# BAB II
# TINJAUAN PUSTAKA

## 2.1 Sistem Operasi Android

Android merupakan sebuah sistem operasi modifikasi dari linux yang digunakan untuk ponsel (*smartphone*)/Tablet hingga perangkat jam tangan sampai televisi pintar, dalam perkembanganya dalam bidang *Handphone* (*Smartphone*), Sistem operasi android sudah menjamur di kalangan masyarakat Indonesia dari anak-anak hingga dewasa. Ponsel (*smartphone*) berbasis android juga sangat berguna dalam berkomunikasi dan mencari sebuah informasi (Sifauttijani, Listyorini, & Meimaharani, 2017).

Selain itu, menurut (Maiyana, 2018) Android merupakan sebuah sistem operasi yang bersifat Open Source yaitu memberikan kebebasan bagi developer untuk mengembangkan sebuah aplikasi, dengan kelebihan dari sistem operasi android, akan banyak membantu pengguna smartphone berbasis android untuk dapat menikmati berbagai aplikasi.

## 2.2 *Location Based Service* (LBS)

*Location Based Service* (LBS) adalah sebuah layanan berbasis lokasi yang bisa mengidentifikasi objek tertentu dan menampilkan posisi lokasinya. LBS dapat diakses pada perangkat *mobile* dengan media internet. Layanan LBS menitikberatkan posisi pengguna dengan cara memanfaatkan posisi sel jaringan atau dengan teknologi *Global Positioning System* (GPS). LBS menggunakan koordinat *latitude* dan *longitude* dalam menentukan titik lokasi pengguna. Metode dalam penelitian ini menggunakan LBS dengan *Google Maps* API (Mertha, Simadiputra, Setyawan, & Suharjito, 2019).

Android memiliki bermacam-macam fitur diantaranya seperti kamera, internet, MMS, *Global Positioning System* (GPS) termasuk teknologi LBS (*Location Based Service*) dan lain-lainnya. Fitur LBS merupakan salah satu fitur android yang digunakan untuk memvisualisasikan teknologi untuk menentukan lokasi yang ingin diketahui oleh pengguna dengan memanfaatkan fasilitas satelit. LBS memiliki unsur utama dalam penggunaannya yaitu *location manager* yang menyediakan fasilitas untuk menampilkan peta dan *location providers* dimana

unsur ini digunakan untuk melakukan pencarian titik lokasi tempat (Putra, Pranatawijaya, & Sari, 2020).

Dalam layanan berbasis lokasi terdapat lima komponen penting seperti terlihat pada **gambar 2.1.**


**Gambar 2. 1** Komponen Dasar LBS (Steiniger, 2006)

Setiap komponen dalam layanan berbasis lokasi mempunyai masing-masing fungsi yaitu: (Steiniger, 2006)

1. *Mobile device*, merupakan suatu alat yang digunakan oleh pengguna untuk meminta informasi yang dibutuhkan. Informasi yang diberikan dalam bentuk suara, gambar dan *text*.
2. *Communication network*, komponen ini mengirim data pengguna dan informasi yang diminta dari *mobile* terminal ke *service provider* kemudian mengirimkan kembali informasi yang diminta kepengguna. *Communication network* dapat berupa jaringan seluler (GSM,CDMA), *Wireless Local Area Network* (WLAN), atau *Wireless Wide Area Network* (WWAN).
3. *Positioning Component*, digunakan untuk memperoses suatu layanan maka posisi pengguna harus diketahui.
4. *Service and Aplication Provider*, penyedia layanan menawarkan berbagai macam layanan kepada pengguna dan bertanggungjawab untuk memproses informasi yang diminta oleh pengguna.
5. *Data Content Provider*, penyedia layanan menyimpan semua data yang dibutuhkan yang dibutuhkan yang bisa diakses oleh pengguna.

