BAB 2
TINJAUAN PUSTAKA

2.1 Profil Universitas Pendidikan Indonesia

Penelitian ini memiliki kaitan dengan Universitas Pendidikan Indonesia, oleh karena itu akan dijelaskan mengenai profil dari Universitas Pendidikan Indonesia. Penjelasan mengenai profil Universitas Pendidikan Indonesia akan dibagi menjadi 4 bagian, yaitu penjelasan Sejarah, Logo, Visi dan Misi, dan Tujuan.

2.1.1 Sejarah

Universitas Pendidikan Indonesia (UPI) merupakan salah satu Perguruan Tinggi di Indonesia yang berdiri sejak tahun 1954. UPI berawal dari sebuah Perguruan Tinggi Pendidikan Guru yang telah mengalami metamorfosis hingga menjadi salah satu Perguruan Tinggi Berbadan Hukum di Indonesia.

Universitas Pendidikan Indonesia semula berupa Perguruan Tinggi Pendidikan Guru (PTPG) yang didirikan pada tanggal 20 Oktober 1954 di Bandung dan diresmikan oleh Menteri Pendidikan Pengajaran Mr. Muhammad Yamin. Di sinilah untuk pertama kalinya para pemuda mendapat gemblengan pendidikan guru pada tingkat universitas.

Sejalan dengan Surat Keputusan Menteri Pendidikan, Pengajaran dan Kebudayaan No. 40718/S, PTPG dapat berdiri sendiri menjadi perguruan tinggi atau perguruan tinggi dalam universitas, maka seiring dengan berdirinya Universitas Padjadjaran (UNPAD), pada tanggal 25 November 1958 PTPG diintegrasikan menjadi fakultas utama Universitas Padjadjaran dengan nama Fakultas Keguruan dan Ilmu Pendidikan (FKIP). Untuk menghilangkan dualisme tersebut, pada tanggal 1 Mei 1963 dikeluarkan Keputusan Presiden Nomor 1 tahun 1963, yang melebur FKIP dan IPG menjadi Institut Keguruan dan Ilmu Pendidikan (IKIP) sebagai satu satunya lembaga pendidikan guru tingkat universitas.

Seiring dengan kebijakan pemerintah di bidang pendidikan tinggi yang memberikan perluasan mandat bagi Lembaga Pendidikan Tenaga Kependidikan (LPTK) yang harus mampu mengikuti tuntutan perubahan serta mengantisipasi segala kemungkinan dimasa datang, IKIP Bandung diubah menjadi Universitas Pendidikan Indonesia melalui Keputusan Presiden RI No. 124 tahun 1999 tertanggal 7 Oktober 1999.

Untuk memperluas jangkauan dalam mendukung pembangunan nasional, UPI harus mampu berdiri sendiri dan berkiprah. Kebulatan tekad ini menumbuhkan keyakinan akan kemampuan yang telah dimilikinya. Mulai tahun 2004, berdasarkan Peraturan Pemerintah Nomor 6 Tahun 2004, UPI diberi otonomi dan menjadi perguruan tinggi BHMN. Pada tahun 2012, status UPI dikembalikan menjadi perguruan tinggi negeri (bahasa resmi: perguruan tinggi yang diselenggarakan oleh pemerintah) berdasarkan Peraturan Presiden Nomor 43 Tahun 2012.

Pada tanggal 28 Februari 2014, UPI berubah menjadi Perguruan Tinggi Negeri Berbadan Hukum (PTNBH), berdasarkan Peraturan Pemerintah Republik Indonesia Nomor 15 Tahun 2014, tentang Statuta Universitas Pendidikan Indonesia. Pengembangan dan peningkatan UPI tidak saja berorientasi pada bidang akademik, tetapi juga dalam berbagai bidang, termasuk pemantapan konsep dan rencana pembangunannya.

2.1.2 Logo

Adapun logo dari Universitas Pendidikan Indonesia yang dapat dilihat pada gambar 2.1 :

Gambar 2.1 Logo Universitas Pendidikan Indonesia

Sumber gambar : mdinuad.blogspot.com

Penjelasan Logo :

1. **Bentuk logo yang bulat penuh**, selain melambangkan pengakuan yang penuh atas kemahabesaran dan kemahasempurnaan Allah SWT, juga melambangkan (menyatakan) bahwa segenap warga UPI, baik sivitas akademika maupun tenaga kependidikannya telah bertekad bulat dan sepakat untuk bersama-sama melaksanakan tugas masing-masing, sesuai dengan visi, misi, dan fungsi UPI, dengan penuh tanggung jawab dan berserah diri secara total kepada Allah Yang Maha Kuasa.
2. **Tiga warna pokok di atas warna dasar putih**, yaitu merah-hitamkuning, merupakan warna khas PTPG, menyiratkan kilasan historis akan masa-masa awal berdirinya perguruan tinggi pendidikan guru di tanah air tercinta ini. Ketiga warna itu menyiratkan juga asas kehidupan kampus yang edukatif, ilmiah, dan religius, di atas landasan kekeluargaan, persaudaraan dan kebersamaan, dalam suasana silih asih, silih asah, dan silih asuh.
3. **Warna merah yang berpadu dengan warna dasar putih**, menyiratkan Dwiwarna Sang Merah Putih sebagai bendera Negara

Kesatuan Republik Indonesia yang melambangkan wawasan kebangsaan serta semangat kesatuan dan persatuan yang harus selalu dibina dalam kehidupan berbangsa dan bernegara, tak terkecuali dalam kehidupan akademik.

