# BAB II

# TINJAUAN PUSTAKA DAN DASAR TEORI

## 2.1 Tinjauan Pustaka

Tinjauan pustaka ini terdapat beberapa referensi yang dapat dijadikan acuan pada penelitian ini.

Penelitian yang dilakukan oleh Sitorus (2019), bertujuan membuat *Bot* menggunakan metode *webhook* untuk pemesanan rental mobil di Kuy *Tour* Yogyakarta dengan memanfaatkan Telegram. Sedangkan penelitian yang dilakukan oleh Nufusula dan Susanto (2018), bertujuan membuat *Chat Bot* untuk server pulsa menggunakan Telegram API. Aplikasi ini berguna untuk menggantikan SMS dengan *Bot* Telegram untuk isi ulang pulsa prabayar.

Penelitian yang dilakukan oleh Hamid (2019), bertujuan untuk membantu pemilihan *framework* untuk membangun aplikasi inventaris HMJ TI STMIK Akakom Yogyakarta. Sedangkan penelitian yang dilakukan oleh Rumpaidus (2019), bertujuan membantu staf tata usaha, dan guru di SMA N 2 untuk mengelola data nilai siswa. Aplikasi ini di bangun menggunakan *framework* Laravel yang menggunakan pola *Model View Controller* (MVC).

Pada penelitian ini, aplikasi pengelolaan keuangan di bangun agar dapat membantu pengurus mengelola keuangan HMJ TI STMIK Akakom secara komputerisasi dan *online* serta menambahkan fitur otomatisasi notifikasi memudahkan pengurus dalam mengingat tenggat waktu membayar uang kas.

Perbandingan antara beberapa penelitian yang digunakan sebagai tinjauan pustaka bisa dilihat pada Tabel 2.1.

**Tabel 2.1 Perbandingan Penelitian**

| Parameter / Peneliti | Objek | Metode/ Teknologi/ *Framework* | Studi Kasus | Hasil |
|---|---|---|---|---|
| Gerhard Felix Sitorus (2019) | Pembuatan *Bot* Untuk Pemesanan Rental Mobil | *Webhook*/ Telegram *Bot* API | Kuy *Tour* Yogyakarta | Sebuah Telegram *Bot* untuk pemesanan rental mobil. |
| Junior Dantje Rumpaidus (2019) | Pengelolaan Data Siswa | Laravel dan Bootstrap | SMA Negeri 2 Kota Biak | Sebuah aplikasi pengelolaan data siswa membantu guru dan staf tata usaha melakukan pengelolaan data siswa secara efisien. |
| Muhammad Nur Hamid (2019) | Inventaris di HMJ TI STMIK Akakom | Perbandingan performa/ Laravel dan CodeIgniter | HMJ TI STMIK Akakom | Analisis dari perbandingan *framework* CodeIgniter dan *framework* Laravel membantu pemilihan *framework* aplikasi inventaris. |
| Raga Nufusula, Ajib Susanto (2018) | Rancang Bangun *Chat Bot* Pada *Server* Pulsa | *Long Polling* /Telegram *Bot* API | *Server* Isi Ulang Pulsa | Sebuah Telegram *Bot* pada *Server* Pulsa yang digunakan sebagai media transaksi pulsa. |
| Usulan Penulis | Aplikasi Pengelolaan Keuangan | *Webhook*/ Telegram *Bot* API/ Laravel | HMJ TI STMIK Akakom Yogyakarta | Sebuah aplikasi pengelolaan keuangan Bendahara berbasis web yang terintegrasi dengan Telegram *Bot* untuk membuat pengelolaan keuangan yang lebih cepat, efisien dan akurat, serta pengurus dapat lebih mengingat tenggat waktu pembayaran uang kas. |

## 2.2 Dasar Teori

### 2.2.1 *Framework*

*Framework* adalah struktur konseptual dasar yang berisi kumpulan fungsi untuk tujuan tertentu yang sudah siap digunakan, sehingga pembuatan aplikasi dapat dilakukan dengan lebih cepat karena kode programnya tidak dibuat dari awal. Beberapa alasan dari digunakan *framework* dalam membuat aplikasi adalah sebagai berikut.

