

II. TINJAUAN PUSTAKA

2.1 Aplikasi

Aplikasi adalah kumpulan perintah program yang dibuat untuk melakukan pekerjaan-pekerjaan tertentu (khusus) (Hendrayudi, 2009).

2.2 Pelaporan

Pelaporan adalah penyampaian perkembangan atau hasil kegiatan atau pemberian keterangan mengenai segala hal yang berkaitan dengan tugas dan fungsi-fungsi kepada pejabat yang lebih tinggi (Gullick, 2011).

2.3 Keluhan

Keluhan atau komplain adalah suatu sistem yang digunakan untuk memonitor sikap dan kepuasan para pelanggan, penyalur, dan partisipan lain dalam sistem pemasaran sehingga manajemen dapat mengambil langkah yang lebih cepat untuk menyelesaikan masalah (Tjiptono, 2008).

2.4 Pelanggan

Pelanggan adalah seorang individu atau kelompok yang membeli produk fisik ataupun jasa dengan mempertimbangkan berbagai macam faktor seperti harga, kualitas, tempat, pelayanan berdasarkan keputusan mereka sendiri (Greenberg, 2010).

Menurut Greenberg (2010), jenis-jenis pelanggan bisnis ada tiga macam yaitu :

1. Pelanggan Internal

Pelanggan internal adalah individu yang bertempat atau berlokasi di dalam perusahaan dan pada umumnya memiliki pengaruh pada kinerja perusahaan.

2. Pelanggan Antara

Pelanggan antara adalah pelanggan yang berperan sebagai perantara antara produsen dan konsumen akhir.

3. Pelanggan Eksternal

Pelanggan eksternal adalah konsumen akhir dari suatu produk dan jasa.

2.5 *Web*

Web adalah halaman informasi yang disediakan melalui jalur internet sehingga bias diakses di seluruh dunia selama terkoneksi dengan jaringan internet. *Web* adalah salah satu alat komunikasi *online* yang menggunakan media internet dalam pendistribusiannya (Prihastono, 2012).

2.5.1 *Website*

Website adalah keseluruhan halaman-halaman *web* yang terdapat dari sebuah *domain* yang mengandung informasi. Salah satu manfaat *website* adalah media untuk memperkenalkan diri atau mempromosikan institusi atau lembaga dengan menyediakan informasi yang akurat dan jelas pada *website* (Yuhefizar, 2013).

2.5.2 *Web Browser*

Web browser disebut juga sebagai perambah. *Web browser* adalah perangkat lunak yang berfungsi menampilkan dan melakukan interaksi dengan dokumen-dokumen yang disediakan oleh server *web* (Mauluddin, 2011).

2.5.3 *Web Server*

Web server merupakan perangkat lunak yang memberikan layanan data yang berfungsi menerima permintaan HTTP atau HTTPS dari klien yang dikenal dengan *browser web* dan mengirimkan kembali hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk HTML (Masruri, 2015).

2.6 *Unified Modeling Language (UML)*

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa dan Shalahuddin, 2014).

Beberapa bagian dari UML diagram yang digunakan dalam perancangan aplikasi, adalah sebagai berikut :

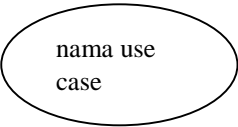


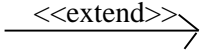
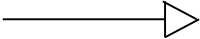
1. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat (Rosa dan Shalahuddin, 2014). Dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case* :

- a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
- b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

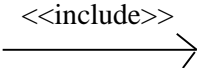
Simbol-simbol yang digunakan dalam diagram *use case* yang disajikan pada tabel 1.

Tabel 1. Simbol-simbol *use case diagram*.

Nama Simbol	Simbol	Keterangan
<i>Use Case</i>		Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama <i>use case</i> .
Aktor/ <i>actor</i>		Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
Asosiasi/ <i>association</i>		Komunikasi antara aktor dengan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor.
Ekstensi/ <i>extend</i>		Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya <i>use case</i> tambahan memiliki nama depan yang sama dengan <i>use case</i> yang ditambahkan.
Generalisasi/ <i>generalization</i>		Hubungan generalisasi-spesialisasi (umum-khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

Sumber : Rosa dan Shalahuddin (2014).