## 2.3 *Global Positioning System* (GPS)

GPS (*Global Positioning System*) dapat berfungsi untuk memberikan informasi secara *realtime* di mana dan bagaimana kondisi sebuah objek dan GPS ini memiliki fasilitas GPS *Photo Tagging* atau *geotagging* yang berfungsi untuk mengetahui informasi posisi data GPS seperti *Latitude, Longitude, Altitude* dalam sebuah foto digital. GPS *Photo Tagging* atau biasa dikenal geotagging merupakan gabungan fitur kamera yang dapat melakukan sinergi langsung dengan fitur GPS (*Global Positioning System*) guna memberikan informasi secara *realtime* di mana dan bagaimana kondisi sebuah objek. Ponsel yang dilengkapi dengan fasilitas *geotagging* dapat digunakan untuk menghasilkan foto yang menyimpan informasi posisi data GPS, seperti garis lintang dan bujur, ketinggian, bantalan, jarak, akurasi data, dan nama tempat. Banyak ponsel yang dilengkapi dengan fitur GPS dapat menambahkan informasi lokasi kedalam metadata foto, atau biasa disebut dengan data EXIF, secara otomatis sehingga lokasi foto tersebut dapat ditampilkan ke dalam peta (Mardani, 2014).

## 2.4 *Geotagging*

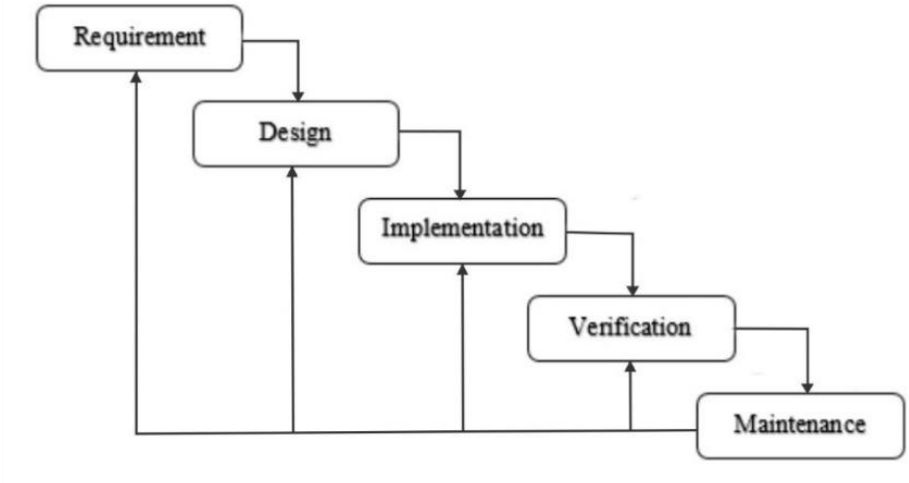
*Geotag* atau *Geotagging* merupakan suatu proses penambahan informasi geografis pada berbagai macam media, seperti foto, video, *website*, dan jejaring sosial. Dasar dari *Geotagging* adalah posisi. Posisi tersebut berasal dari *Global Positioning System* (GPS) , dan berdasarkan lintang atau bujur sistem koordinat yang menyajikan setiap lokasi di bumi dari 180° BB hingga 180° BT sepanjang Khatulistiwa dan 90° utara melalui 90° selatan sepanjang meridian utama . Ada dua pilihan utama untuk foto *geotagging*, yaitu menangkap informasi GPS pada saat foto diambil atau melampirkan foto untuk memetakan setelah gambar diambil. *Geotagging* adalah sebuah proses penambahan informasi posisi data pada GPS berupa informasi *latitude* dan *longitude* dalam sebuah foto digital. Dengan adanya fitur *geotagging* dalam informasi sebuah foto maka letak pengambilan foto tersebut dapat dengan mudah diketahui (Nandipati, 2011).

Untuk melakukan *Geotagging* dengan media foto diperlukan dua perangkat sekaligus, yaitu perangkat sensor kamera dan perangkat GPS (*Global positioning*

*system*). Sensor kamera digunakan untuk merekam data foto dan GPS digunakan untuk merekam posisi koordinat lintang dan bujur titik exposure foto.

## 2.5 Metode *Waterfall*

Metode air terjun atau disebut metode *waterfall* sering dinamakan siklus hidup klasik *(classic life cycle),* dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan *Requirement, Design System, Implementation, Verification, Maintenance* (Pressman, 2015).




