BAB II

LANDASAN TEORI

BAB II
LANDASAN TEORI

2.1 Landasan Teori

Dalam bab ini menjelaskan tentang landasan teori yang terdiri dari berbagai teori yang dikemukakan oleh para ahli, serta mencakup komponen-komponen yang mendukung sistem informasi ini untuk permohonan izin ketidakhadiran.

2.1.1 Sistem Informasi

Sistem informasi adalah serangkaian langkah untuk mengumpulkan, menyimpan, dan menganalisis informasi dengan maksud tertentu. Dengan melibatkan data sebagai masukan dan menghasilkan laporan sebagai keluaran, yang kemudian diterima oleh sistem lain. Selain itu, sistem informasi juga terlibat dalam strategi dan kegiatan organisasi untuk mengambil tindakan atau membuat keputusan[4].

Selain itu definisi lain dari sistem informasi yaitu, merupakan gabungan modul yang terstruktur dan berasal dari elemen-elemen terkait seperti perangkat keras, perangkat lunak, orang-orang dan jaringan, yang beroperasi pasca serangkaian komputer yang saling terhubung atau berinteraksi untuk mengolah data menjadi informasi guna mencapai tujuan tertentu[5].

Dari uraian penjelasan sistem informasi juga memiliki batasan yang dirancang untuk memastikan operasional yang efisien dan fokus pada tujuan yang spesifik, menghindari kerumitan yang tidak perlu serta menjaga integritas data. Informasi yang dihasilkan harus berkualitas tinggi, relevan, dapat dipercaya dan berguna bagi pengguna. Untuk mencapai hal ini, sistem informasi harus beroperasi berdasarkan prinsip-prinsip yang telah ditetapkan, seperti keandalan, keamanan, dan konsistensi. Selain itu, ketepatan waktu dalam penyampaian informasi juga penting, karena informasi yang akurat. Sistem informasi harus mampu menyediakan informasi yang akurat dan tepat waktu guna mendukung pengambilan keputusan yang cepat dan efektif.

2.1.2 Izin

Perizinan adalah elemen penting untuk melindungi hak hukum atas kepemilikan atau pelaksanaan suatu kegiatan. Adanya perizinan memberikan kepastian dan perlindungan hukum terhadap tindakan atau kepemilikan. Tanpa perizinan, baik untuk kepemilikan benda maupun pelaksanaan kegiatan, dapat muncul berbagai masalah yang mengganggu ketertiban dan tatanan masyarakat. Ini terjadi karena perizinan bertujuan untuk mengendalikan kegiatan atau perilaku individu atau kelompok secara protektif[6].

2.1.3 Pemrograman Berorientasi Objek

Definisi dari pemrograman berorientasi objek adalah suatu pendekatan baru dalam berpikir dan berlogika dalam menangani masalah-masalah yang ingin dipecahkan dengan bantuan komputer. Dalam pemrograman berorientasi objek, setiap entitas tunggal disebut sebagai objek, yang memiliki kombinasi struktur data dan fungsi yang spesifik. Objek dapat merepresentasikan berbagai hal dalam dunia nyata seperti manusia, tempat, benda, peristiwa atau konsep-konsep penting dalam suatu aplikasi. Contohnya, objek bisa berupa benda seperti mesin, gedung atau mobil serta kejadian seperti pembayaran uang pendidikan, registrasi biodata siswa atau membaca buku. Dengan demikian, objek merupakan sesuatu yang nyata dan dapat diidentifikasi, dirasakan, diakses dan memberikan manfaat bagi penggunanya[7].

Pemrograman berorientasi objek, memiliki konsep seperti berikut :