Tabel 1. (Lanjutan)

Nama Simbol	Simbol	Keterangan
<i>Include</i>		Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya.

Sumber : Rosa dan Shalahuddin (2014).




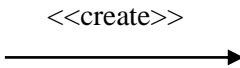
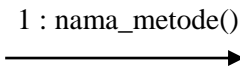
2. *Sequence Diagram*

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 *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *sequence diagram* juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

Pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence diagram* sehingga semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak (Rosa dan Shalahuddin, 2014).

Simbol-simbol yang digunakan dalam diagram *sequence diagram* yang disajikan pada tabel 2.

Tabel 2. Simbol-simbol *sequence diagram*.

Nama Simbol	Simbol	Keterangan
Actor/Aktor	 <div style="border: 1px solid black; padding: 2px; display: inline-block;">nama objek : nama kelas</div>	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
Objek		Menyatakan objek yang berinteraksi pesan.
Lifeline/garis hidup		Menyatakan kehidupan suatu objek.
Waktu aktif		Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, aktor tidak memiliki waktu aktif.
Pesan tipe <i>create</i>		Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe <i>call</i>		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 yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.

Sumber : Rosa dan Shalahuddin (2014).

3. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas

memiliki apa yang disebut atribut dan metode atau operasi (Rosa dan Shalahuddin, 2014). Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga *programmer* dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada *class diagram* sebaiknya memiliki jenis-jenis kelas berikut ini :

a. Kelas main

Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

b. Kelas yang menangani tampilan sistem (*view*)

Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

c. Kelas yang diambil dari pendefinisian *use case* (*controller*)

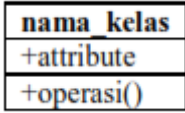
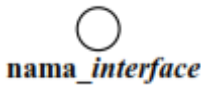

Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

d. Kelas yang diambil dari pendefinisian data (*model*)

Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut *multivalue* pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap ada di dalam perancangan kelas.

Simbol-simbol yang digunakan dalam diagram *sequence diagram* yang disajikan pada tabel 3.

Tabel 3. Simbol-simbol *class diagram*.

Nama Simbol	Simbol	Keterangan
Kelas		Kelas pada struktur sistem.
Antarmuka / <i>Interface</i>		Sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek.
Asosiasi / <i>Association</i>		Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .

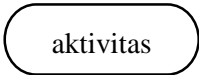

Sumber : Rosa dan Shalahuddin (2014).

4. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. *Activity diagram* menggambarkan tentang aktivitas sistem bukan menggambarkan tentang apa yang dilakukan aktor (Rosa dan Salahuddin, 2014).

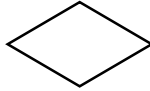


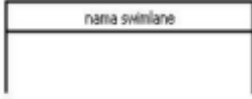
Simbol-simbol yang digunakan dalam diagram *sequence diagram* yang disajikan pada tabel 4.

Tabel 4. Simbol-simbol *activity diagram*

Nama Simbol	Simbol	Keterangan
<i>Activity</i>		Aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja.
Status awal		Status awal aktivitas sistem, sebuah <i>activity diagram</i> memiliki status awal.

Sumber : Rosa dan Shalahuddin (2014).

Tabel 4. (Lanjutan)

Nama Simbol	Simbol	Keterangan
Percabangan/ <i>decision</i>		Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan/ <i>join</i>		Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir		Sistem akhir yang dilakukan sistem, sebuah <i>activity diagram</i> memiliki status akhir.
<i>Swimlane</i>		Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Sumber : Rosa dan Shalahuddin (2014).