**Gambar 2. 2** Tahapan Metode *waterfall* (Pressman, 2015)

1. *Requirement*

Tahap ini pengembang sistem diperlukan komunikasi yang bertujuan untuk memahami perangkat lunak yang diharapkan oleh pengguna dan batasan perangkat lunak tersebut. Informasi ini biasanya dapat diperoleh melalui wawancara, diskusi atau survey langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

2. *Design*

Spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam fase ini dan dibuat menjadi rancangan sistem. Sehingga terlihat gambaran bentuk sistem yang akan dibuat. Desain dapat berupa bagan (*chart*) yang menunjukkan desain proses bisnis, desain basis data, rancangan antarmuka dan prosedur sistem secara logika.

3. *Implementation*

Pada tahap ini merupakan implementasi dari tahap design. Gambaran sistem di tahap sebelumnya di implementasikan dalam bentuk *coding* atau kode-kode program sampai menghasilkan sistem akhir atau sistem jadi.

4. *Verification*

Tahapan ini merupakan tahap dimana sistem atau aplikasi yang telah dibuat di uji. Apakah sistem tersebut layak atau masih harus diperbaiki. Pengujian biasaya dilakukan untuk melihat apakah sistem masih error atau bahkan tidak jalan. Apabila sistem dianggap sudah baik maka sistem itu bisa diterapkan.

5. *Maintenance*

Tahap akhir dalam metode waterfall. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

## 2.6 Alat Bantu Perancangan Sistem

Menurut (A.S. & Shalahuddin, 2018), *Unified Modelling Language* (UML) adalah salah satu standar bahasa yang banyak digunakan didunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

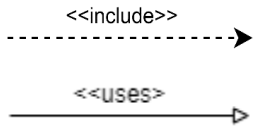
*Unified Modelling Language* (UML) terbagi ke dalam 3 (tiga) kategori yaitu diagram struktur *(structure diagrams)*, diagram kelakuan sistem *(behavior diagrams)*, dan diagram interaksi sistem *(interaction diagrams)*.

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antara subsistem pada suatu sistem.

### 2.6.1 *Use Case Diagram*

*Use Case Diagram* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (A.S. & Shalahuddin, 2018). Adapun simbol dari *use case* diagram dijelaskan pada Tabel 2.1

**Tabel 2. 1** Simbol *use case diagram (A.S. & Shalahuddin, 2018)*

| No | Simbol | Deskripsi |
|---|---|---|
| 1. | *Use Case*<br>  | Fungsionalitas yang disediakan dalam sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Biasanya dinyatakan dengan menggunakan kata kerja di awal frase name *use case.* |
| 2. | Menggunakan/*include*/*uses*<br>  | Relasi *use case* tambahan ke sebuah *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case ini*. |
| 3. | Aktor/*actor*<br>  | Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat. Simbol dari aktor adalah orang tetapi aktor belum tentu merupakan orang, biasanya menggunakan kata benda di awal frase nama aktor. |

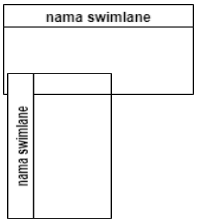
| No | Simbol | Deskripsi |
|---|---|---|
| 4. | Ekstensi/*extend*<br><<extend>><br>- - - - - - - - - - - - - - - - - -➤ | Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu. |
| 5. | Generalisasi/*generalization*<br>——————————▷ | Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum digunakan dari lainnya |
| **No** | **Simbol** | **Deskripsi** |
| 6. | Asosiasi/*association*<br>—————————— | Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor. |

### 2.6.2 *Activity Diagram*

*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (A.S. & Shalahuddin, 2018). Adapun simbol dari *Activity Diagram* dijelaskan pada Tabel 2.2

**Tabel 2. 2** Simbol *activity diagram (A.S. & Shalahuddin, 2018)*

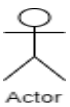
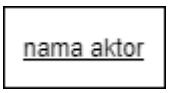
| No | Simbol | Deskripsi |
|---|---|---|
| 1. | Status awal<br>● | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal. |
| 2. | Aktivitas<br>aktivitas | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja. |
| 3. | Percabangan/*decision*<br>◇ | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |

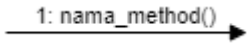
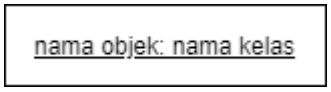
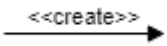
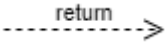
| No | Simbol | Deskripsi |
|---|---|---|
| 4. | Penggabungan/*join*  | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
| 5. | Status akhir  | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir. |
| 6. | *Swimlane*   atau | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |

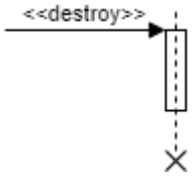
### 2.6.3 *Sequence Diagram*

*Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case* (A.S. & Shalahuddin, 2018). Berikut ini adalah simbol-simbol dari *sequence diagram* dijelaskan pada Tabel 2.3

**Tabel 2. 3** Simbol *sequence diagram (A.S. & Shalahuddin, 2018)*

| No | Simbol | Deskripsi |
|---|---|---|
| 1. | Aktor   atau   tanpa waktu aktif | Orang,proses,atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri, jadi walaupun simbol dari aktor adalah orang, tapi aktor belum tentu merupakan orang. Biasanya dinyatakan menggunakan kata benda di awal frase nama aktor. |

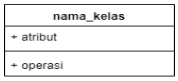
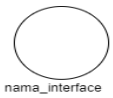
| No | Simbol | Deskripsi |
|---|---|---|
| 2. | Pesan tipe *call*<br>  | Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode. maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan objek yang berinteraksi. |
| 3. | Objek<br>  | Menyatakan objek yang berinteraksi |
| 4. | Waktu aktif<br>  | Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung<br>dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya. |
| 5. | Pesan tipe *create*<br>  | Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat. |
| 6. | Garis hidup / *lifeline*<br> | Menyatakan kehidupan suatu objek. |
| 7. | Pesan tipe *send*<br> | Menyatakan suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi. |
| 8. | Pesan tipe *return*<br>  | Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian. |

| No | Simbol | Deskripsi |
|---|---|---|
| 9. | Pesan tipe *destroy*   | Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy* |

### 2.6.4 *Class Diagram*

*Class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metodse atau operasi (A.S. & Shalahuddin, 2018). Berikut ini adalah *class* diagram sistem informasi pusat karir yang akan dibangun ditunjukan pada Tabel 2.4

**Tabel 2. 4** Simbol *classs diagram* (A.S. & Shalahuddin, 2018)

| No | Simbol | Deskripsi |
|---|---|---|
| 1. | Kelas   | Kelas pada struktur sistem. |
| 2. | Antarmuka/*interface*   | Sama dengan konsep *interface* dalam pemrograman berorientasi objek |
| 3. | Asosiasi/*association*  | Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity* |
| 4. | Asosiasi berarah/*directed/association*  | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*. |
| 5. | Generalisasi  | Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus) |

| No | Simbol | Deskripsi |
|---|---|---|
| 6. | Kebergantungan/*dependency*  | Kebergantungan antar kelas |
| 7. | Agregasi / *aggregation*  | Relasi antar kelas dengan makna semua-bagian *(whole-part).* |

## 2.7 *User Interface* (UI)

*User interface* merupakan serangkaian tampilan grafis yang dapat dimengerti oleh pengguna komputer dan diprogram sedemikian rupa sehingga dapat terbaca oleh sistem operasi komputer dan beroperasi sebagaimana mestinya. *User interface* adalah salah satu faktor yang menentukan peningkatan *traffic* pada sebuah *website*. Karena *user* berinteraksi dengan logika pemrograman melalui *user interface*. Dan desain *user interface* sendiri menjadi sangat penting mengingat semakin efektif dan efisien suatu desain, makin betah pula *user* untuk berlama lama di *website* tersebut (Agarina, Sutedi, & Karim, 2019).

## 2.8 Bahasa Pemrograman JavaScript

*JavaScript* merupakan bahasa pemrograman tingkat tinggi yang diperkenalkan pertama kali tahun 1995 untuk lingkungan pengembangan aplikasi berbasis *web* dinamis di sisi *client*, yang memungkinkan pengembang untuk mengembangkan *website* dengan tampilan menarik. *JavaScript* mengalami peningkatan kemampuan yang pesat, antara lain berupa: dukungan terhadap pemrograman berorientasi objek, pemrograman fungsional, struktural, prosedural, *event-driven*, *prototyping*, serta kemudahan pengembangan aplikasi di sisi server (Pratama, 2020).

