BAB II
TINJAUAN PUSTAKA

2.1 Studi Literatur

Pada penulisan skripsi ini, peneliti menggali informasi dari penelitian-penelitian sebelumnya sebagai bahan perbandingan, baik mengenai metode yang digunakan dan kesimpulan yang ada. Selain itu, peneliti juga menggali dari buku-buku maupun skripsi dalam rangka mendapatkan suatu informasi yang ada sebelumnya tentang teori tentang judul yang digunakan untuk memperoleh landasan teori ilmiah.

1. Penelitian yang dilakukan oleh Anggi Oktaviani (2016) ialah *E-Commerce Merchandise K-Pop* Pada Toko Haruna 88 Jakarta Menggunakan *Unified Modelling language* (UML). Dalam penelitian tersebut, peneliti membangun sebuah aplikasi *e-commerce* untuk membantu pelanggan dan petugas toko dalam melakukan pemesanan produk yang ada dalam Toko Haruna 88 Jakarta yang bertujuan untuk meningkatkan efisiensi dalam berbelanja. Dalam penelitian ini, aplikasi dibangun menggunakan bahasa pemrograman PHP dan database mysql.
2. Penelitian yang dilakukan oleh Ika Nurida Perwitasari (2013) yaitu Perancangan Sistem Informasi Penjualan Berbasis Web Pada *Cesso Shop* Yogyakarta. Pada penelitian yang dilakukan oleh Ika Nurida Perwitasari membangun Aplikasi Sistem Informasi Penjualan Berbasis Web pada *Cesso Shop* Yogyakarta, studi kasus *Cesso Shop* memasarkan produk *fashion* wanita dan kebutuhan penggemar Korean Pop (*K-Pop*) dari pabrik di Korea Selatan, Jepang dan China ke Indonesia. Dalam penelitian tersebut dilakukan analisis kelemahan sistem, analisis kebutuhan sistem, dan analisis kelayakan sistem. Penelitian tersebut menghasilkan sebuah perancangan proses, perancangan basis data, dan perancangan tampilan sistem informasi penjualan berbasis web.
3. Penelitian yang dilakukan oleh Nicky (2017) Perancangan sistem informasi Penjualan *E-Commerce* Berbasis *Web* pada Toko MM. Dalam penelitian tersebut dilaksanakan analisis sistem berjalan pada Toko MM untuk

mendapatkan kebutuhan dari sistem yang akan dikembangkan. Penelitian ini menghasilkan sebuah website e-commerce yang dibangun menggunakan CMS *Prestashop* yang terintegrasi dengan basis data.

2.2 *E-commerce*

Menurut Wong (2010:33), *E-commerce* adalah pembelian, penjualan dan pemasaran barang serta jasa melalui sistem elektronik. Seperti televisi, radio, jaringan komputer atau internet. Juga dapat didefinisikan sebagai suatu cara berbelanja atau berdagang dengan *online* atau *direct* selling yang memanfaatkan fasilitas internet dimana terdapat *website* yang menyediaakan layanan *get* and *delivery*. Ruang lingkup dalam *e-commerce* diantaranya :

1. *Bussines to Bussines* (B2B)
   Merupakan sistem komunikasi bisnis antar pelaku bisnis perusahaan atau transaksi secara elektronik antar perusahaan yang dilakukan secara rutin dan dalam kapasitas produk yang besar.
2. *Bussines to Consumer* (B2C)
   Merupakan sistem komunikasi bisnis antar pelaku bisnis dengan konsumen untuk memenuhi kebutuhan tertentu pada saat tertentu.
3. *Consumer to Consumer* (C2C)
   Merupakan sistem komunikasi dan transaksi bisnis antar konsumen untuk memenuhi kebutuhan tertentu pada saat tertentu.
4. *Consumer to Bussines* (B2C)
   Merupakan individu yang menjual produk atau jasa kepada organisasi dan individu yang mencari penjualan dan melakukan transaksi.