4. **Buku yang terbuka dan tinta emas (bulatan kecil),** mengisyaratkan bahwa UPI merupakan tempat diselenggarakannya pendidikan formal, tempat belajar dan mengajar, tempat mengggali dan mengembangkan ilmu pengetahuan, teknologi, dan seni. Dari buku, insan-insan kampus dapat memperoleh informasi dan butir-butir kebenaran, dan dengan pena dan tinta, insan-insan kampus dapat mengekspresikan pikiran, perasaan dan kebenaran. Selain menggambarkan tinta, bulatan emas juga mengingatkan kita bahwa ilmu yang diamalkan ibarat matahari yang merupakan sumber cahaya (penerang) dan sumber energi. Buku yang terbuka dan pena dan logo digambarkan putih, tanpa warna, menyatakan bahwa ilmu yang digambarkan dan dikembangkan itu murni, jernih, bebas dari kontaminasi dan muatan kepentingan yang sempit.
5. **Sepasang daun berwarna hitam dan sepasang mahkota bunga serta kepala putik berwarna merah**, menyiratkan bahwa UPI merupakan persemaian dan berseminya tunas-tunas bangsa serta kader-kader pemimpin bangsa. Pasangan daun dan bunga juga merujuk kepada Tanah Priangan yang subur dan Bandung Kota Kembang yang menjadi semangat kebangkitan negara Asia Afrika yang marak semerbak, serta Bumi Siliwangi yang harum mewangi, tempat berdiri tegaknya lembaga ini. Warna merah melambangkan semangat keberanian dalam menemukan dan membela kebenaran, baik kebenaran ilmiah maupun kebenaran ilahiah. Warna hitam menyiratkan suasana solid dan stabil sebagai dasar pertumbuhan yang mantap. Warna kuning melambangkan ketegaran, kebesaran (hati), kewibawaan, dan kearifan.

6. **Garis-garis lengkung cakrawala (horison)**, menyiratkan bahwa UPI mempunyai visi ke depan, berwawasan nasional, serta berpandangan global. Garis cakrawala yang berjumlah tiga melambangkan *hablumminallah*, yang menyiratkan semangat religius yang mewarnai segala lini kehidupan insan kampus. Garis lurus vertikal itu juga menyirat misi UPI sebagai lembaga pembina dan penghasil insaninsan berilmu, beriman, dan bertakwa kepada Allah Yang Maha Esa, penghasil insan yang berakhlaq, *akhlaqul karimah*.

2.1.3 Visi dan Misi

a. Visi

Sejalan dengan arah pengembangan, jati diri, dan tantangan ke depan, rumusan visi Universitas Pendidikan Indonesia adalah Pelopor dan Unggul (Leading and Outstanding).

b. Misi

1. Menyelenggarakan Pendidikan dengan membina dan mengembangkan disiplin ilmu pendidikan dan pendidikan disiplin ilmu, serta disiplin ilmu agama, ilmu sosial, ilmu alam, ilmu formal, ilmu terapan secara proposional untuk memperkuat disiplin ilmu pendidikan dan pendidikan disiplin ilmu.
2. Mengembangkan Pendidikan Profesional Guru yang terintegrasi dalam pendidikan akademik dan profesi untuk semua jalur dan jenjang pendidikan.
3. Menyebarluaskan pengalaman dan temuan-temuan inovatif dalam disiplin ilmu pendidikan, pendidikan disiplin ilmu, ilmu agama, ilmu humaniora, ilmu sosial, ilmu alam, ilmu formal, dan ilmu terapan demi kemajuan masyarakat.

2.1.4 Tujuan

Universitas Pendidikan Indonesia memiliki tujuan :

1. Menghasilkan pendidik, tenaga kependidikan, ilmuwan, dan tenaga ahli pada semua jenis dan program pendidikan tinggi, yang bertakwa

kepada Tuhan Yang Maha Esa dan memiliki keunggulan kompetitif dan komparatif global

2. Menghasilkan, mengembangkan, dan menyebarluaskan ilmu pengetahuan dan teknologi untuk meningkatkan kesejahteraan masyarakat