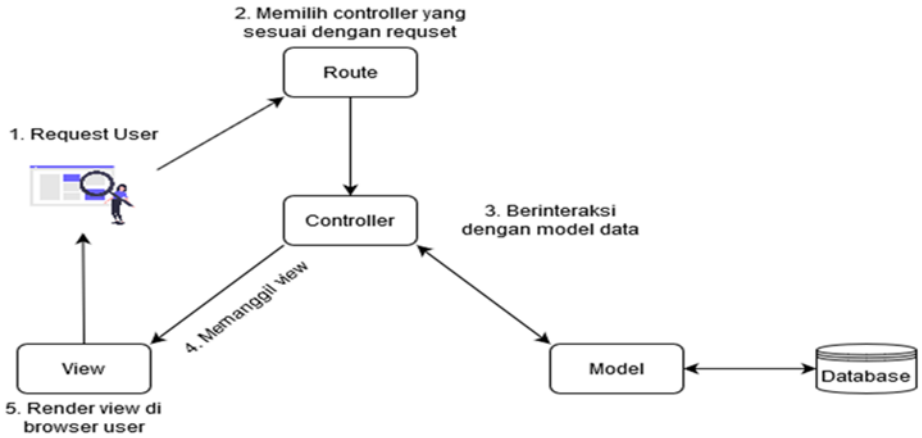
## 2.9 Framework React Native

*React native* merupakan kerangka kerja yang bersifat *open source*, dibangun oleh platform *facebook* yang dapat melakukan pengembangan untuk aplikasi *mobile* android maupun IOS. *React native* memiliki deskripsi di situs resminya "*Learn once, write anywhere*". Cara kerja *React native* menggunakan konsep

*bridge* dimana pengguna akan membangun aplikasi menggunakan kode *react* untuk membuat antarmuka aplikasi, kemudian kode *react* akan diinterpretasikan menjadi Bahasa pemrograman *javascript* dan dapat digunakan untuk aplikasi *mobile*. Fitur *bridge* digunakan untuk mengelola dan menghubungkan co*debase native module* (IOS dan *android*) sehingga *native module* siap digunakan pada *platform* yang sudah terbentuk (Siregar & Fauzi, 2021).

## 2.10 Framework Laravel

Laravel sebuah kerangka kerja *(framework) web* gratis dengan kode sumber terbuka yang menggunakan bahasa pemrograman PHP, dirancang oleh *Taylor Otwell*, Laravel adalah sebuah MVC *web development framework* PHP yang didesain untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan dan perbaikan serta meningkatkan produktifitas pekerjaan dengan sintak yang bersih dan fungsional set yang dapat mengurangi banyak waktu untuk implementasi (Widodo and Purnomo, 2016) . Beberapa fitur Laravel adalah sistem pengemasan paket modular dengan *dependencies* manager khusus, berbagai cara untuk mengakses basis data relasional, utilitas yang membantu dalam penerapan dan pemeliharaan aplikasi, *templating engine* untuk menampilkan halaman *web* dinamis menggunakan *Blade*, dan berorientasi ke *syntactic suga*r. Laravel dapat digunakan untuk membangun aplikasi web apapun, mulai dari *web* sederhana hingga aplikasi level *enterprise* (Korop, 2018).




**Gambar 2. 3** Konsep mvc *framework laravel*

1. *Model*

   *Model* mewakili struktur data. Biasanya *model* berisi fungsi-fungsi yang

membantu seseorang dalam pengelolaan basis data. Berikut adalah contoh kode untuk *model* pada *framework laravel.*

**Kode Program 2. 1** Contoh *model framework laravel*

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * The table associated with the model.
     *
     * @var string
     */
    protected $table = 'users';
}
```

2. *Controller*

*Controller* merupakan bagian yang menjembatani *model* dan *view.* Berikut adalah contoh kode untuk *controller* pada *framework* laravel.

**Kode Program 2. 2** Contoh *controller framework laravel*

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\User;

class UserController extends Controller
{
    /**
     * Show the profile for the given user.
     *
     * @param  int  $id
     * @return View
     */
    public function show($id)
    {
        return view('user.profile', ['user' => User::findOrFail($id)]);
    }
}
```

3. *View*

*View* adalah bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web. Berikut adalah contoh kode program *view* pada *framework* laravel.