2.7 Metode Pengembangan Perangkat Lunak

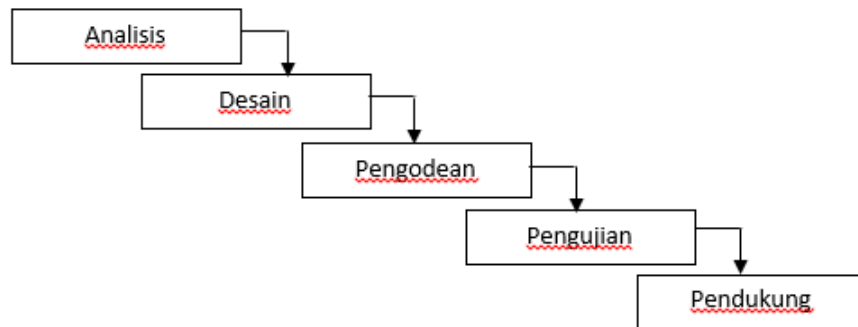
2.7.1 Model *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya berdasarkan *best practice* atau cara-cara yang sudah teruji baik (Rosa dan Salahuddin, 2014).

2.7.2 Metode *Waterfall*

Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier (*sequential linier*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekkatan alur hidup perangkat lunak secara sekuensial

atau terturut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*). Model air terjun (*waterfall*) dapat dilihat pada Gambar 1.



Gambar 1. Model air terjun (*waterfall*)

Sumber : Rosa dan Salahuddin (2014)

1. Analisis

Analisis kebutuhan perangkat lunak adalah proses pengumpulan kebutuhan dilakukan secara intensif untuk memesifikasi kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*.

2. Desain

Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean.

3. Pengkodean

Pengkodean atau pembuatan kode program desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

Pengujian fokus pada perangkat lunak dari segi logik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung

Pendukung (*support*) atau pemeliharaan (*maintenance*). Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk membuat perangkat lunak baru.

2.8 Konsep Dasar Basis Data

2.8.1 Data

Data merupakan penghubung antara *user* dengan komputer, sehingga menjadi komponen terpenting DBMS. Data adalah fakta dari fenomena fisik yang mewakili suatu objek yang disimpan dalam bentuk angka, huruf, symbol, teks, gambar dan lain sebagainya (Indrajani, 2014).

2.8.2 Basis Data

Setiap basis data dapat berisi sejumlah objek basis data (seperti tabel, *index* dan lain-lain) dan setiap basis data juga mengandung definisi struktur (Fathansyah, 2012).

Basis data merupakan sekumpulan data yang secara sistematis tersimpan di dalam komputer tanpa pengulangan (*redudansi*) yang saling berhubungan satu sama lain (Hutahaean J. , 2014).

2.8.3 Bahasa Basis Data

Bahasa basis data dikelompokkan ke dalam dua bentuk (Fathansyah, 2012), antara lain:

1. *Data Definition Language*

Bahasa khusus yang dapat menggambarkan skema basis data secara keseluruhan.

2. *Data Manipulation Language*

Bahasa yang berguna untuk melakukan manipulasi dan pengambilan data yang terdapat pada suatu *database*.

2.8.4 Sistem Basis Data

Sistem basis data merupakan suatu sistem penyusunan dan pengelolaan *record-record* menggunakan komputer, dengan tujuan untuk menyimpan serta memelihara data secara lengkap sehingga mampu menyediakan informasi yang diperlukan *user* (Lubis, 2016).

Sistem basis data memiliki beberapa komponen penting (Yanto R. , 2016), yaitu:

1. Data.
2. Perangkat keras (*Hardware*).
3. Sistem operasi (*Operating System*).
4. Basis data (*Database*).

5. Sistem pengelola basis data (*Database Management System/DBMS*).
6. Pengguna (*user*).
7. Aplikasi Lainnya.

2.9 Bahasa Pemrograman Website

2.9.1 *Hypertext Markup Language* (HTML)

Hyper Text Markup Language atau HTML adalah sebuah bahasa *markup* yang digunakan untuk membuat sebuah halaman *web* dan menampilkan berbagai informasi di dalam sebuah *browser* internet (Prayitno, 2010).