2.1.5 Struktur Organisasi Direktorat Sistem dan Teknologi Informasi

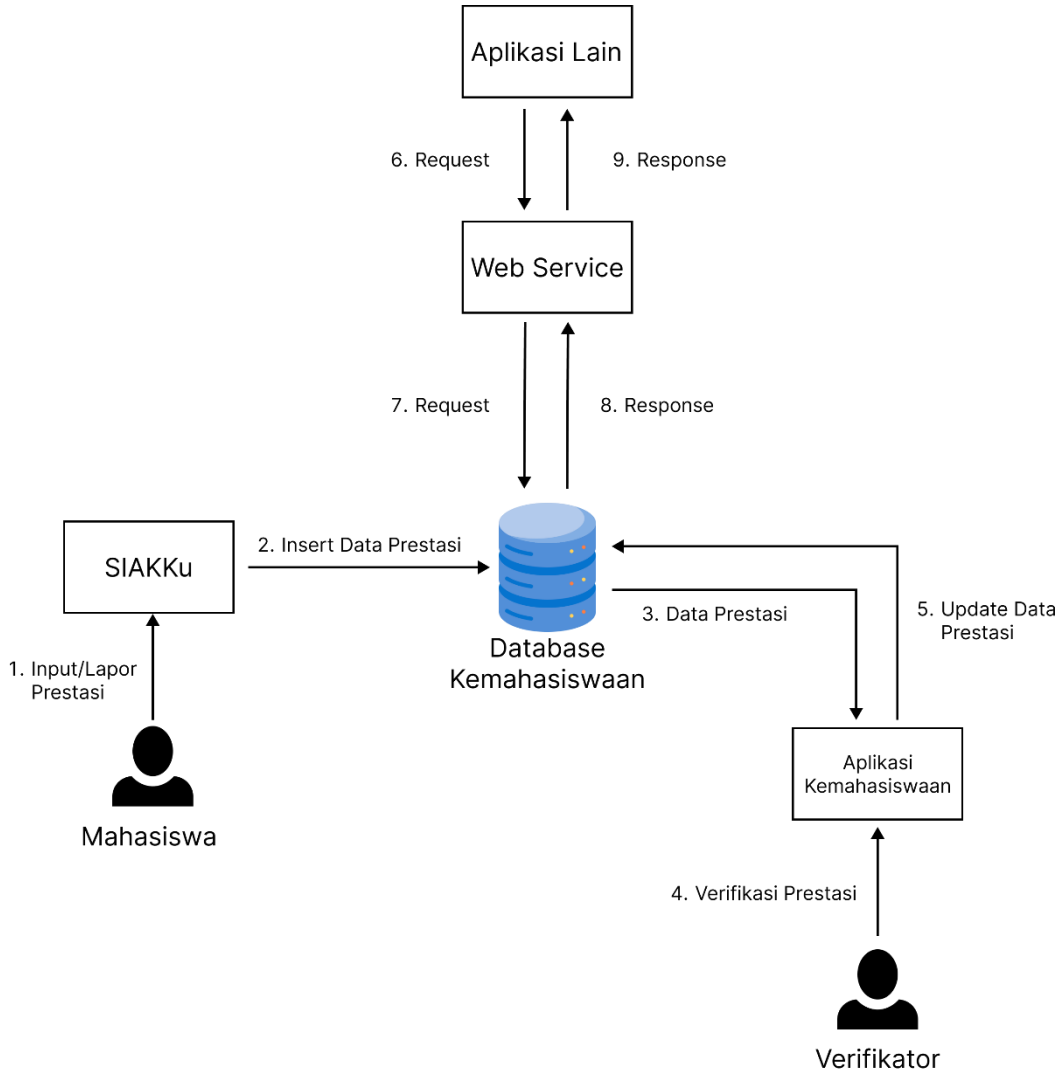
Direktorat Sistem dan Teknologi Informasi (DSTI) merupakan unit kerja di bawah Universitas Pendidikan Indonesia yang bertanggung jawab dalam pengelolaan teknologi informasi dan komunikasi di lingkungan UPI. Berikut ini struktur organisasi DSTI yang disampaikan pada Gambar 2.2 Struktur Organisasi DSTI :

Gambar 2.2 Struktur Organisasi DSTI

2.2 Web Service Data Prestasi Mahasiswa

Universitas Pendidikan Indonesia memiliki 2 sistem yang digunakan untuk keperluan yang berkaitan dengan kemahasiswaan, yaitu *SIAKKu* (student.upi.edu) dan aplikasi Kemahasiswaan. Terdapat kebutuhan data baru di Universitas Pendidikan Indonesia yang nantinya akan diintegrasikan ke 2 sistem tersebut, yaitu kebutuhan data tentang prestasi Mahasiswa. Hal ini menimbulkan suatu permasalahan, yaitu Mahasiswa harus mengakses 2 portal untuk memasukan data-data yang berkaitan dengan prestasi Mahasiswa tersebut.

Untuk mengatasi permasalahan tersebut, pada tahun 2022, dibangun sebuah Web Service Data Prestasi Mahasiswa yang bertujuan agar Mahasiswa hanya mengakses 1 portal saja, yaitu *SIAKKu*. Dengan demikian, Mahasiswa melakukan proses *input* data melalui *SIAKKu*, lalu data akan disinkronkan secara otomatis ke aplikasi Kemahasiswaan. Gambaran dari alur Web Service Data Prestasi Mahasiswa yang sedang berjalan disajikan pada Gambar 2.3 Alur Web Service Data Prestasi Mahasiswa :



Gambar 2.3 Alur Web Service Data Prestasi Mahasiswa

Gambar 2.3 merupakan penjelasan bagaimana alur dari Web Service Data Prestasi Mahasiswa yang sedang berjalan saat ini, untuk penjelasan yang lebih rinci akan dijelaskan pada poin-poin berikut :

1. Mahasiswa menginputkan atau melaporkan prestasi ke SIAKKu.
2. Data yang telah diinputkan atau dilaporkan oleh Mahasiswa disimpan di Database Kemahasiswaan.
3. Aplikasi Kemahasiswaan memakai data prestasi yang disimpan di database kemahasiswaan
4. Verifikator melakukan verifikasi terhadap data prestasi yang disimpan di database kemahasiswaan.
5. Aplikasi Kemahasiswaan melakukan *request update* ke database kemahasiswaan berdasarkan data prestasi yang sudah diverifikasi oleh Verifikator.
6. Aplikasi lain melakukan *request* ke web service untuk get data prestasi mahasiswa.
7. Web Service melakukan *request* untuk *get* data prestasi mahasiswa ke Database Kemahasiswaan.
8. Database Kemahasiswaan mengirim *response* yang berisi data prestasi mahasiswa.
9. Web Service mengirim *response* yang berisi data prestasi mahasiswa ke Aplikasi Lain.

2.3 JSON

Javascript Object Notation (JSON) merupakan seperangkat aturan untuk memformat data berbasis teks yang ringan digunakan pada pertukaran data. JSON berasal dari bahasa pemrograman ECMAScript, merupakan bahasa pemrograman yang independen. JSON mendefinisikan seperangkat kecil aturan untuk mengatur representasi secara portabel dari data terstruktur [10].

JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, Javascript, Perl, Python,

dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. JSON terbuat dari dua struktur :

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (hash table), daftar berkunci (*keyed list*), atau associative array.
2. Daftar nilai yang terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

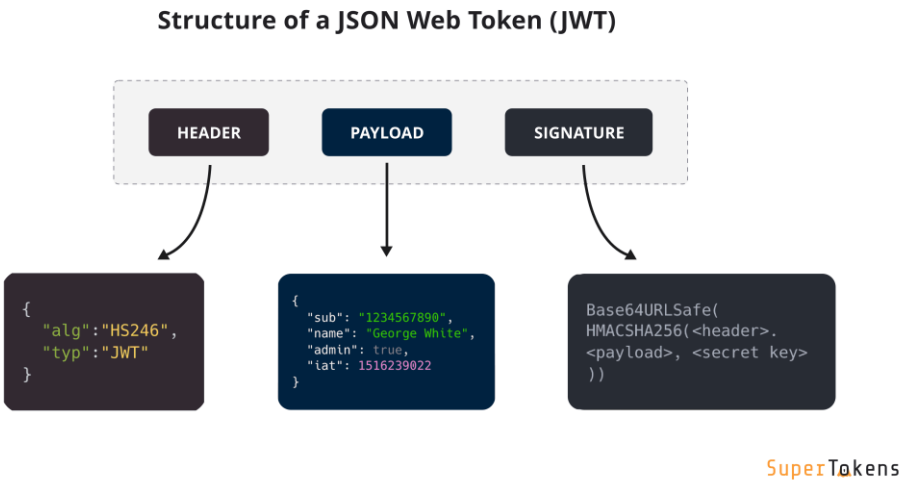
Struktur tersebut adalah struktur data universal, hampir semua bahasa pemrograman modern mendukung struktur tersebut dalam satu bentuk atau bentuk yang lain. Dengan demikian bahwa format data bisa dipertukarkan dengan bahasa pemrograman didasarkan pada struktur tersebut [10].

2.4 Access Token

Access Token merupakan *string unique* dan acak yang berisi angka dan huruf. Access token adalah sebuah kunci yang dibutuhkan oleh pemohon layanan yang harus digunakan untuk mengidentifikasi dan memverifikasi apakah pemohon memiliki hak dalam mengakses web service [11]. Masing-masing pemilik akun pada layanan web service memiliki sebuah access token yang bersifat unik. Access token ini berfungsi untuk menambahkan tingkat keamanan pada web service, sehingga web service hanya dapat diakses oleh pihak yang berwenang saja.

2.5 JSON Web Token (JWT)

JWT merupakan sebuah token yang menyimpan informasi untuk proses otentikasi dan penukaran informasi. JWT umumnya digunakan dalam proses otorisasi karena token yang dihasilkan berisi informasi mengenai siapa yang mengakses sistem [12]. Informasi yang terdapat pada token JWT dapat dilihat pada Gambar 2.4 Jenis Informasi pada Token JWT :



Gambar 2.4 Jenis Informasi pada Token JWT

Sumber gambar : supertokens.com

Pada gambar 2.4 terdapat 3 jenis informasi yang disimpan di dalam token. 3 jenis informasi tersebut adalah *header*, *payload*, dan *signature*. Pada bagian header, terdapat informasi tentang jenis token dan penandatanganan algoritma yang digunakan. Kemudian pada bagian payload, terdapat beberapa data yang berisi tentang identitas token dan juga data yang menyimpan informasi tentang pengguna. Bagian terakhir adalah signature, di bagian signature ini terdapat secret key atau kunci rahasia yang digunakan sebagai enkripsi validasi dari header dan payload [12]. Adapun algoritma *kriptografi* yang biasa digunakan oleh JWT dapat dilihat pada Gambar 2.5 Algoritma JSON Web Token :

"alg" parameter value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended
ES384	ECDSA using 384 and SHA-384	Optional
ES512	ECDSA using P-512 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

Gambar 2.5 Algoritma JSON Web Token

Sumber gambar : IOP Conference Series: Materials Science and Engineering [6]

2.6 Library Tymon/JWT-Auth

Tymon/JWT-Auth adalah sebuah *library* atau modul yang digunakan untuk mengimplementasikan sistem autentikasi berbasis JSON Web Token (JWT) di dalam aplikasi Laravel. *Library* ini dibangun oleh Sean Tymon pada tahun 2014, pada versi awal ini, *library* menyediakan fitur dasar seperti pembuatan dan validasi token, *middleware* untuk melindungi *route*, dan pengelolaan Pengguna yang terautentikasi. Seiring berjalannya waktu, banyak fitur tambahan dan perbaikan yang dilakukan. Beberapa fitur yang ditambahkan termasuk *refresh* token dan peningkatan keamanan. Berikut merupakan *method-method* atau fitur yang ada pada *library* Tymon/JWT-Auth :

1. Attempt()

Attempt() adalah *method* untuk memverifikasi identitas seorang Pengguna dengan menggunakan beberapa kredensial, biasanya berupa kombinasi *username* dan *password*. Berikut merupakan contoh sintaks *method* attempt() yang ditunjukkan pada Tabel 2.1 Method Attempt() :

Tabel 2.1 Method Attempt()

```
1. // Generate a token for the user if the credentials are valid
2. $token = auth()->attempt($credentials);
```

Method ini mengembalikan token JWT jika kredensial valid, dan mengembalikan *null* jika kredensial tidak valid.

2. Login()

Login() adalah *method* untuk membuat Pengguna login dan membuatkan token JWT bagi Pengguna. Berikut merupakan contoh sintaks *method* login() yang ditunjukkan pada Tabel 2.2 Method Login() :

Tabel 2.2 Method Login()

```
1. // Get some user from somewhere
2. $user = User::first();
3.
4. // Get the token
5. $token = auth()->login($user);
```

Method ini mengembalikan token JWT jika *true*.

3. User()

User() adalah *method* untuk mengambil data Pengguna yang sudah terautentikasi. Berikut merupakan contoh sintaks *method* user() yang ditunjukkan pada Tabel 2.3 Method User() :

Tabel 2.3 Method User()

```
1. // Get the currently authenticated user
2. $user = auth()->user();
```

Method ini mengembalikan data Pengguna jika Pengguna sudah terautentikasi, jika belum maka akan mengembalikan *null*.