**Kode Program 2. 3** Contoh *view framework laravel*

```html
<html>
    <body>
        <h1>Hello, {{ $name }}</h1>
    </body>
</html>
```

4. *Route*

*Route* merupakan pendaftaran setap *URL* yang akan diakses oleh pengguna. Berikut adalah contoh kode program *route* pada *framework.*

**Kode Program 2. 4** Contoh *route framework laravel*

```php
<?php
    Route::get('user/{id}', 'UserController@show');
```

Berdasarkan situs resmi laravel versi yang rilis saat ini adalah laravel versi 8 dan pada versi 8 ini melanjutkan peningkatan yang dibuat di Laravel 7.x dengan memperkenalkan versi laravel *jetstream*, *model factory classes*, *migration squashing*, pengelompokan pengerjaan, pembatasan tingkat yang ditingkatkan, peningkatan antrian, komponen Blade dinamis, tampilan paginasi Tailwind, pembantu pengujian waktu, peningkatan artisan serve, pendengar acara perbaikan, dan berbagai perbaikan bug lainnya dan peningkatan kegunaan (Otwell, 2020). Laravel berada di bawah lisensi *MIT License* dengan menggunakan *Github* sebagai tempat berbagi *code* menjalankannya (Mediana & Nurhidayat, 2018).

## 2.11 Basis Data *(Database)*

Sistem basis data adalah sistem yang terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan (A.S. & Shalahuddin, 2018). Dalam pengembangan sebuah perangkat lunak pasti ada yang namanya basis data, basis data menjadi peran penting untuk menyimpan suatu data, dengan adanya basis data pengguna dapat mengakses data secara mudah dan cepat. Secara teori Basis Data adalah suatu sistem yang memproses *input* berupa data menjadi *output* yaitu informasi yang diinginkan. Sedangkan menurut fungsinya adalah himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.

## 2.12 MySQL

Dikutip dari laman resmi MySQL, MySQL merupakan sistem manajemen basis data relasional (RDBMS) yang bersifat *open source* yang paling populer dikembangkan, didistribusikan dan didukung oleh *Oracle Corporation* (Oracle, 2021). MySQL menggunakan arsitektur *client-server*, di mana yang bertugas untuk memanipulasi basis data adalah *server*. Sedangkan *client* digunakan untuk berkomunikasi dengan *server* dengan menggunakan perintah yang ditulis dalam bahasa SQL (*Structured Query Language*). Pernyataan *(statement)* SQL dapat digolongkan atas dua golongan, yaitu:

1. *Data Definition Language* (DDL) yang mendefinisikan struktur data. Perintah-perintah SQL yang termasuk DDL antara lain *create, alter,* dan *drop.*
2. *Data Manipulation Language* (DML) yang mencari *query* dan mengubah *(modify)* suatu tabel. Perintah-perintah SQL yang termasuk DML antara lain *select, insert, update,* dan *delete*.

Dalam MySQL, sebuah *database* terdiri dari satu atau lebih tabel, dan setiap tabel terdiri dari baris dan kolom. Dalam bahasa SQL, pada umumnya informasi tersimpan dalam tabel-tabel yang secara logika merupakan struktur dua dimensi yang terdiri atas baris-baris data yang berada dalam satu atau lebih kolom.

## 2.13 *Application Programming Interface* (API)

API merupakan suatu dokumentasi yang terdiri dari *interface*, fungsi, kelas, struktur dan sebagainya untuk membangun sebuah perangkat lunak. Dengan adanya API ini, maka memudahkan *programmer* untuk membongkar suatu *software* untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya yang memungkinkan *programmer* menggunakan *system function*. *Google* menyediakan layanan *Google Maps* API yang memungkinkan para pengembang untuk mengintegrasikan *GoogleMaps* ke dalam *website* masing-masing dengan menambahkan data point sendiri. *Google Maps* dapat muncul di *website* tertentu menggunakan suatu API *key*. API *key* merupakan kode unik yang digenerasikan oleh *Google* untuk suatu *website* tertentu, agar dapat dikenali server *Google Maps* (Davis, 2006).