2.9.2 *Hypertext Preprocessor* (PHP)

Hypertext Preprocessor atau PHP adalah bahasa skrip yang dapat ditanamkan atau disisipkan ke dalam HTML (Anisya, 2013). PHP banyak dipakai untuk membuat situs *web* dinamis, yaitu kode PHP diselipkan diantara script kode-kode HTML yang merupakan bahasa *markup* standar untuk dunia *web* (Supardi, 2015).

2.9.3 *Cascading Style Sheet* (CSS)

Cascading Style Sheet atau CSS adalah suatu bahasa *stylesheet* yang digunakan untuk mengatur *style* suatu dokumen. Pada umumnya CSS dipakai untuk memformat tampilan halaman *web* yang dibuat dengan bahasa HTML dan XHTML (Sulistiyawan dkk, 2008).

2.9.4 *Javascript*

Javascript adalah bahasa script populer yang dipakai untuk menciptakan halaman *web* yang dapat berinteraksi dengan pengguna dan dapat merespon *event*

yang terjadi pada halaman. *Javascript* merupakan perekat yang menyatukan halaman-halaman *web* (Winarno dkk, 2014).

2.9.5 MySQL

MySQL adalah suatu perangkat lunak *database* relasi (*Relational Database Management System* atau *DBMS*), seperti halnya ORACLE, POSTGRESQL, MSSQL, dan sebagainya. SQL merupakan singkatan dari *Structure Query Language*, didefinisikan sebagai suatu sintaks perintah-perintah tertentu atau bahasa program yang digunakan untuk mengelola suatu *database* (Anisya, 2013).

2.9.6 PHPMyAdmin

PHPmyadmin adalah sebuah *software* berbasis pemrograman PHP yang dipergunakan sebagai administrator MySQL melalui *browser (web)* yang digunakan untuk manajemen *database* (Rahman, 2013).

2.9.7 XAMPP

XAMPP adalah sebuah *software web server* apache yang di dalamnya sudah tersedia *database server mysql* dan *support php programming* (Handayani, 2012).

2.10 Software Aplikasi Yang Dibutuhkan (Tools)

2.10.1 Bootstrap

Bootstrap adalah paket aplikasi siap pakai untuk membuat *front-end* sebuah *website*. *Bootstrap* adalah *template* desain *web* dengan fitur plus. *Bootstrap* diciptakan untuk mempermudah proses desain *web* bagi berbagai tingkat pengguna, mulai dari level pemula hingga yang sudah berpengalaman (Rozi, 2015).

2.10.2 *Notepad++*

Notepad++ merupakan sebuah *software* yang digunakan untuk menampilkan dan menyunting teks dan berkas kode sumber berbagai bahasa pemrograman yang berjalan di atas sistem operasi *Microsoft Windows* (Masruri, 2015).

2.10.3 *Framework*

Framework adalah kerangka kerja. *Framework* juga dapat diartikan sebagai kumpulan *script* (terutama *class* dan *function*) yang dapat membantu *developer/programmer* dalam menangani berbagai masalah-masalah dalam pemrograman seperti koneksi ke *database*, pemanggilan *variable*, dan *file*. Sehingga *developer* lebih fokus dan lebih cepat membangun aplikasi (Rosmala dkk, 2011).

2.10.4 *Model-View-Controller (MVC)*

Model-View-Controller atau MVC adalah sebuah konsep yang diperkenalkan oleh penemu *Smalltalk* (*Trygve Reenskaug*) untuk enkapsulasi data bersama dengan pemrosesan (*model*), mengisolasi dari proses manipulasi (*controller*) dan tampilan (*view*) untuk dipresentasikan pada sebuah *user interface* (Hidayat dkk, 2012).

Definisi teknis dari arsitektur *Model View Controller* (MVC) dibagi menjadi tiga lapisan (Hidayat dkk, 2012) :

1. *Model*

Model digunakan untuk mengelola informasi dan memberitahu pengamat ketika ada perubahan informasi. Hanya model yang mengandung data dan fungsi

yang berhubungan dengan pemrosesan data.