1. Aplikasi akan memiliki standar pemrograman yang universal.
2. Menghindari *repetitive work*.
3. Memudahkan dalam *team work*.
4. Memudahkan dalam *maintenance* dan pengembangan aplikasi dimasa mendatang.
5. Hemat waktu dan biaya.

### 2.2.2 Laravel

Laravel adalah *framework* bahasa pemrograman *Hypertext Preprocessor* (PHP) yang ditujukan untuk pengembangan aplikasi berbasis web dengan menerapkan konsep *Model View Controller* (MVC). *Framework* ini dibuat oleh Taylor Otwell dan pertama kali dirilis pada tanggal 9 Juli 2011. Laravel berlisensi *open source* yang artinya bebas digunakan tanpa harus melakukan pembayaran. Alamat *website* resmi dari *framework* laravel adalah https://laravel.com (Anonim, 2016)

Fitur-fitur modern Laravel yang sangat membantu developer dalam membuat aplikasi adalah *Bundles, Eloquent* ORM (*Object-Relational Mapping*),

*Query Builder, Application Logic, Reverse Routing, Resource Controller, Class Auto Loading, View Composers, Blade, IoC, Containers, Migration, Database Seeding, Unit Testing, Automatic Pagination, Form request, Middleware*.

*Framework* laravel juga memiliki beberapa keunggulan sebagai berikut:

1. Menggunakan *Command Line Interface* (CLI) Artisan.
2. Menggunakan *Package manager* PHP Composer.
3. Penulisan kode program lebih singkat, mudah dimengerti, dan ekspresif. Kemudian untuk cara instalasi *framework* Laravel dapat dilakukan dengan 3 cara yaitu.
    a. Melalui *Installer* Laravel.
    b. Menggunakan *Composer* dengan mengetikan perintah *create-project*.
    c. *Download source code* Laravel secara lengkap melalui GitHub dengan alamat https://github.com/laravel/laravel/ (Otwell, 2019)

Pada 03 Maret 2020 *framework* Laravel versi 7 resmi dirilis. Fitur *framework* Laravel yang ditekankan pada penelitian ini adalah *Blade, Migration, Eloquent ORM, Resource Controller, dan Middleware.* Berikut adalah penjelasan mengenai lima fitur tersebut.

1. *Blade*

*Blade* adalah *template engine*. Pada dasarnya *Blade* adalah *view* namun dengan menggunakan *Blade* akan mempermudah untuk mengatur tampilan *website* dan menampilkan data.

Cara untuk membuat *file view Blad*e adalah dengan menambahkan ekstensi .blade.php pada *file view*. Dan cara untuk memanggil *file Blade* sama dengan cara memanggil *file view* biasa.

2. *Migration*

*Migration* adalah fitur yang menyediakan cara baru untuk membuat *database*. Dengan menggunakan *Migration* cara membuat *database* melalui *Command Line Interface* (CLI) *database* atau dengan menggunakan aplikasi *database manager* digantikan dengan menggunakan *class*. Tahapan menggunakan *Migration* adalah membuat *class* kemudian melakukan perintah *migrate* melalui *Command Line Interface* (CLI) artisan.

Keuntungan menggunakan *Migration* adalah *class* yang dibuat bisa dipakai untuk membuat *database* pada berbagai macam *Relation Database Management System* (RDBMS) yang didukung oleh Laravel. Sebagai contoh misalnya aplikasi yang digunakan selama ini menggunakan *database* MYSQL, kemudian karena alasan pengembangan aplikasi maka akan dilakukan penggantian *database* ke *PostgreSQL*. Dalam proses penggantian tersebut tidak perlu membuat *class* lagi, tinggal melakukan perintah *migrate* melalui *Command Line Interface* (CLI) artisan.

Keuntungan lain dari menggunakan *Migration* adalah semua perubahan yang dilakukan pada *database* akan disimpan pada suatu tabel. Sehingga bisa dilakukan pembatalan dengan cara memasukkan perintah *rollback* pada *database* jika melakukan perubahan yang tidak benar.