2.3 Aplikasi

Yuhefizar (2012) mengatakan bahwa aplikasi merupakan program yang dikembangkan untuk memenuhi kebutuhan pengguna dalam menjalankan pekerjaan tertentu. Selanjutnya menurut Pramana (2020) aplikasi merupakan suatu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas seperti perniagaan, *game*, pelayanan masyarakat, periklanan, atau semua proses yang hampir dilakukan oleh manusia. Berdasarkan dua definisi di atas, dapat diambil kesimpulan bahwa aplikasi merupakan kumpulan program yang

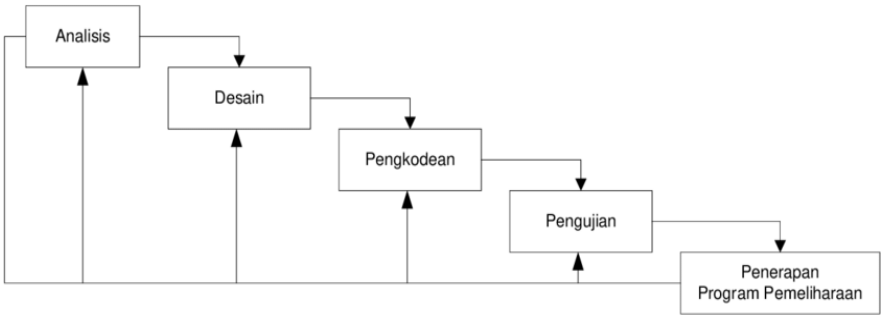
dikembangkan sesuai dengan kebutuhan pengguna untuk melayani aktivitas yang dilakukan oleh pengguna.

2.4 *Website*

Menurut (Susanti, 2016), *Website* adalah suatu kumpulan yang kompleks dalam suatu jaringan komputer yang cukup besar maupun kecil yang dapat saling berkomunikasi dengan menggunakan jaringan yang terdapat di seluruh dunia. Manusia pada umumnya dapat aktif dalam berpartisipasi, sehingga *website* tersebut dapat memberikan informasi yang tentunya akan sangat berharga bagi para penggunanya. Secara umum, *website* dapat dipahami sebagai sekumpulan halaman yang didalamnya terdapat berbagai laman yang menyediakan informasi yang luas melalui *computerized* seperti gambar, teks, maupun animasi yang disediakan oleh masing-masing *web*, sehingga dapat dengan mudah diakses oleh pengguna di seluruh dunia yang telah memiliki koneksi jaringan internet.

2.5 *Waterfall*

Model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini sebenarnya adalah "*Linear Sequential Model"*. Model ini disebut juga dengan "*Classic Life Cycle"* atau metode *waterfall*. Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winson Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan waterfall karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan (Pressman, 2015). Adapaun Fase-fase dalam *Waterfall Model* menurut referensi Pressman :



Gambar 2.1 Model *Waterfall*

Penjelasan tahapan *waterfall* tersebut yaitu:

1. Analisis

   Fase ini merupakan proses analisis terhadap aplikasi yang sedang berjalan dengan tujuan untuk mendapatkan kebutuhan sesuai dengan pengguna mengenai pengguna sistem, cara kerja sistem dan waktu penggunaan sistem, sehingga kebutuhan yang diperlukan untuk sistem baru akan didapatkan.

2. Desain

   Desain merupakan proses menentukan cara kerja aplikasi seperti perancangan antarmuka, database, dan perancangan alur program. Perancangan diperlukan untuk menggambarkan sistem baru untuk memenuhi kebutuhan *user*.

3. Pengkodean

   Tahap pengkodean yaitu tahap rancangan sistem yang dibentuk menjadi suatu kode program untuk pembuatan aplikasi.

4. Pengujian

   Pada tahap pengujian untuk mengetahui kesesuaian aplikasi apakah berjalan sesuai dengan kebutuhan dan memastikan sistem terhindar dari *error* terjadi. Pengujian juga dilakukan untuk memastikan kevalidan dalam proses *input* sehingga dapat menghasilkan *output* yang sesuai.