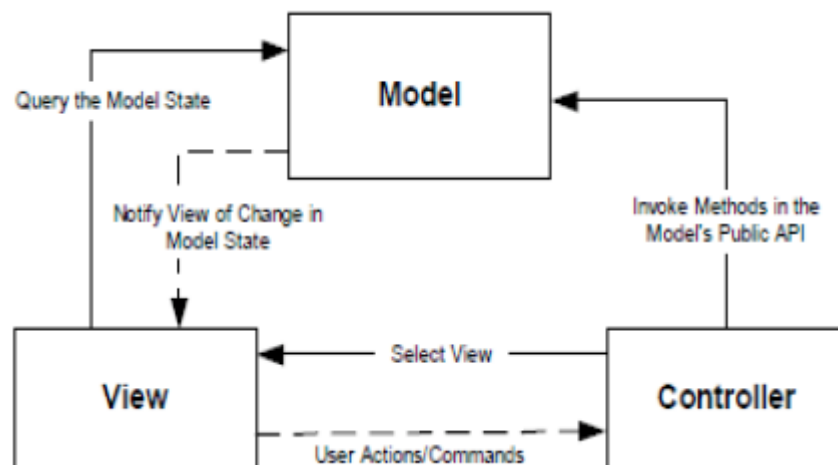
2. *View*

View bertanggung jawab untuk pemetaan grafis ke sebuah perangkat. *View* biasanya memiliki hubungan dengan sebuah permukaan layar dan tahu bagaimana untuk membuatnya. *View* melekat pada model dan me-render isinya ke permukaan layar.

3. *Controller*

Controller menerima input dari pengguna dan menginstruksikan *model* dan *view* untuk melakukan aksi berdasarkan masukan tersebut. *Controller* bertanggung jawab untuk pemetaan aksi pengguna akhir terhadap respon aplikasi.

Model, *view* dan *controller* saling berhubungan. Oleh karena itu, untuk mengilustrasikan hubungan dasar *Model-View-Controller* dapat dilihat pada Gambar 2.



Gambar 2. Hubungan *model*, *view* dan *controller*

Sumber : Hidayat dkk (2012)

Arsitektur *Model-View-Controller* atau MVC memiliki beberapa manfaat antara lain (Hidayat dkk, 2012) :

1. Memisahkan antara *model* dan *view* memungkinkan beberapa *view* menggunakan *model* yang sama.
2. Komponen *model* sebuah aplikasi lebih mudah untuk diterapkan, diuji, dan dipelihara karena semua akses ke *model* berjalan melalui komponen ini.

2.10.5 Laravel

Laravel merupakan *framework* PHP yang dibangun dengan konsep *Model-View-Controller* (MVC), sama seperti *framework* lainnya yang menggunakan konsep MVC, laravel dilengkapi dengan “*command line tool*” yang bernama “artisan” yang membantu dalam melakukan perintah-perintah pada pembangunan suatu aplikasi atau *website* (Aminudin, 2015).

2.10.6 Fitur-fitur Laravel

Laravel memiliki fitur-fitur yang mempermudah pengguna untuk membangun *website*, fitur tersebut adalah (Aminudin, 2015) :

1. *Bundles* yaitu sebuah fitur dengan sistem pengemasan modular dan tersedia beragam di aplikasi.
2. *Eloquent ORM* merupakan penerapan PHP lanjutan menyediakan metode internal dari pola “*active record*” yang mengatasi masalah pada hubungan objek *database*.
3. *Application Logic* merupakan bagian dari aplikasi, menggunakan *controller* atau bagian *route*.
4. *Reverse Routing* mendefinisikan relasi atau hubungan antara *Link* dan *Route*.
5. *Restful controllers* memisahkan logika dalam melayani HTTP *GET and*

POST.