## 2.14 Aplikasi Postman

*Postman* adalah *platform* API yang digunakan untuk membangun dan menggunakan sebuah API. Postman menyederhanakan setiap langkah dari siklus hidup API dan kolaborasi sehingga dapat menggunakan API secara cepat dan lebih baik. *Postman* merupakan sebuah platform kolaborasi untuk pengembangan API. Beberapa hal yang dapat dilakukan oleh *Postman* diantaranya adalah dapat bertindak sebagai *client* yang mengakses *REST* secara langsung, pengujian yang terotomatisasi, simulasi *endpoint* secara langsung, dokumentasi API, pemantauan

performa dan waktu respon dari API. API menyediakan konteks berbagi dalam *workspace* dalam membangun dan menggunakan API secara *real-time* (Postman, 2022).

## 2.15 Metode Pengujian Perangkat Lunak

### 2.15.1 *Black Box Testing*

*Black box testing* berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh set kondisi *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program (Pressman, 2015). *Black box* testing juga disebut *functional testing*, sebuah teknik pengujian fungsional yang merancang *test case* berdasarkan informasi dari spesifikasi (Nidhra & Dondeti, 2012).

Pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Metode *Blackbox Testing* merupakan salah satu metode yang mudah digunakan karena hanya memerlukan batas bawah dan batas atas dari data yang diharapkan, estimasi banyaknya data uji dapat dihitung melalui banyaknya *field* data *entry* yang akan diuji, aturan entri yang harus dipenuhi serta kasus batas atas dan batas bawah yang memenuhi. Dan dengan metode ini dapat diketahui jika fungsionalitas masih dapat menerima masukan data yang tidak diharapkan maka menyebabkan data yang disimpan kurang valid (Cholifah, Yulianingsih, & Sagita, 2018).

*Black box testing* berusaha untuk menemukan kesalahan dalam kategori berikut :

a. Fungsi yang tidak benar atau fungsi yang hilang
b. Kesalahan antarmuka
c. Kesalahan dalam struktur data atau akses *database* eksternal
d. Kesalahan perilaku (*behavior*) atau kesalahan kinerja
e. Inisialisasi dan pemutusan kesalahan

Kategori error yang akan diketahui melalui *black box testing* :

a. Fungsi yang hilang atau tak benar
b. Error dari antarmuka
c. Error dari struktur data atau akses eksternal database

d. Error dari kinerja atau tingkah laku
e. Error dari inisialisasi dan terminasi

### 2.15.2 *User Acceptance Test* (UAT)

Pengujian UAT atau Uji Penerimaan Pengguna adalah suatu proses pengujian oleh pengguna yang dimaksudkan untuk menghasilkan dokumen yang dijadikan bukti bahwa *software* yang telah dikembangkan telah dapat diterima oleh pengguna, apabila hasil pengujian (*testing*) sudah bisa dianggap memenuhi kebutuhan dari pengguna.

*User Acceptance Testing* merupakan pengujian yang dilakukan oleh *end-user* dimana *user* tersebut adalah *staff*/karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya (Perry, 2006).

Menurut (Sugiyono, 2011) Skala *Likert* digunakan untuk mengukur sikap, pendapat atau presepsi seseorang atau kelompok terhadap sesuatu, dalam hal pendapat pengguna terhadap perangkat lunak yang dikembangkan. Data hasil kuesioner yang berupa jawaban-jawaban pengguna terhadap setiap item pertanyaan dalam kuesioner mempunyai gradasi nilai dari sangat positif sampai sangat negatif. Kuesioner sendiri terdiri dari beberapa sampel pernyataan dimana setiap pertanyaan diberi lima pilihan jawaban, yaitu :

a. SB (Sangat Baik) diberi poin 5.
b. B (Baik) diberi poin 4.
c. CB (Cukup Baik) diberi poin 3.
d. KB (Kurang Baik) diberi poin 2.
e. TB ( Tidak Baik) diberi poin 1.

Data yang dihasilkan dari kuesioner tersebut merupakan gambaran pendapat atau persepsi pengguna perangkat lunak, dalam hal ini yang berkaitan dengan faktor kualitas *usability* perangkat lunak yang dikembangkan. Data yang dihasilkan dari kuesioner merupakan data yang bersifat kualititatif. Data tersebut dapat dikonversi ke data kuantatif dalam bentuk data interval atau rasio menggunakan Skala Likert.