3. *Eloquent ORM*

*Eloquent ORM* adalah implementasi dari *ActiveRecord* yang digunakan untuk mengatur relasi antar tabel di *database*. Pada *Eloquent ORM* tabel direpresentasikan dalam bentuk kelas dan data yang tersimpan di dalam tabel direpresentasikan dalam bentuk objek. Relasi yang dapat diatur menggunakan *Eloquent ORM* adalah sebagai berikut.

a. *One-to-One* yaitu relasi satu ke satu. Pada relasi ini digunakan *method hasOne* dan *belongsTo*.
b. *One-to-Many* yaitu relasi banyak ke satu. Pada relasi ini digunakan *method hasMany* dan *belongsTo*.
c. *Many-to-One* yaitu relasi banyak ke satu. Pada relasi ini digunakan *method belongsTo* dan *hasMany*.
d. *Many-to-Many* yaitu relasi banyak ke banyak. Pada relasi ini digunakan *method belongsToMany*.

4. *Resource Controller*

*Resource Controller* adalah fitur yang digunakan untuk mempercepat perbuatan *controller*. Sebagai contoh misalnya ada *controller* yang menangani semua HTTP *request* terhadap data dosen, untuk membuat *controller* tersebut hanya perlu mengetikan perintah yang dapat dilihat pada gambar 2.1.

```
php artisan make:controller KasPhController
```

**Gambar 2.1 Pembuatan *Controller***

Perintah di atas akan menghasilkan *controller* KasPHController.php yang disimpan pada folder app/Http/Controllers.

Setelah membuat *controller* SiswaController.php hal yang harus dilakukan selanjutnya adalah membuat satu baris kode program pada *route*, yang dapat dilihat pada gambar 2.2

```
Route::resource('KasPH', 'KasPHController');
```

**Gambar 2.2 Pembuatan *Route***

Satu baris kode program *route* tersebut bertujuan untuk melihat, menambah, mengedit, dan menghapus data KasPH.

Jadi dapat disimpulkan dengan menggunakan fitur *Resource Controller* dapat mempercepat pembuatan *controller* serta dapat menyederhanakan *route* untuk *controller*.

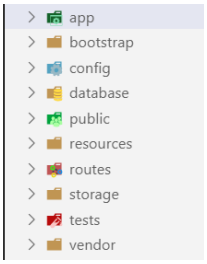
5. *Middleware*

*Middleware* adalah fitur yang menyediakan mekanisme untuk memfilter HTTP *request* yang masuk ke aplikasi. Laravel memiliki beberapa *Middleware* yaitu *Authenticate, EncryptCookies, RedirectIfAuthenticated,* dan *VerifyCsrfToken.*

Sebagai pembahasan akan dibahas *middleware Authenticate*. Middleware tersebut akan memeriksa apakah *user* sudah *login* maka *request* akan dilanjutkan ke halaman yang dikehendaki oleh *user*. Tetapi jika *user* belum *login* maka *Middleware Authenticate* akan mengarahkan *user* ke halaman *login*.

Jika *Middleware* yang sudah ada pada Laravel kurang sesuai dengan kebutuhan ataupun tidak sesuai dengan kebutuhan maka dapat dibuat sendiri *Middleware* yang sesuai dengan kebutuhan.

Struktur folder dari *framework* Laravel 7 yang masih *default* dapat dilihat pada gambar 2.1.




**Gambar 2.3 Struktur Folder Laravel 7**

a. Folder *app* adalah folder yang berisi kode program inti dari aplikasi yang akan dibuat. Model dan *Controller* tersimpan pada folder ini.
b. Folder *bootstrap* adalah folder yang berisi kode konfigurasi *autoloading* dan terdapat juga folder *cache* yang menyimpan *file-file* yang dihasilkan secara otomatis oleh Laravel untuk mengoptimasi kinerja dari sistem yang dihasilkan.
c. Folder *config* adalah folder yang berisi semua *file* konfigurasi aplikasi.
d. Folder *database* adalah folder yang berisi *file database migration* dan *seeds*.
e. Folder *public* adalah folder yang berisi *file* index.php. *File* tersebut digunakan sebagai *entry poin*t untuk menangani semua *request* yang masuk ke aplikasi. Pada folder ini juga dapat disimpan beberapa aset dari aplikasi seperti gambar, *JavaScript*, dan CSS.
f. Folder *resources* adalah folder yang berisi *file view* dari aplikasi yang dibuat. Selain itu terdapat juga *file language* yang digunakan aplikasi.

g. Folder *route* adalah folder yang berisi *file* yang digunakan untuk mendefinisikan semua *route* ke aplikasi. Secara *default* ada tiga *file route* yang disediakan Laravel yaitu api.php, console.php, dan web.php.
h. Folder storage adalah folder yang berisi *template Blade* yang dikompilasi, *file session, file cache*, dan *file* lainnya yang dihasilkan secara otomatis oleh Laravel.
i. Folder *test* adalah folder yang berisi semua *file test* yang dibuat untuk aplikasi.
j. Folder *vendor* adalah folder yang menyimpan semua *library* yang digunakan.