6. *Class Auto Loading* menyediakan loading otomatis untuk *class* PHP.
7. *View Composer* adalah kode unit logikal yang dapat dieksekusi ketika view sedang *loading*.
8. *IoC Container* memungkinkan obyek baru dihasilkan dengan pembalikan controller.
9. *Migration* menyediakan sistem kontrol untuk skema database.
10. *Unit Testing* banyak tes untuk mendeteksi dan mencegah regresi.
11. *Automatic Pagination* menyederhanakan tugas dari penerapan halaman.

2.11 Object Oriented Programming (OOP)

Pemrograman berorientasi objek atau *object-oriented programming* merupakan suatu pendekatan pemrograman yang menggunakan *object* dan *class* (Wibowo, 2015). Konsep dasar pemrograman berorientasi objek atau *object-oriented programming* (OOP) yang dapat berjalan di PHP adalah :

1. *Class* didefinisikan dengan menampung nilai properti dan *method*, properti adalah sebuah data yang menjelaskan tentang *class* dan metode adalah tingkah laku yang dapat dilakukan oleh *object*.
2. *Object* adalah hasil instaniasi dari *class* dan mengandung seluruh *resource* yang telah didefinisikan pada *class*.
3. *Encapsulation* adalah mekanisme “membungkus” sebuah data pada sebuah *object*. Tiga *modifier* yang dapat diimplementasikan untuk melakukan “pembungkusan” data yaitu *private*, *protected*, dan *public*.
4. *Polymorphisme* membuat objek-objek yang berasal dari *subclass* yang berbeda, diperlakukan sebagai objek-objek dari satu *superclass*.

5. *Constructor*, *class* yang memiliki metode konstruktor memanggil metode ini pada setiap objek yang baru dibentuk.
6. *Destructor*, *destructor* akan segera dipanggil setelah tidak ada referensi lain.
7. *Inheritance* (pewarisan) adalah cara untuk menggunakan kembali kode objek yang ada, atau untuk mendirikan *subtype* dari objek yang sudah ada, atau keduanya, tergantung pada dukungan bahasa pemrograman.
8. *Final Keyword* adalah untuk mencegah proses *overriding method* pada *class* anak (sub-*class*) hal ini dapat diterapkan pada metode dan *class*.
9. *Class Abstraction*, *class* yang mendefinisikan sebagai *abstract* tidak bisa diintiasi, dan *class* yang terdiri paling tidak satu *method abstract* harus didefinisikan sebagai *abstract class*.
10. *Object Interfaces* memungkinkan untuk membuat kode yang menentukan *method* mana yang akan diimplementasi tanpa harus mendefinisikan bagaimana *method* tersebut akan bekerja (hanya nama *method* saja).

2.12 **Black Box Testing**

Black box testing atau pengujian kotak hitam, disebut juga pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak. Teknik pengujian kotak hitam memungkinkan untuk membuat beberapa kumpulan kondisi masukan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk program (Pressman, 2010).

Menurut Simarmata (2010), klasifikasi *black box testing* mencakup beberapa pengujian yaitu :

1. Pengujian fungsional (*functional testing*)

Perangkat lunak diuji untuk persyaratan fungsional. Pengujian dilakukan dalam bentuk tertulis untuk memeriksa perintah-perintah pengguna, manipulasi data, serta operasi *back-end*.

2. Pengujian tegangan (*stress testing*)

Pengujian tegangan berkaitan dengan kualitas aplikasi di dalam lingkungan.

3. Pengujian beban (*load testing*)

Pada pengujian beban aplikasi akan diuji dengan beban berat atau masukan, seperti yang terjadi pada pengujian situs *web*.

4. Pengujian khusus (*ad-hoc testing*)

Jenis pengujian ini dilakukan tanpa penciptaan rencana pengujian (*test plan*) atau kasus pengujian (*test case*).

5. Pengujian penyelidikan (*exploratory testing*)

Pengujian penyelidikan sama dengan pengujian khusus dan dilakukan untuk mempelajari atau mencari aplikasi.