4. UserOrFail()

UserOrFail() sebenarnya sama seperti *method* User(), yaitu mengambil data Pengguna. Namun yang membedakan adalah pada penanganan kondisi jika Pengguna belum terautentikasi. Berikut merupakan contoh sintaks *method* userOrFail() yang ditunjukkan pada Tabel 2.4 Method UserOrFail() :

Tabel 2.4 Method UserOrFail()

```
1. try {
2.     $user = auth()->userOrFail();
3. } catch (\Tymon\JWTAuth\Exceptions\UserNotDefinedException $e) {
4.     // do something
5. }
```

Method ini mengembalikan data Pengguna jika Pengguna terautentikasi, jika tidak maka akan melemparkan (*thrown*) sebuah *exception yaitu* Tymon\JWTAuth\Exceptions\UserNotDefinedException.

5. Logout()

Logout() adalah *method* untuk membuat Pengguna *logout*, yang berarti akan membuat token yang dimiliki Pengguna saat ini menjadi tidak valid (*invalidate*) dan menghapus (*unset*) Pengguna yang terautentikasi. Berikut merupakan contoh sintaks *method* logout() yang ditunjukkan pada Tabel 2.5 Method Logout() :

Tabel 2.5 Method Logout()

```
1. auth()->logout();
2.
3. // Pass true to force the token to be blacklisted "forever"
4. auth()->logout(true);
```

6. Refresh()

Refresh() adalah *method* untuk membuat ulang (*refresh*) token yang sudah ada dan membuat token yang sudah ada tersebut menjadi tidak valid (*invalidate*). Berikut merupakan contoh sintaks *method* refresh() yang ditunjukkan pada Tabel 2.6 Method Refresh :

Tabel 2.6 Method Refresh()

```
1. $newToken = auth()->refresh();
2.
3. // Pass true as the first param to force the token to be blacklisted "forever".
4. // The second parameter will reset the claims for the new token
5. $newToken = auth()->refresh(true, true);
```

7. Invalidate()

Invalidate() adalah *method* untuk membuat sebuah token menjadi tidak valid (*invalidate*). Berikut merupakan *method* invalidate() yang ditunjukkan pada Tabel 2.7 Method Invalidate() :

Tabel 2.7 Method Invalidate()

```
1. auth()->invalidate();
2.
3. // Pass true as the first param to force the token to be blacklisted "forever".
4. auth()->invalidate(true);
```

Jika parameter *true* ditambahkan pada *method* tersebut maka token akan ditambahkan ke dalam *blacklist*.

8. TokenById()

TokenById() adalah *method* untuk mendapatkan token berdasarkan kode atau *id* Pengguna. Id Pengguna dimasukkan pada parameter *method* tokenById(). Berikut merupakan contoh sintaks *method* tokenById yang ditunjukkan pada Tabel 2.8 Method TokenById() :

Tabel 2.8 Method TokenById()

```
1. $token = auth()->tokenById(123);
```

9. Payload()

Payload() adalah *method* untuk mendapatkan bagian *payload* token JWT mentah (*raw*). Setelah *payload* didapat, *claims-claims* yang ada pada *payload* dapat diakses. Berikut merupakan contoh sintaks *method* payload() yang ditunjukkan pada Tabel 2.9 Method Payload() :

Tabel 2.9 Method Payload()

```
1. $payload = auth()->payload();
2.
3. // then you can access the claims directly e.g.
4. $payload->get('sub'); // = 123
5. $payload['jti']; // = 'asfe4fq434asdf'
6. $payload('exp') // = 123456
7. $payload->toArray(); // = ['sub' => 123, 'exp' => 123456, 'jti' =>
'asfe4fq434asdf'] etc
```

10. Validate()

Validate() adalah method untuk memvalidasi kredensial milik Pengguna. Berikut merupakan contoh sintaks *method* validate() yang ditunjukkan pada Tabel 2.10 Method Validate() :

Tabel 2.10 Method Validate()

```
1. if (auth()->validate($credentials)) {
2.     // credentials are valid
3. }
```

Terdapat juga kegunaan lanjutan yang dapat dilakukan oleh *library* Tymon/JWT-Auth, yaitu :

1. Menambah *custom claims*

   Default-nya, terdapat 6 *claims* yang harus ada pada *payload* token JWT, diantaranya adalah *iss, iat, exp, nbf, sub, jti*. Namun, *library* Tymon/JWT-Auth menyediakan *method* untuk menambah *claims* (*custom claims*) yang dapat ditambahkan sesuai kebutuhan. Berikut merupakan contoh sintaks *method* claims() yang ditunjukkan pada Tabel 2.11 Method Claims() :

Tabel 2.11 Method Claims()

```
1. $token = auth()->claims(['foo' => 'bar'])->attempt($credentials);
```

2. *Set* token secara eksplisit

   Library Tymon/JWT-Auth menyediakan *method* untuk menetapkan nilai token. Yang berarti token tidak dibuat secara otomatis, namun ditetapkan secara langsung (manual). Berikut merupakan contoh sintaks *method* setToken() yang ditunjukkan pada Tabel 2.12 Method SetToken() :

Tabel 2.12 Method SetToken()

```
1. $user = auth()->setToken('eyJhb...')->user();
```

3. *Set request* secara eksplisit

   Berikut merupakan contoh sintaks *method* setRequest() yang ditunjukkan pada Tabel 2.13 Method SetRequest() :

Tabel 2.13 Method SetRequest()

```
1. $user = auth()->setRequest($request)->user();
```

4. *Override* token *TTL*

   TTL (*Time To Live*) atau waktu hidup merupakan lamanya waktu valid dari token JWT. TTL menentukan berapa lama token akan tetap valid sebelum

kadaluwarsa (*invalidate*). *Default*-nya, *library* Tymon/JWT-Auth menetapkan waktu token valid selama 60 menit. Namun terdapat method yang dapat meng-override konfigurasi tersebut, yaitu *method* setTTL(). Berikut merupakan contoh sintaks *method* setTTL yang ditunjukkan pada Tabel 2.14 Method SetTTL() :

Tabel 2.14 Method SetTTL()

```
1. $token = auth()->setTTL(7200)->attempt($credentials);
```

Satuan waktu yang digunakan pada *method* tersebut adalah detik.

