' (Gilmore, 2015).

2.4.1. Laravel

Laravel adalah sebuah *framework* PHP yang ditulis Taylor Otwell, dengan tujuan untuk membantu pengembang dalam membuat *web application* dengan *syntax* yang sederhana, elegan, ekspresif, dan menyenangkan. Pembuatan *web application* dengan Laravel dapat menghasilkan *web application* dengan *syntax* yang ekspresif dan elegan. "Dengan Laravel, tugas-tugas umum developer dapat dikurangi pada sebagian besar proyek-proyek web seperti *routing*, *sessio*n dan *caching*." (Nugraha, n.d.)

Laravel sebagai *framework* berbasis PHP umumnya digunakan pengembang bersama dengan sebuah *database* seperti MySQL atau PostgreSQL. Karena itu, sebelum melakukan pengembangan *web application* dengan Laravel, pengembang perlu melakukan instalasi PHP dengan versi 5.4 atau lebih baru dan *database* yang didukung oleh Laravel (MySQL, PostgreSQL, SQLite, dan Microsoft SQL Server). Jika pengembang ingin melakukan pengembangan tanpa melakukan instalasi PHP, pengembang juga dapat melakukan instalasi Laravel Homestead, sebuah *virtual machine* berbasis Vagrant yang membundel semua hal yang diperlukan untuk memulai pengembangan situs web dengan Laravel (Gilmore, 2015).



Gambar 2.2. Laravel dan *Database*
(Sumber: http://maxoffsky.com/code-blog/laravel-first-framework-chapter-6-database-operations/)

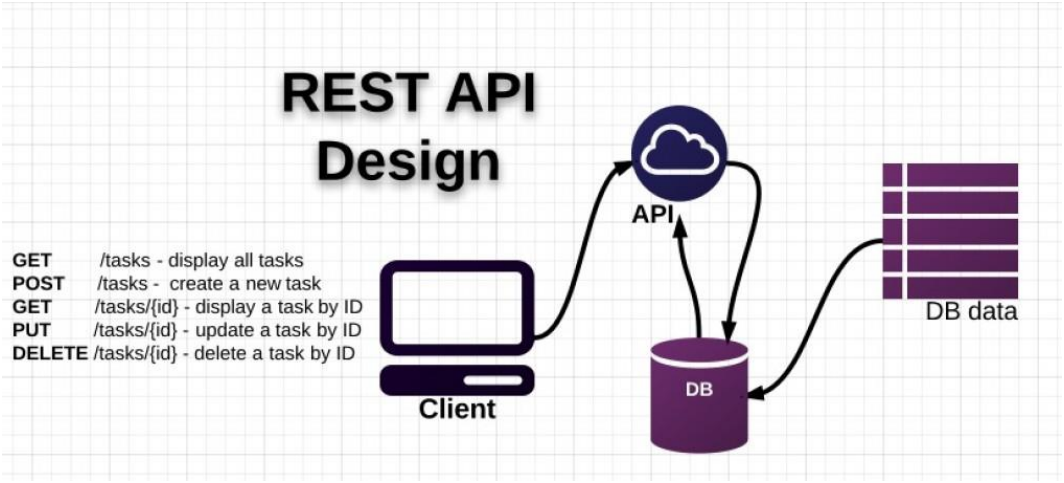
2.5. REST

REST (*REpresentational State Transfer*) adalah suatu gaya arsitektur dan suatu pendekatan untuk melakukan komunikasi yang sering dipakai dalam pengembangan *web service*. Pemakaian *REST* lebih sering dipilih dibandingkan *SOAP* (*Simple Object Access Protocol*) karena *REST* tidak membutuhkan terlalu banyak *bandwidth* seperti halnya *SOAP*. Arsitektur yang bersifat terpisah dari *REST*, disertai pemakaian data komunikasi yang lebih ringan antara produsen dan konsumen membuat *REST* menjadi gaya pengembangan yang populer dalam membangun *API* berbasis *cloud*. *Web service* yang memakai arsitektur *REST* disebut sebagai *RESTful API* atau *REST API* (Rouse, 2014).

Menurut Fielding (2000), ada enam batasan yang dideskripsikan oleh gaya arsitektural *REST*. Enam batasan ini berlaku bagi arsitektur *REST* dan menjadi basis bagi gaya *RESTful* (dalam Fredrich, 2012, p. 6). Enam batasan tersebut adalah :

- *Uniform Interface* : Batasan *uniform interface* mendefinisikan *interface* antara klien dan server. Ia menyederhanakan dan memisahkan arsitektur, sehingga memungkinkan setiap bagian dikembangkan secara mandiri.
- *Stateless* : Keadaan yang diperlukan untuk menangani *request* terkandung dalam *request* itu sendiri, entah sebagai bagian dari *URI*, *query-string parameter*, *body*, atau *header*.
- *Cacheable* : Respons yang dikirimkan *REST* harus menyatakan dirinya apakah ia dapat di-*cache* atau tidak, untuk mencegah klien memakai ulang data yang sudah tidak berlaku atau tidak sesuai dalam respons untuk *request* yang diterima berikutnya.
- *Client-Server* : *Uniform interface* memisahkan klien dari server, sehingga klien tidak berurusan dengan penyimpanan data. Dengan begitu, kode yang ada pada sisi klien menjadi semakin *portable*. Server tidak berurusan dengan keadaan atau *interface* yang ada pada sisi klien, sehingga server menjadi semakin sederhana dan semakin *scalable*. Dengan begitu, klien dan server dapat dikembangkan secara independen, selama koneksi antara klien dan server tidak diubah.
- *Layered System* : Klien tidak dapat mengetahui begitu saja apakah ia langsung terhubung dengan server akhir atau ada server perantara diantaranya. Server perantara dapat meningkatkan *scalability* sistem dengan mengaktifkan *load-balancing* dan menyediakan *shared cache*.
- *Code on Demand* (opsional) : Server dapat untuk sementara memperluas atau menyesuaikan fungsionalitas klien dengan mengirimkan logika yang dapat dijalankan oleh klien.