6. Pengujian usabilitas (*usability testing*)

Pengujian ini disebut juga pengujian untuk keakraban pengguna (*testing for user-friendliness*). Pengujian ini dilakukan jika *interface* pengguna dari aplikasi penting dan harus spesifik untuk jenis pengguna tertentu.

7. Pengujian asap (*smoke testing*)

Pengujian ini disebut pengujian kenormalan (*sanity testing*). Pengujian ini dilakukan untuk memeriksa apakah aplikasi tersebut sudah siap untuk pengujian yang lebih besar dan bekerja dengan baik tanpa cela sampai tingkat yang paling diharapkan. Pada sebuah pengujian baru atau perbaikan peralatan yang

terpasang, jika aplikasi “berasap”, aplikasi tersebut tidak bekerja! Istilah ini juga merujuk kepada pengujian fungsi perangkat lunak dasar.

8. Pengujian pemulihan (*recovery testing*)

Pengujian pemulihan (*recovery testing*) pada dasarnya dilakukan untuk memeriksa seberapa cepat dan baiknya aplikasi dapat pulih terhadap semua jenis *crash* atau kegagalan hardware.

9. Pengujian volume

Pengujian volume dilakukan terhadap efisiensi dari aplikasi.

10. Pengujian domain (*domain testing*)

Pengujian domain merupakan penjelasan yang sangat sering menjelaskan teknik pengujinya.

11. Pengujian scenario

Pengujian skenario adalah pengujian yang realistis, kredibel dan memotivasi stakeholders, tantangan untuk program dan mempermudah penguji untuk melakukan evaluasi.

12. Pengujian regresi (*regression testing*)

Pengujian regresi adalah gaya pengujian yang berfokus pada pengujian ulang (*retesting*) setelah ada perubahan.

13. Penerimaan pengguna (*user acceptance*)

Pada jenis pengujian ini, perangkat lunak akan diserahkan kepada pengguna untuk mengetahui apakah perangkat lunak memenuhi harapan pengguna dan bekerja seperti yang diharapkan.

14. Pengujian alfa (*alpha testing*)

Pengujian ini, pengguna akan diundang ke pusat pengembangan.

15. Pengujian beta (*beta testing*)

Pada jenis ini, perangkat lunak didistribusikan sebagai sebuah versi beta dengan pengguna yang menguji aplikasi di situs mereka.

DAFTAR PUSTAKA

- Aminudin. (2015). *Cara Efektif Belajar Framework Laravel*. Yogyakarta: Lokomedia.
- Anisya. (2013, Agustus). Aplikasi Sistem Database Rumah Sakit Terpusat. *Jurnal Momentum*, 15, 10.
- Bahra, A. (2013). *Analisis dan Desain Sistem Informasi*. Yogyakarta: Graha Ilmu.
- Fathansyah. (2012). *BASIS DATA*. Bandung: Informatika Bandung.
- Greenberg, P. (2010). *Customer Relationship Management as the Speed of Light*. Fourth Edition McGraw-Hill.
- Gullick, L. M. (2011, Maret 27). *Papers on the Science of Administration*. Institute of Public Administration. Retrieved from kbbi.kemendikbud.go.id.
- Handayani, H. (2012). XAMPP. *XAMPP*, 4. Retrieved from imulti.org.
- Haviluddin. (2011). Memahami Penggunaan UML (Unified Modelling Language). *Jurnal Informatika Mulawarman*, 6, 15.
- Hendrayudi. (2009). *VB 2008 untuk Berbagai Keperluan Programming*. Jakarta: PT. Elex Media Komputindo.
- Hidayat, A., & Surarso, B. (2012, Maret 10). Penerapan Arsitektur Model View Controller (MVC) . *Seminar Nasional Teknologi Informasi dan Komunikasi 2012 (SENTIKA 2012)*, 8.
- Hutahaean, J. (2014). *Konsep Sistem Informasi*. Yogyakarta: CV BUDI UTAMA.
- Hutahaean, J. (2014). *Konsep Sistem Informasi*. Yogyakarta: CV Budi Utama.
- Indrajani. (2014). *Pengantar Sistem Basis Data Case Study All In One*. Jakarta: PT Elex Media Komputindo.
- Indrajani. (2015). *Database Design*. Jakarta: PT. Elex Media Komputindo.
- Indriyanti, A. D., & Pratama, R. (2015). Perancangan dan Pembuatan Forum Makanan Berbasis Web. *Jurnal Manajemen Informatika*, 6.
- Koespradono, Suraya, & K, Y. R. (2013). Sistem Informasi Pengolahan Data. *Jurnal SCRIPT*, 1, 9.
- Lubis, A. (2016). *Basis Data Dasar*. Yogyakarta: CV. Budi Utama.