2.7 Web Service

Web Service adalah sistem perangkat lunak yang didesain dapat dioperasikan mesin ke mesin melalui jaringan [11]. Web Service memiliki dua metode yang berorientasi pada sumber daya informasi dan layanan informasi, yaitu *Simple Object Access Protocol* (SOAP) dan *Representational State Transfer* (REST). Saat ini metode REST adalah yang paling dominan digunakan untuk pengembangan Web Service dikarenakan lebih efektif dalam pengembangan dan penggunaan [13]. Format response yang dimiliki SOAP hanya berupa XML, berbeda dengan REST yang dapat memberikan *response* berupa XML, Javascript Object Notation (JSON) ataupun format text lainnya [14].

2.8 HMAC SHA

Tujuan dari HMAC adalah untuk mengautentikasi sumber pesan dan integritasnya dengan melampirkan *MAC (Message Authentication Code)* ke pesan tersebut. MAC dihasilkan dengan menggunakan dua parameter fungsional yang berbeda, yaitu *message input*, *M*, dan *secret key*, *K*, sedangkan komponen dasar dari HMAC adalah fungsi *hash* yang digunakan. Secara khusus, HMAC terdiri dari fungsi hash dan XOR [15].

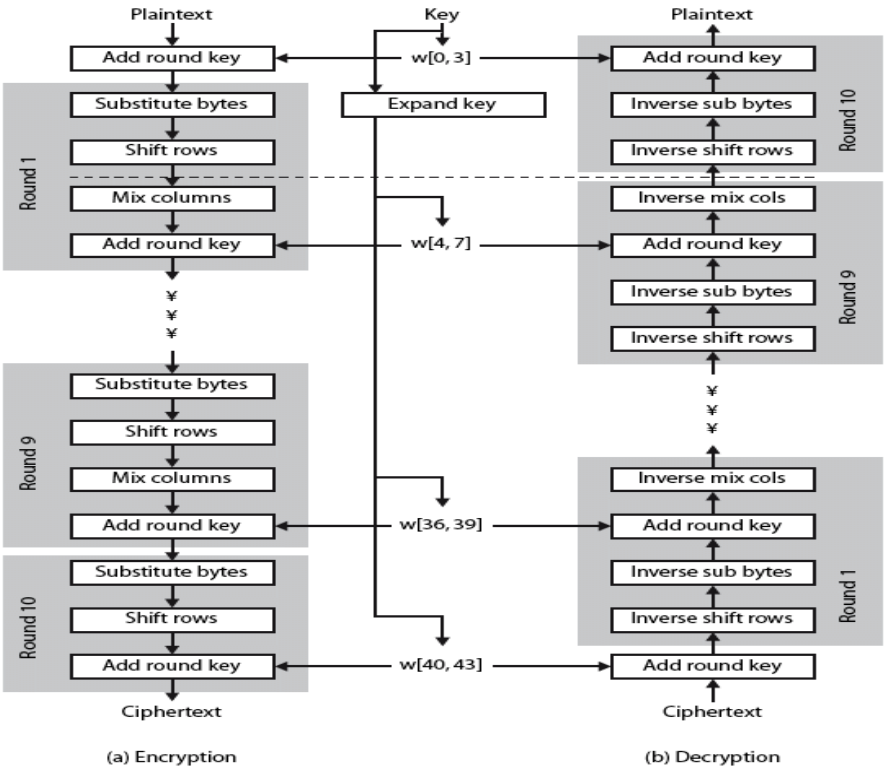
Fungsi hash memecah pesan menjadi blok berukuran tetap (*fixed-size blocks*) dan melakukan pemrosesan iteratif terhadapnya. Sebagai contoh, fungsi hash MD5, SHA-1, dan SHA-256 beroperasi pada blok 512-bit, sedangkan SHA-512 beroperasi pada blok 1024-bit. Untuk size output, HMAC(M), sama dengan

fungsi hash yang digunakan (128, 160, 256, atau 512 bit untuk MD5, SHA-1, SHA-256, atau SHA-512) [15].

2.9 AES (Advanced Encryption Standard)

Algoritma AES adalah algoritma kriptografi yang dapat mengenkripsi dan mendekripsi data dengan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit. AES memiliki kecepatan enkripsi dan dekripsi tertinggi, berikutnya Blowfish, DES dan IDEA [16]. Algoritma kriptografi AES termasuk dalam klasifikasi algoritma kriptografi kunci simetri, kunci yang digunakan pada enkripsi sama dengan kunci yang digunakan untuk dekripsi [17].