5. Penerapan Program Pemeliharaan

   Pada fase penerapan program dan pemeliharaan sistem yang berguna untuk melihat kemampuan aplikasi, dan mengecek jika masih ada ditemukan *error* atau ada penambahan fitur-fitur yang belum ada pada sistem tersebut. Pemeliharaan diperlukan ketika adanya perubahan dari pengguna seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.6 *Model View Controller* (MVC)

Menurut Daqiqil (2011:5) "MVC adalah singkatan dari *Model View Controller*. MVC sebenarnya adalah sebuah *pattern*/teknik pemograman yang memisahkan *bisnis logic* (alur pikir), *data logic* (penyimpanan data) dan *presentation logic* (antarmuka aplikasi) atau secara sederhana adalah memisahkan antara desain, data dan proses". Adapun komponen-komponen MVC antara lain:

1. Model

   Model berhubungan dengan data dan interaksi ke *database* atau *webservice*. Model juga merepresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks, file XML maupun *webservice*. Biasanya di dalam model akan berisi *class* dan fungsi untuk mengambil, melakukan update dan menghapus data *website*. Sebuah aplikasi *web* biasanya menggunakan basis data dalam menyimpan data, maka pada bagian model biasanya akan berhubungan dengan perintah-perintah *query SQL*.

2. *View*

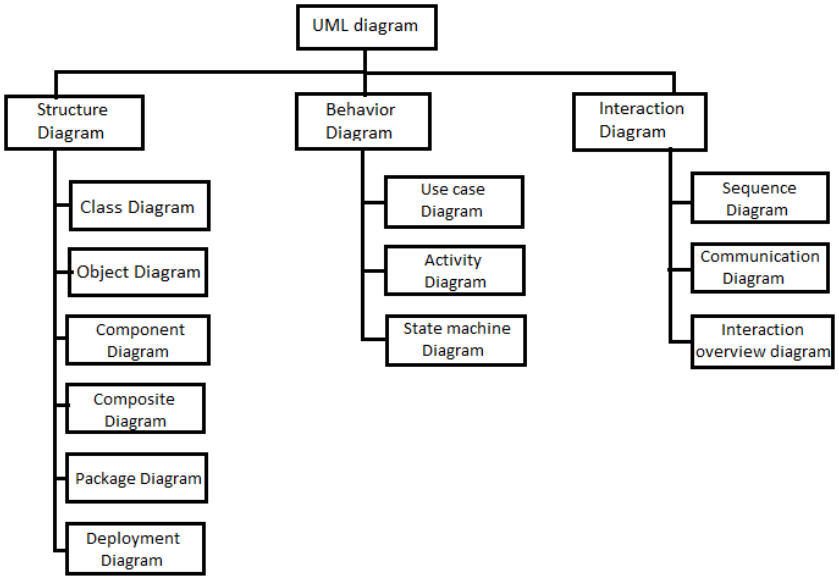
   View berhubungan dengan segala sesuatu yang akan ditampilkan ke end-user. *View* berhubungan dengan segala sesuatu yang akan ditampilkan ke end-user. Bisa berupa halaman *web*, rss, *javascript* dan lain-lain. Kita harus menghindari adanya logika atau pemrosesan data di *view*. Di dalam view hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. *View* dapat dikatakan sebagai halaman *website* yang dibuat dengan menggunakan HTML dan bantuan CSS atau *JavaScript*. Di dalam *view* jangan pernah ada kode untuk melakukan koneksi ke basis data.

3. *Controller*

   Controller bertindak sebagai penghubung data dan view. Di dalam *controller* inilah terdapat *class-class* dan fungsi-fungsi yang memproses permintaan dari view ke dalam struktur data di dalam model. *Controller* juga tidak boleh berisi kode untuk mengakses basis data karena tugas mengakses data telah diserahkan kepada model. Tugas *controller* adalah menyediakan berbagai variabel yang akan ditampilkan di *view*, memanggil model untuk melakukan akses ke basis data.

2.7 *Unified Modeling Language* (UML)

Menurut (Sukamto & Sallahuddin, 2015), dalam UML terdapat 13 diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.2 di bawah ini :



Gambar 0.1 Klasifikasi Diagram UML

Berikut penjelasan singkat dari Gambar 2.2 :

1. *Structure Diagram,* yaitu kumpulan diagram-diagram yang menggambarkan struktur statis dari sistem yang dimodelkan.
2. *Behavior Diagram*, yaitu kumpulan diagram-diagram yang menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi di dalam sistem.
3. *Interaction Diagram,* yaitu kumpulan diagram-diagram yang menggambarkan interaksi sistem dengan sistem lain ataupun interaksi antar subsistem dalam sebuah sistem.