Maka dari definisi tersebut, dapat dikatakan bahwa UAT merupakan pengujian yang dilakukan oleh pengguna dari sistem tersebut untuk memastikan fungsi-fungsi yang ada pada sistem tersebut telah berjalan dengan baik dan sesuai

dengan kebutuhan pengguna. Proses dalam UAT adalah pemeriksaan dan pengujian terhadap hasil pekerjaan. Diperiksa apakah item-item yang ada dalam dokumen *requirement* sudah ada dalam *software* yang diuji atau tidak. Diuji apakah semua item yang telah ada telah dapat memenuhi kebutuhan penggunanya.

## 2.16 Kajian Terkait

Penelitian dilakukan berdasarkan pada penelitian yang pernah dilakukan sebelumnya. Berikut ini adalah penelitian yang serupa dengan penelitian yang akan dilakukan:

Penelitian yang dilakukan oleh (DP, Hartiningtyas, Salsabila, & Al-Haswan, 2021) yang berjudul "Aplikasi *Adopt Me*". Tujuan dari penelitian ini adalah terciptanya suatu aplikasi berbasis Android yang bertujuan untuk memudahkan para pecinta hewan dalam mencari hewan peliharaannya sekaligus melindungi hewan peliharaan agar tidak terlantar ketika pemiliknya tidak ingin mengurusnya lagi.

Penelitian yang dilakukan oleh (Nurzaman & Settiawan, 2018) yang berjudul "Implementasi Teknologi *Geotagging* Pada Aplikasi Pertolongan Kecelakaan Lalu Lintas". Aplikasi yang dibuat bernama *photo tagging* untuk pertolongan kecelakaan lalu lintas menggunakan teknologi *geotagging* pada platform android. Dengan pembangunan aplikasi ini diharapkan menjadi solusi yang tepat untuk mengurangi kesalahan memberikan lokasi korban kecelakaan, memberikan gambaran lebih mengenai keadaan korban dan membantu petugas menentukan puskesmas atau rumah sakit mana yang siap dan paling dekat dengan korban kecelakaan.

Penelitian yang dilakukan oleh (Taryadi, 2016) yang berjudul "Aplikasi Pencarian Tempat Wisata Kuliner Di Kota Pekalongan Berbasis *Location Based Service* Dan *Geotagging* Pada Android". Penelitian menghasilkan suatu aplikasi Android untuk pencarian tempat wisata kuliner di area Kota Pekalongan berbasis LBS dan *geotagging*. Aplikasi ini mampu memberikan informasi tempat wisata kuliner beserta menu dan penunjuk jalannya, serta dapat melakukan penambahan data. Hasil pengujian menggunakan metode black-box menunjukkan bahwa semua fungsi dapat bekerja dengan benar. Adapun hasil evaluasi usability mendapatkan nilai sebesar 84.78%, hal ini menunjukkan bahwa aplikasi Android yang telah

dibangun sangat mudah dipahami dan dimengerti oleh pengguna.

Berikut ini adalah penelitian yang dilakukan seperti pada Tabel 2.5

**Tabel 2. 5** Penelitian yang dilakukan

| Penulis | Judul | Keterangan |
|---|---|---|
| Yordan Maranatha Sinulingga | Aplikasi *Mobile* Android Adopsi Hewan Peliharaan Dengan Fitur Pencarian Berbasis *Location Based Service* | 1. Aplikasi berbasis *mobile* yang berjalan di sistem operasi android<br>2. Menampilkan informasi hewan lepas adopsi dan hewan hilang<br>3. Dapat melakukan pembuatan postingan hewan lepas adopsi dan hewan hilang<br>4. Dapat mengambil data lokasi dari foto postingan secara otomatis menggunakan *geotaging*<br>5. Dapat memberikan informasi lokasi ataupun mengirim pesan ke pemilik hewan hilang<br>6. Dapat melakukan permintaan pindah adopsi kepada pemilik hewan lepas adopsi<br>7. Dapat menampilkan hasil rute lokasi pengguna menuju lokasi pemilik hewan<br>8. Dapat menampilkan hasil rute lokasi menuju lokasi *shelter* terdekat ataupun *shelter* yang masih buka jam operasionalnya<br>9. Dapat melakukan pengaduan postingan yang dinilai tidak pantas |