Enkripsi Algoritma AES dimulai dengan memasukan XOR plainteks/*state* yang akan dienkripsi dengan *round key*, setelah selesai melakukan XOR plainteks dengan round key, dilakukan substitusi dengan s-Box. setelah hasil substitusi dengan s-Box selesai, dilakukan *shiftrow*. Setelah hasil shiftrow didapat, maka langkah selanjutnya yaitu melakukan *Mix Columns* dengan mengalikan matriks. Setelah perhitungan mix column selesai, maka dilakukan *addroundkey* dengan melakukan XOR state dengan round key. Lakukan sampai iterasi 10, namun pada saat putaran/iterasi yang ke 10, setelah step shiftrow, lompati step Mix Column dan langsung lanjut melakukan XOR hasil state saat shift row dengan round key [17]. Berikut merupakan struktur dasar dari algoritma AES yang dapat dilihat pada Gambar 2.6 Basic Structure of AES :



Gambar 2.6 Basic Structure of AES

Sumber Gambar : Artikel Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data [18]

2.10 Representational State Transfer (REST)

REST merupakan gaya arsitektur dalam mendesain sebuah Web Service dimana desain REST memiliki *resource* yang dapat diakses melalui sebuah alamat HTTP URL yang *unique.* REST mengirimkan perintah yang akan dikerjakan oleh *server* menggunakan metode-metode HTTP *request method* yang disebut *verb.* Terdapat delapan HTTP request method, yaitu *GET, POST, PUT, DELETE OPTIONS, HEAD, TRACE, dan CONNECT* [11]. Dalam penggunaan *API REST* hanya menggunakan empat dari metode-metode tersebut, yaitu : *GET, POST, PUT, dan DELETE.*

Pesan yang diterima dari server berupa kode HTTP berhasil atau gagal di dalam header dan isi pesan hasil pengolahan program itu sendiri. Berikut adalah kode-kode HTTP yang sering digunakan dalam penggunaan REST API [11] :

a. 200 *OK*

   Perintah yang dikirim ke server benar dan berhasil dijalankan.

b. 400 *Bad Request*

   Perintah yang dikirim ke server berisi isian yang salah.

c. 401 *Unauthorized*

   Pengirim perintah mengirimkan kode kunci yang salah.

d. 403 *Forbidden*

   Pengirim tidak memiliki hak akses ke dalam resource yang dituju.

e. 404 *Not Found*

   Resource yang dituju tidak ditemukan dalam server.

f. 429 *Too Many Requests*

   Pengirim perintah mengakses mencapai/melebihi dari limit yang telah ditentukan dari batas waktu tertentu.

g. 500 *Internal Server Error*

   Server atau potongan program dalam resource mengalami kesalahan.

2.11 Simple Object Access Protocol (SOAP)

SOAP adalah sebuah XML-based mark-up languange untuk pergantian pesan diantara aplikasi. SOAP berguna seperti sebuah amplop yang digunakan untuk pertukaran data object didalam network. SOAP mendefinisikan empat aspek dalam komunikasi : *Message Envelope*, *Encoding*, *RPC call convention*, dan bagaimana menyatukan sebuah message didalam protokol transport [2].

Sebuah SOAP message terdiri dari *SOAP Envelop* dan bisa terdiri dari *attachments* atau tidak memiliki attachment. SOAP Envelop tersusun dari *SOAP header* dan *SOAP body*, sedangkan SOAP attachments membolehkan non-XML data untuk dimasukkan kedalam *SOAP message*, di-encoded, dan diletakkan kedalam SOAP message dengan menggunakan MIME-multipart. HTTP berbasis API berarti API yang diekspos sebagai salah satu atau lebih HTTP URI dan

respon berupa XML/JSON. Skema respon dapat dikustomisasi untuk setiap objek [2].

2.12 Autentikasi

Autentikasi merupakan sebuah proses untuk pembuktian keaslian dari objek. Proses autentikasi terhadap seseorang dilakukan untuk verifikasi identitas, sedangkan autentikasi pada objek untuk konfirmasi nilai kebenaran dari objek tersebut. Autentikasi terdiri dari beberapa jenis, yaitu autentikasi dengan *username* dan *password*, autentikasi dengan *code* atau *certificate*, autentikasi dengan *biometric*, dan autentikasi dengan *smart card*. Teknik autentikasi terus berkembang dengan pemanfaatan keunikan dari suatu objek [19].

2.13 Cross-Site Scripting (XSS)

Cross-Site Scripting/XSS adalah kerentanan (*vulnerability*) yang dapat menyebabkan seorang penyerang mengirimkan kode berbahaya kepada pengguna lain. XSS juga dapat diinterpretasikan sebagai kelemahan yang terjadi karena server web tidak dapat memvalidasi data masukan yang diberikan pengguna [20]. Serangan XSS dapat dieksekusi di situs web dengan keamanan yang rentan, baik ditulis dalam kode HTML, Javascript, VBScript, PHP, maupun bahasa pemrograman lain [21]. Serangan XSS dibagi menjadi 3 tipe [22], berikut penjelasannya :

1) Reflected XSS

Pada tipe ini, penyerang menempatkan script/kode untuk mencuri *cookie* milik korban, dengan tujuan untuk mengidentifikasi dirinya (penyerang) seolah-olah itu adalah *session* penyerang. Dengan *cookie* yang telah didapat, penyerang dapat melakukan aksi dengan menggunakan izin dari korban tanpa menggunakan berbagai tipe password apapun. Serangan ini umumnya terjadi pada *search engine*, biasanya, kode disisipkan melalui *form*, *url*, *cookies*, *flash program*, atau video. Serangan ini mengeksploitasi kerentanan dalam aplikasi web yang menggunakan (atau mencerminkan) informasi yang diberikan oleh pengguna untuk menghasilkan *exit page*.

Pada tipe ini, kode diarahkan melalui mekanisme ketiga. Sebagai contoh, melalui *spoofing* (e-mail). Konsekuensinya adalah pengalihan lalu lintas pengguna (*user's traffic*) ke aplikasi web milik penyerang. Jika aplikasi web memiliki kerentanan XSS, eksekusinya akan dilakukan dalam lingkungan terpercaya dari situs web yang menyelenggarakan (*host*) aplikasi tersebut.