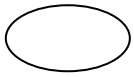
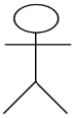
Secara garis besar, beberapa diagram utama sudah dapat menggambarkan keseluruhan sistem. Diagram tersebut antara lain *use case diagram, class diagram, sequence diagram,* dan *activity diagram*

2.7.1 Use Case

Menurut (Sukamto & Sallahuddin, 2015), diagram *use case* merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan 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. Berikut adalah simbol-simbol yang ada pada diagram *use case*.

Tabel 0.1 Simbol Diagram *Use Case*

No.	Notasi	Nama	Deskripsi
1.		Use Case	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case.
2.		Actor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
3.		Association	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.

No.	Notasi	Nama	Deskripsi
4.	<<extend>> - - - - - - - ->	Extended (Ekstensi)	Relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan.
5.	<<include>> < - - - - - - - -	Include (Menggunakan)	Relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan.

2.7.2 Class Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Sukamto & Sallahuddin, 2015). *Class diagram* memiliki apa yang disebut atribut dan operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut:

1. *Private* (-), hanya dapat digunakan oleh *class* yang memilikinya.
2. *Public* (+), dapat digunakan oleh *class* lain.
3. *Protected* (#), hanya dapat dipanggil oleh *class* yang bersangkutan jumlah suatu anak yang mewarisinya.

Nilai kardinalitas atau *multiplicity* sebuah *class* menunjukan jumlah suatu objek yang dapat berhubungan dengan objek lain. Berikut nilai kardinalitas atau *multiplicity* pada Tabel 2.2 (Tohari, 2014) dan notasi *class diagram* pada Tabel 2.3 (Sukamto & Sallahuddin, 2015).

Tabel 0.2 Jenis-Jenis *Multiplicity*

No	Indikator	Keterangan
1	0..1	Nol atau satu
2	1	Hanya satu
3	0..*	Nol atau lebih
4	1..*	Satu atau lebih

Tabel 0.3 Deskripsi Notasi Pada *Class Diagram*

No	Notasi	Nama	Deskripsi
1		*Class*	Menggambarkan konsep dasar pemodelan sistem.
2	————	Asosiasi (*Association*)	Sebuah garis solid antara dua *class*, ditarik dari *class* sumber ke *class* target lebih spesifik, digunakan dalam struktur pewarisan.
3	- - - - - - - - ->	Ketergantungan (*Dependency)*	Relasi antara dua elemen jika perubahan definisi sebuah

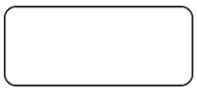
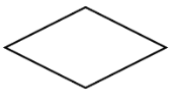
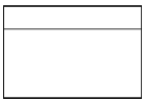
No	Notasi	Nama	Deskripsi
			elemen (*supplier* atau sumber) dapat menyebabkan perubahan pada elemen lainnya (*Client* atau target).

2.7.3 Activity Diagram

Activity Diagram menggambarkan *workflow* (aliran kerja) proses bisnis dan urutan aktivitas dalam sebuah proses. *Activity diagram* sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Membuat *activity diagram* pada awal pemodelan proses cukup menguntungkan untuk membantu memahami keseluruhan proses. *Activity diagram* juga bermanfaat untuk menggambarkan *parallel behavior* atau menggambarkan interaksi antara beberapa *use case.* Berikut merupakan simbol notasi *Activity Diagram* pada Tabel 2.4 (Sukamto & Sallahuddin, 2015).

Table 2.4 Simbol *Activity Diagram*

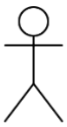
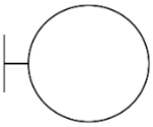
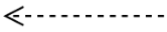
No.	Notasi	Nama	Deskripsi
1		Status Awal (*initial node)*	Status awal aktivitas, sebuah diagram aktivitas memiliki status awal.
2		Status Akhir (*final node)*	Status akhir yang dilakukan sistem.