### 2.2.3 *Bootstrap*

*Bootstrap* adalah *framework* bahasa pemrograman *Cascade Style Sheet (CSS), Hyper Text Markup Language (HTML),* dan *Javascript* yang ditujukan untuk membuat tampilan aplikasi berbasis web menjadi responsif. Maksud responsif adalah tampilan aplikasi web akan menyesuaikan dengan ukuran layar dari perangkat yang mengaksesnya. *Framework* ini dibuat oleh Mark Otto dan Jacob Thornton. *Bootstrap* pertama kali dirilis pada tanggal 19 Agustus 2011 dan berlisensi *open source* yang artinya bebas digunakan tanpa harus melakukan pembayaran. Alamat *website* resmi dari *framework bootstrap* adalah http://getbootstrap.com (Anonim 1, 2015) Untuk mengunduh *framework bootstrap* dapat melalui GitHub dengan alamat https://github.com/twbs/bootstrap/. (Otto, 2015)

### 2.2.4 MySQL

MySQL adalah *database server* yang digunakan untuk menyimpan dan memanajemen data, dalam bahasa inggris disebut *Database Management System* (DBMS). MySQL merupakan implementasi dari sistem manajemen basis data relasional, dalam bahasa inggris disebut *Relational Database Management System* (RDBMS). Secara umum *Structured Query Language* (SQL) pada MySQL dibagi menjadi dua yaitu.

1. *Data Definition Language* (DDL) yang digunakan untuk membuat objek pada basis data seperti tabel, indeks, *sequence*, dan *view*. Yang termasuk dalam perintah DDL adalah *CREATE*, *ALTER*, dan *DROP*.
2. *Data Manipulation language* (DML) yang digunakan untuk memanipulasi objek pada basis data. Yang termasuk dalam perintah DML adalah *SELECT*, *INSERT, UPDATE*, dan *DELETE.*

*Structured Query Language* (SQL) yang dipakai pada aplikasi ini adalah *CREATE, DROP, SELECT, INSERT,UPDATE,* dan *DELETE.*

### 2.2.5 Telegram *Bot*

Telegram mempersilahkan para pengembang untuk mengembangkan aplikasinya dengan Telegram API. Ada 2 (dua) jenis API yang disediakan Telegram, API yang pertama adalah klien Telegram di mana semua orang bebas untuk membuat, memodifikasi dan mendistribusikan aplikasi pesan instannya versi mereka sendiri. Untuk hal tersebut, disediakan *source code* yang digunakan pada saat ini sehingga pengembang tidak harus membangun aplikasi Telegram dari awal.

Jenis lainnya ialah Telegram *Bot* API, API jenis kedua ini memungkinkan

pengembang untuk membuat *Bot* yang dapat membalas pesan dari semua penggunanya jika mengirimkan pesan perintah yang telah diatur dalam *Bot* itu sendiri. Layanan ini hanya tersedia bagi pengguna Telegram saja sehingga untuk dapat berkomunikasi dengan *Bot* Telegram, dibutuhkan aplikasi dan akun Telegram.

Telegram *Bot* merupakan akun khusus yang tidak memerlukan nomor telepon tambahan sebagai syarat khususnya. Akun *Bot* tersebut berfungsi sebagai antarmuka untuk kode yang dapat dijalankan pada server pengembang. *Bot* tersebut dapat melakukan beberapa pekerjaan yaitu:

a. Mengintegrasikan dengan layanan lainnya

   *Bot* dapat mengirimkan komentar jarak jauh atau mengendalikan *smart home*. Selain itu, *Bot* juga mampu mengirimkan pemberitahuan melalui Telegram ketika terjadi sesuatu di suatu tempat.

b. Menciptakan alat khusus

   *Bot* mampu memberikan pemberitahuan maupun memberikan sebuah peringatan, ramalan cuaca, terjemahan, atau layanan lain.

c. Membangun *single player* ataupun *multiplayer game*.