2) Direct XSS

Penyerang menyisipkan kode HTML berbahaya secara langsung ke halaman web atau situs yang rentan terhadap serangan tersebut. Dalam serangan ini memerlukan tag-programming (seperti *script* pada *javascript*). Kode-kode ini dibuat permanen untuk semua pengguna di halaman web yang terkena serangan setelah serangan pertama berhasil dijalankan. Akibatnya, setiap kali seseorang memasuki section/halaman web yang dimana terdapat kode yang disisipkan, kode tersebut akan dieksekusi di browser mereka, dan mereka akan mematuhi tindakan-tindakan yang diprogram dalam *script* mereka.

Tipe ini lebih berbahaya karena didasarkan pada penyisipan kode berbahaya dalam konten yang disimpan di server aplikasi web eksternal. Artinya, data yang dikirim oleh penyerang disimpan secara permanen di server dan kemudian ditampilkan kepada pengguna yang mengunjungi situs web.

3) DOM XSS

Dikenal sebagai *Type 0* atau *DOM-Based XSS*, tipe ini dianggap sebagai serangan yang rumit dan kurang dikenal. Perbedaannya adalah kode berbahaya disisipkan melalui URL tetapi tidak dimuat sebagai bagian dari web dalam *source code*. Deteksinya lebih sulit karena muatan berbahaya (*malicious load*) tidak mencapai server. Tipe ini dianggap sebagai *local XSS* karena kerusakan disebabkan oleh *script* yang ada di sisi klien. Pada dasarnya, ketika halaman yang terinfeksi dibuka, kode berbahaya mengeksploitasi beberapa kerentanan untuk menginstal kode berbahaya tersebut di dalam file browser web dan dieksekusi tanpa verifikasi sebelumnya. Tidak seperti *Reflected* atau *Direct* XSS, tipe ini tidak melibatkan server pada serangannya. Namun, seperti *Reflected XSS*, serangan ini memerlukan pengguna untuk mengklik tautan, sehingga *script* pada

halaman web memilih *URL variables* dan mengeksekusi kode didalamnya. Metode ini lebih efektif untuk mencuri *session cookies*.

2.14 PHP Dan Laravel

PHP adalah sebuah bahasa pemrograman sumber terbuka (*open source*) yang ditujukan pada pemrograman web dan dapat diaplikasikan ke HTML. PHP merupakan bahasa skrip yang ditanam dalam HTML. Ini berarti bahwa kita dapat menggabungkan kode PHP dan HTML dalam berkas yang sama [11]. PHP merupakan bahasa pemrograman *server-side* yang dimana program akan dijalankan di server dan hasilnya akan diintegrasikan ke dalam kode sumber HTML.

Laravel adalah sebuah *framework* PHP yang dirilis dibawah lisensi MIT dengan kode sumber yang sudah disediakan oleh Github, sama seperti *framework-framework* yang lain [23]. Laravel dibangun dengan konsep MVC (*Model-Controller-View*), kemudian Laravel dilengkapi dengan *command line tool* yang bernama "*artisan*" yang bisa digunakan untuk *packaging bundle* dan instalasi *bundle* melalui *command prompt*.

2.15 MySQL

MySQL adalah sebuah aplikasi server basis data yang dijalankan di server, MySQL menggunakan sintak SQL standar dalam penggunaannya. MySQL bersifat relasional yang memungkinkan satu tabel dapat berelasi dengan tabel lainnya, sehingga memungkinkan dilakukannya normalisasi untuk mencegah redundansi dalam pemakaian data [11].

2.16 JavaScript

JavaScript adalah bahasa pemrograman yang dinamis dan fleksibel, digunakan secara luas untuk pengembangan web. Awalnya dikembangkan oleh Netscape sebagai cara untuk menambahkan logika dinamis dan interaktif ke situs web, JavaScript telah berkembang menjadi salah satu bahasa paling populer di dunia. Ini dapat dijalankan di sisi klien (*browser*) untuk membuat halaman web lebih interaktif dan responsif, serta di sisi server (menggunakan platform seperti Node.js) untuk membangun aplikasi *back-end* yang *scalable* dan efisien.

JavaScript mendukung paradigma pemrograman berorientasi objek, fungsional, dan imperatif, memungkinkan pengembang untuk menulis kode yang modular dan mudah dipelihara. Dengan bantuan kerangka kerja dan pustaka seperti React, Angular, dan Vue.js, JavaScript mempermudah pengembangan aplikasi web yang kompleks dan kaya fitur. Selain itu, JavaScript adalah inti dari pengembangan teknologi web modern seperti AJAX, yang memungkinkan pemuatan data secara asinkron tanpa perlu me-refresh halaman. Sebagai bahasa pemrograman yang terus berkembang, JavaScript memiliki ekosistem yang luas dan komunitas yang aktif, menjadikannya pilihan utama untuk banyak proyek pengembangan web dan aplikasi modern.

2.17 Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah “bahasa” yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat model untuk segala jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi, karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C#, atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C [24].

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. UML mendefinisikan diagram menjadi 8 bagian, yaitu *use case diagram*, *class diagram*, *statechart diagram*, *activity diagram*, *sequence diagram*, *collaboration diagram*, *component diagram*, dan *deployment diagram*

[24]. Terdapat empat diagram yang akan digunakan pada penelitian ini, yaitu use case diagram, class diagram, activity diagram, dan sequence diagram.

2.18 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem [24], [25].

2.19 Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state* diagram khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu activity diagram tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

2.20 Sequence Diagram

Sequence Diagram menggambarkan interasi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Sequence diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait) [24].

Sequence Diagram biasa digunakan untuk menggambarkan skenario atau langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk

menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan [24], [25].

2.21 Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package*, dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [24], [25].