No.	Notasi	Nama	Deskripsi
3		Aktivitas (*activity)*	Aktivitas yang dilakukan oleh sistem, biasanya di awali oleh kata kerja.
4		Percabangan (*decision)*	Asosiasi percabangan di mana jika ada pilihan aktivitas lebih dari satu.
5		Penggabungan (*join)*	Asosiasi penggabungan di mana lebih dari satu aktivitas digabungkan menjadi satu.
6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.7.4 Sequence Diagram

Menurut (Sukamto & Sallahuddin, 2015) berpendapat bahwa, *sequence diagram* menggambarkan alur objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Oleh karena itu, untuk menggambarkan *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Berikut adalah simbol-simbol yang ada pada *sequence diagram* pada Tabel 2.5.

Tabel 2.5 Simbol *Sequence Diagram*

No	Notasi	Nama	Deskripsi
1		*Actor*	Menggambarkan orang yang sedang berinteraksi dengan sistem
2		*Object*	Menyatakan objek yang berinteraksi pesan.
3		*A focus of control & A life line*	Menyatakan kehidupan suatu objek.
4		*Boundary Class*	Menggambarkan sebuah penggambaran dari *form.*
5		*Control Class*	Menggambarkan penghubung antara boundary dengan tabel.
6		*Message*	Menggambarkan pesan atau interaksi antar objek.
7		*Message to self*	Menggambarkan pesan balikan atau reaksi dari objek sebelumnya.

2.8 Teknologi Pendukung

2.8.1 *PHP: Hypertext Preprocessor*

Menurut (MADCOM, 2016) "*PHP (Hypertext Preprocessor)* adalah bahasa *script* yang dapat ditanamkan atau disisipkan ke dalam *HTML*. *PHP* banyak dipakai untuk membuat program situs *web* dinamis". *PHP* dapat digunakan dengan

gratis (*free*) dan bersifat *Open Source*. *PHP* dirilis dalam lisensi *PHP license*. Untuk membuat program *PHP* kita diharuskan untuk menginstal *web server* terlebih dahulu. *PHP* mendukung komentar seperti pada bahasa 'C', 'C++', dan *Unix shell-style*. (*Perl style*).

2.8.2 XAMPP

Menurut (MADCOM, 2016) "XAMPP adalah sebuah paket kumpulan software yang terdiri dari *Apache*, *MySQL*, *PhpMyAdmin*, *PHP*, *Perl*, *Filezilla*, dan lain." XAMPP berfungsi untuk memudahkan instalasi lingkungan PHP, di mana biasanya lingkungan pengembangan *web* memerlukan PHP, Apache, MySQL dan PhpMyAdmin.

2.8.3 *JavaScript*

JavaScript adalah bahasa *script* yang disisipkan pada kode HTML dan diproses di sisi klien yaitu sering disebut *client side*. Bahasa ini menjadikan dokumen HTML menjadi semakin luas (Agusriandi, 2018).

2.8.4 HTML

Menurut (Wardana, 2016). *Hypertext Markup Language* (HTML) merupakan bahasa pemrograman dasar untuk mengelola website. Akan tetapi HTML hanya terbatas pada pembuatan website statis (*website* yang tidak dapat berinteraksi aktif dengan user). Maka dari itu HTML biasa dikombinasikan dengan Bahasa pemrograman *web* lainnya.

2.8.5 CSS

CSS adalah suatu bahasa digunakan untuk mengendalikan dan membangun berbagai komponen dalam web sehingga tampilan *web* akan lebih rapi, terstruktur, dan seragam. CSS merupakan salah satu kode pemrograman yang bertujuan untuk menghias dan mengatur gaya tampilan atau *layout* halaman *web* agar lebih menarik. CSS adalah sebuah teknologi internet yang direkomendasikan oleh *World Wide Web Consortium* atau W3C pada tahun 1996 (Wahyudi, 2017).