- Masruri, M. H. (2015). *Membangun SMS Gateway dengan Gammu dan Kalkun*. Jakarta: PT. Elex Media Komputindo.
- Mauluddin, A. (2011, November). Aplikasi Web Browser Menggunakan Metode URL. *Jurnal Infromasi*, 4, 12.
- Neyfa, B. C. (2016). Perancangan Aplikasi E-Canteen Berbasis Adroid Dengan Menggunakan Metode Object Oriented Analisis dan Design (OOAD). *Jurnal Aplikasi*, 10.
- Prayitno, I. (2010). *Kupas Tuntas Malware*. Jakarta: PT. Elex Media Komputindo.
- Pressman, R. S. (2010). *Rekayasa Perangkat Lunak - Buku Satu, Pendekatan Praktisi (Edisi 7)*. Yogyakarta: Penerbit ANDI.
- Prihastono, E. (2012). Pengukuran Kepuasan Konsumen Pada Kualitas. *Jurnal Web*, 11.
- Rahman, S. (2013). *Cara Gampang Bikin CMS Tanpa Ngoding*. Jakarta: Mediakita.
- Riska, Harihanto, & Nurmanina, A. (2013). Studi Tentang Penggunaan Internet Oleh Pelajar. *eJournal Sosiatri-Sosiologi*, 1, 13.
- Rosa, & Shalahuddin, M. (2014). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika Bandung.
- Rosmala, D., Ichwan, M., & Gandalisha, M. I. (2011, Mei). Komparasi Framework MVC (Code Igniter dan CakePHP). *Jurnal Informatika*, 2, 9.
- Rozi, Z. A. (2015). *Bootstratp Design Framework*. Jakarta: PT. Elex Media Komputindo.
- Silitonga, J., & dkk. (2012). Pendaftaran Mahasiswa Baru Berbasis Mobile. *Jurnal PHPMyAdmin*, 3.
- Sulistiyawan, Rubianto, & Rahmad, S. (2008). *Modifikasi Blog Multiply dengan CSS*. Jakarta: PT. Elex Media Komputindo.
- Supardi, Y. (2015). *Buku Kuliah Web Programming 1*. Jakarta: @Dotakom Lintas Buana.
- Tarmuji, A., & Hastiany. (2013). Pembuatan Enterprise Architecture Dengan Menggunakan Kerangka Kerja . *Jurnal Informatika*, 7, 11.
- Tjiptono, F. (2008). *Service Management (mewujudkan layanan prima)*. Yogyakarta: Andi Offset.
- Wibowo, K. (2015, Desember). Analisa Konsep Object Oriented Programming Pada Bahasa Pemrograman PHP. *JURNAL KHATULISTIWA INFORMATIKA*, 3, 1.
- Winarno, E., Zaki, A., & Community, S. (2014). *24 Jam Belajar PHP*. Jakarta: PT. Elex Media Komputindo.

Yanto, R. (2016). *Manajasis Basis Data Menggunakan MySQL*. Yogyakarta: CV. Budi Utama.

Yuhefizar. (2013). *Cara Mudah dan Murah Membangun dan Mengelola Website*. Yogyakarta: Graha Ilmu.