   Keunggulan lainnya yaitu *Bot* mampu memainkan permainan seperti catur.

d. Membangun layanan sosial

   Sebuah *Bot* dapat menghubungkan orang-orang untuk mencari mitra percakapan berdasarkan kepentingan bersama.

Dalam penggunaannya, pengembang tidak perlu repot untuk mengetahui protokol enkripsi Telegram karena hal tersebut akan ditangani oleh API Telegram.

API Telegram berupa sebuah kode autentikasi yang disebut *token*. *Token* tersebut didapatkan ketika telah melakukan pendaftaran akun pada Telegram.

Pada implementasinya, pengembang hanya memerlukan *token* sebagai syarat untuk menggunakan Telegram *Bot*. Pada Telegram *Bot* API tersedia beberapa metode dalam pengiriman pesan yaitu *getMe, sendMessage, sendDocument, sendPhoto,* dan lain-lain ("*All Method*," n.d.). Setiap metode tersebut harus memiliki parameter chat_id yang mendefinisikan identitas target obrolan.

Namun, terdapat perbedaan parameter pada setiap metode misalnya *sendMessage* wajib memiliki parameter *text* yang memiliki nilai berupa pesan yang akan dikirim. Sedangkan *sendDocument* harus memiliki parameter *document* yang berisi *file* yang akan dikirimkan. Berikut daftar perintah yang akan dibuat pada Telegram *Bot* :

a. /cara_registrasi, untuk informasi cara mendaftar ke *Bot* Telegram.

b. /cara_pesan, untuk informasi cara pemesanan mobil melalui *Bot* Telegram.

c. /daftar_mobil, untuk melihat daftar mobil yang tersedia.

d. /konfirmasi_pemesanan, untuk melakukan konfirmasi pembayaran dengan mengunggah bukti transfer melalui *Bot* Telegram.

e. /ubah_pemesanan, untuk mengubah pesanan yang telah dibuat.

f. /batalkan_pemesanan, untuk membatalkan pesanan yang telah dibuat.

g. /status_pemesanan, untuk melihat detail status pemesanan yang telah dibuat

(Anonim 2, 2015).

### 2.2.6 *Unified Modeling Language* (UML)

UML adalah bahasa untuk melakukan spesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan untuk dihasilkan oleh proses pembuatan perangkat lunak, *artifact* tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak. UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera *Rational Software Corps*.

UML digunakan untuk memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi objek. Tujuan utama UML di antaranya untuk :

a. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
b. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan praktik rekayasa.
c. Menyatukan praktik-praktik terbaik yang terdapat dalam pemodelan.

Bagian-bagian utama dari UML adalah *view*, diagram, model *element*, dan *general mechanism*.

1. *View* digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda.
2. *Use Case View* Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya.

View ini digambarkan dalam *use case* diagram dan kadang-kadang dengan *activity* diagram. View ini digunakan terutama untuk pelanggan, perancang(*designer*), pengembang(*developer*), dan penguji sistem(*tester*).

3. *Logical View* Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class*, *object*, dan *relationship*) dan kolaborasi dinamis yang terjadi ketika objek mengirim pesan ke objek lain dalam suatu fungsi tertentu. View ini digambarkan dalam *class* diagram untuk struktur statis dan dalam *state, sequence, collaboration*, dan *activity* diagram untuk model dinamisnya. View ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).
4. *Component View* Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi *administrative* lainnya. View ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).
5. *Concurrency View* Membagi sistem ke dalam proses dan prosesor. View ini digambarkan dalam diagram dinamis (*state, sequence, collaboration*, dan *activity* diagram) dan diagram implementasi (*component* dan *deployment* diagrams) serta digunakan untuk pengembang (*developer*), pengintegrasian (*integrator*), dan penguji (*tester*).
6. *Deployment View* Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. View ini digambarkan dalam *deployment* diagram dan digunakan untuk pengembang (*developer*), pengintegrasian (*integrator*), dan penguji (*tester*). (Pratama, 2019)