Di antara batasan-batasan yang telah dituliskan, hanya *Code on Demand* yang bersifat opsional. Jika sebuah servis melanggar batasan lainnya, ia tidak dapat dinyatakan seutuhnya sebagai *RESTful* (Fredrich, 2012).



Gambar 2.3. REST API

(Sumber: http://blog.ciaranoconnor.me/2016/03/20/rest-api-swagger/)

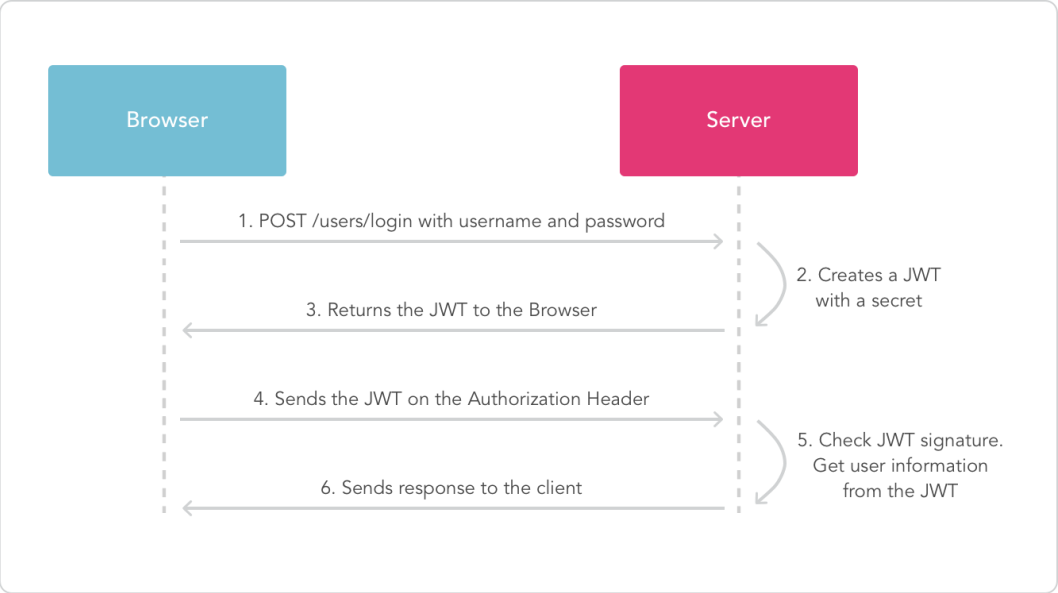
2.6. JSON Web Token

JSON Web Token adalah sebuah standar terbuka yang mendefinisikan cara yang tersusun dan mandiri untuk mengamankan informasi pengiriman antara beberapa pihak sebagai sebuah obyek *JSON*. Informasi ini dapat diverifikasi dan dipercayai karena ditandatangani secara *digital*. *JSON Web Token* memiliki kegunaan dalam hal melakukan autentikasi dan pertukaran informasi. *JSON Web Token* terdiri atas tiga bagian yang dipisahkan dengan tanda titik (.), yaitu :

- *Header* : terdiri atas dua bagian, yaitu tipe token dan algoritma *hashing* yang digunakan.
- *Payload* : mengandung *claim*, yaitu pernyataan mengenai sebuah entitas dan *metadata* tambahan.
- *Signature* : dipakai untuk memverifikasi bahwa pengirim *JWT* adalah seperti yang terkandung dalam token dan untuk memastikan bahwa pesan tidak diubah selama pemakaian *JSON Web Token*.

Sehingga, struktur *JSON Web Token* umumnya terlihat seperti "xxxxx.yyyyy.zzzzz", di mana "xxxx" adalah *header*, "yyyyy" adalah *payload*, dan "*zzzzz*" adalah *signature*. Dalam pemakaiannya untuk autentikasi, ketika seorang pengguna berhasil melakukan *login* memakai kredensial yang dimiliki, sebuah *JSON Web Token* akan dikembalikan dan harus disimpan secara lokal, dibandingkan dengan pendekatan tradisional, yaitu membuat sebuah *session* di

dalam server dan mengembalikan *cookie*. Ketika pengguna ingin mengakses sumber daya yang diamankan, *JSON Web Token* harus dikirim dari sisi pengguna. Cara ini disebut sebagai mekanisme autentikasi yang *stateless*, karena *state* pengguna tidak pernah disimpan di dalam memori server. Karena *JSON Web Token* bersifat mandiri, semua informasi yang diperlukan berada di dalam *JSON Web Token*, mengurangi keperluan untuk melakukan *query database* berulang kali (Auth0, n.d.).



Gambar 2.4. Mekanisme Autentikasi dengan JSON Web Token
(Sumber: https://jwt.io/introduction/)