2.8.6 Laravel

Menurut Aminudin (2015) Laravel adalah sebuah *Framework* PHP dirilis dibawah lisensi MIT dengan kode sumber yang sudah disediakan oleh Github, sama seperti *framework-framework* yang lain, Laravel dibangun dengan konsep MVC (*Model-Controller-View*), kemudian Laravel dilengkapi juga *command line tool* yang bernama "Artisan" yang bisa digunakan untuk *packaging bundle* dan instalasi *bundle* melalui *command prompt.*

Berikut ini beberapa fitur yang dimiliki oleh *framework* Laravel menurut Aminudin (2015) :

4. *Bundles*

   Bundles yaitu sebuah fitur dengan system pengemasan *modular* dan berbagai bundle telah tersedia untuk digunakan dalam aplikasi Anda.

5. *Eloquent ORM*

   Eloquent ORM merupakan penerapan PHP lanjutan dari pola "*active record*" menyediakan metode internal untuk mengatasi kendala hubungan antara objek *database*. Pembangun *query* Laravel *Fluent* didukung *Eloquent*.

6. *Application Logic*

   Application Logic merupakan bagian dari aplikasi yang dikembangkan, baik menggunakan *Controllers* maupun sebagai bagian dari deklarasi *Route*. Sintaks yang digunakan untuk mendefinisikannya mirip dengan yang digunakan oleh *framework* Sinatra.

7. *Reverse Routing*

   Reverse Routing mendefinisikan hubungan antara *link* dan route, sehingga jika suatu saat ada perubahan pada *route* secara otomatis akan tersambung dengan link yang relevan. Ketika link yang dibuat dengan menggunakan nama-nama dari route yang ada, secara otomatis laravel akan membuat URI yang sesuai.

8. *Restful Controllers*

   Restful Controllers memberikan sebuah *option* (pilihan) untuk memisahkan logika dalam melayani HTTP GET dan permintaan POST.

9. *Class Auto Loading*

Class Auto Loading menyediakan otomatis loading untuk *class-class* PHP, tanpa membutuhkan pemeriksaan manual terhadap jalur masuknya. Fitur ini mencegah *loading* yang tidak perlu.

10. *View Composers*

    View Composers adalah kode unit *logical* yang dapat dijalankan ketika sebuah *view* di *load*.

11. *IoC Container*

    IoC Container memungkinkan untuk objek baru yang dihasilkan dengan mengikuti prinsip *control* pembalik, dengan pilihan contoh dan referensi dari objek baru sebagai *Singletons*.

12. *Migrations*

    Migrations menyediakan versi sistem control untuk skema *database*, sehingga memungkinkan untuk menghubungkan perubahan adalah basis kode aplikasi dan keperluan yang dibutuhkan dalam merubah tata letak *database*. Mempermudah dalam penempatan dan memperbarui aplikasi.

13. *Unit Testing*

    Unit Testing mempunyai peran penting dalam *framework* Laravel, dimana *unit testing* ini mempunyai banyak tes untuk mendeteksi dan mencegah regresi. *Unit testing* dapat dijalankan melalui fitur "artisan *command-line*".

14. *Automatic Pagination*

    Automatic Pagination menyederhanakan tugas dari penerapan halaman, menggantikan penerapan yang manual dengan metode otomatis yang terintegrasi ke Laravel

2.9 Pengujian Aplikasi

Pengujian dapat menjadi alat ukur kualitas perangkat lunak yang dibutuhkan sebelum sebuah perangkat lunak tersebut dijalankan. Hal tersebut perlu dilakukan untuk membuktikan bahwa perangkat lunak tersebut sudah layak untuk digunakan dan memenuhi kebutuhan kinerja sesuai kebutuhan dari pengguna.

2.9.1 Pengujian Black Box

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 berfokus pada spesifikasi fungsional dari perangkat lunak. Tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. *Black Box Testing* bukanlah solusi alternatif dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh White Box Testing. *Black Box Testing* cenderung untuk menemukan hal-hal berikut (Mustaqbal, 2015).

1. Fungsi yang tidak benar atau tidak ada
2. Kesalahan antarmuka (*interface errors*)
3. Kesalahan pada struktur data dan akses basis data
4. Kesalahan performansi (*performance errors*)
5. Kesalahan inisialisasi dan terminasi