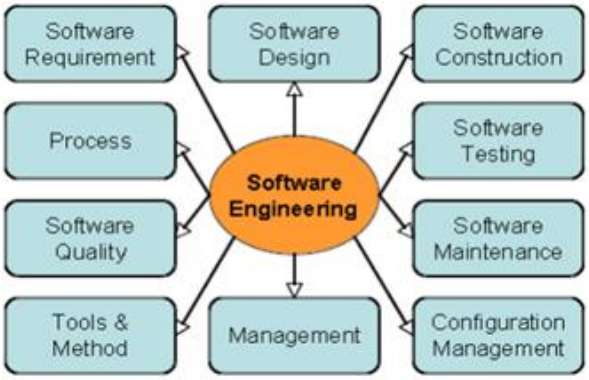
1. Kelas (*Class*), kelas adalah kerangka kerja dasar dari mana objek dibuat.
2. Objek (*Object*), objek adalah entitas yang hidup dalam memori dan mewakili data.
3. Enkapsulasi (*Encapsulation*), enkapsulasi adalah prinsip yang menyembunyikan detail implementasi internal dari sebuah objek dan hanya mengekspos antarmuka publik. Hal ini membantu melindungi data dari modifikasi yang tidak disengaja dan meningkatkan modularitas.
4. Pewarisan (*Inheritance*), mekanisme yang memungkinkan sebuah kelas untuk mewarisi sifat dan perilaku dari kelas lain. Pewarisan memungkinkan menggunakan kembali kode dan membentuk hubungan hierarki antara kelas.
5. Polimorfisme (*Polymorphism*), adalah konsep yang memungkinkan objek untuk diperlakukan sebagai *instance* dari kelas induk mereka, bukan kelas mereka yang sebenarnya. Polimorfisme memfasilitasi penggunaan metode yang sama dengan implementasi yang berbeda di kelas yang berbeda.
6. Abstraksi (*Abstraction*), adalah proses menyederhanakan kompleksitas dengan menyembunyikan detail implementasi yang tidak perlu dan hanya menampilkan fungsi penting kepada pengguna. Abstraksi membantu dalam menangani kompleksitas sistem besar.

Dalam pemrograman berorientasi objek, program diatur dengan mengelompokkan data dan fungsi yang berhubungan ke dalam unit-unit yang disebut objek, yang memungkinkan pengkodean menjadi lebih standar, dapat digunakan kembali dan lebih mudah untuk dikelola.

2.1.4 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah penerapan pendekatan yang tersruktur dan sistematis dalam pengembangan, pengoperasian dan pemeliharaan perangkat lunak yaitu aplikasi dari perangkat lunak itu sendiri. Pemilihan model proses untuk rekayasa perangkat lunak bergantung

pada sifat dari aplikasi dan proyek yang bersangkutan, metode serta alat bantu yang akan digunakan, pengawasan, serta penyampaian yang dibutuhkan[8]. Ruang lingkup rekayasa perangkat lunak terbagi atas beberapa bagian dengan di antaranya sebagai berikut[9]:

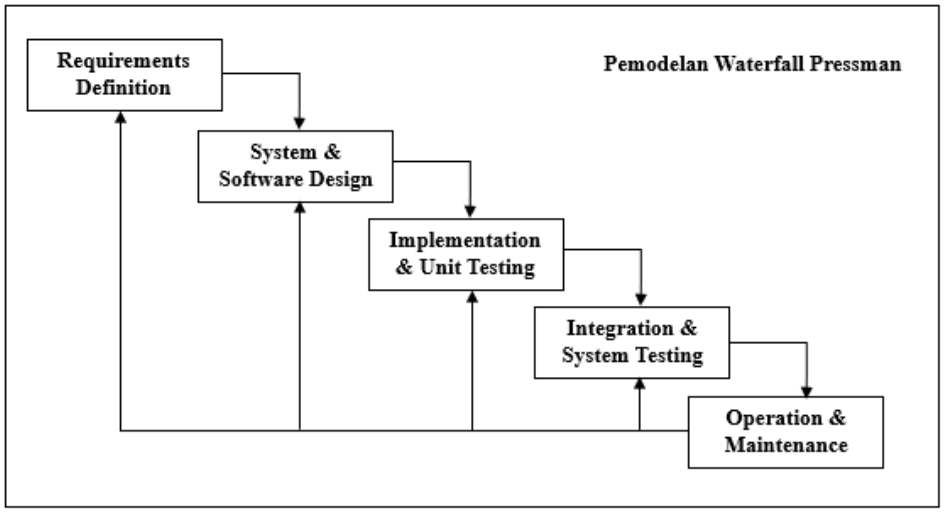


Gambar 2. 1 Ruang Lingkup Rekayasa Perangkat Lunak

1. *Software requirements*, hubungannya saling berkaitan dengan spesifikasi kebutuhan perangkat lunak dan persyaratan perangkat lunak.
2. *Software design*, melibatkan proses menentukan struktur, komponen, interface dan atribut lainnya dari perangkat lunak.
3. *Software construction*, berhubungan dengan aspek-aspek detail dalam pengembangan perangkat lunak, termasuk algoritma, penulisan kode, pengujian dan identifikasi kesalahan.
4. *Software testing*, melibatkan pengujian terhadap setiap perilaku dari perangkat lunak secara keseluruhan.
5. *Software maintenance*, melibatkan serangkaian usaha yang dilakukan setelah perangkat lunak tersebut telah digunakan secara aktif.
6. *Software configuration management*, berkaitan dengan upaya mengubah konfigurasi perangkat lunak guna memenuhi kebutuhan yang spesifik.
7. *Software engineering management*, melibatkan pengaturan serta penilaian dari proses rekayasa perangkat lunak , yang mencakup juga perencanaan proyek perangkat lunak.
8. *Software engineering tools and methods*, mencakup analisis teoritis terhadap alat bantu dan teknik yang digunakan dalam proses rekayasa perangkat lunak.
9. *Software engineering process*, berhubungan dengan pengertian, penerapan, evaluasi, manajemen, perubahan dan peningkatan dari proses rekayasa perangkat lunak.
10. *Software quality*, menekankan pentingnya kualitas dan siklus hidup dari perangkat lunak.

Metode pengembangan system yang digunakan pada sistem ini adalah menggunakan metode *waterfall*, metode *waterfall* adalah suatu pendekatan yang terstruktur dan berurutan yang dimulai

dari penentuan kebutuhan sistem dan kemudian berlanjut ke tahap analisis, perancangan, pengkodean, pengujian/verifikasi, dan pemeliharaan. Dinamakan "air terjun" karena setiap tahap dalam metode ini harus menunggu penyelesaian tahap sebelumnya, yaitu tahap penentuan kebutuhan[10]. Berikut metode *waterfall* menurut Roger S. Pressman Ph.D. :



Gambar 2. 2 Metode *Waterfall* menurut Roger S. Pressman Ph.D.

1. *Requirement Definition*

   Proses pencarian kebutuhan perangkat lunak difokuskan agar pengembang memahami fungsionalitas dan antarmuka pengguna. Hasil dari pencarian kebutuhan ini harus didokumentasikan dan disampaikan kepada pelanggan

2. *System and Software Design*

   Proses ini bertujuan mengubah kebutuhan menjadi rancangan perangkat lunak sebelum *coding* dimulai. Desain harus mengimplementasikan kebutuhan yang telah diidentifikasi dan didokumentasikan sebagai bagian dari konfigurasi perangkat lunak.

3. *Implementasion and Unit Testing*

   Pada tahap ini, desain yang telah dibuat diimplementasikan ke dalam kode program. Setiap unit atau komponen dari perangkat lunak diuji secara individual (*unit testing*) untuk memastikan bahwa setiap bagian berfungsi dengan benar.

4. *Integration and System Testing*

   Segala sesuatu yang dibuat harus diuji, termasuk perangkat lunak. Semua fungsi perangkat lunak harus diuji untuk memastikan tidak ada kesalahan dan hasilnya sesuai dengan kebutuhan yang telah ditentukan sebelumnya.

5. *Operation and Maintenance*

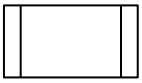
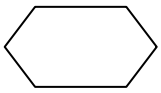
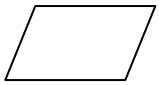
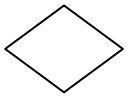
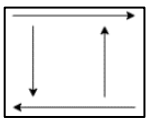
   Pemeliharaan perangkat lunak diperlukan, termasuk pengembangannya, karena perangkat lunak tidak statis. Kesalahan kecil mungkin ditemukan atau fitur baru perlu ditambahkan.

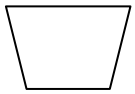
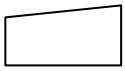
Pengembangan juga diperlukan saat ada perubahan eksternal, seperti pergantian sistem operasi atau perangkat.

2.1.5 *Flowchart*

Flowchart atau diagram alir adalah jenis diagram yang menggambarkan algoritma atau langkah-langkah instruksi dalam urutan tertentu dalam suatu sistem. *Flowchart* digunakan sebagai dokumentasi untuk menjelaskan secara logis bagaimana sistem yang akan dibangun dan dapat membantu dalam menemukan solusi untuk masalah yang mungkin timbul selama pengembangan sistem. *Flowchart* menggunakan simbol-simbol, di mana setiap simbol mewakili proses tertentu dan proses-proses tersebut dihubungkan dengan garis penghubung untuk menunjukkan urutan jalannya proses-proses tersebut[13].

Tabel 2. 1 Simbol-simbol *Flowchart*

No.	Simbol	Nama	Keterangan
1.		*Terminal*	Memulai dan mengakhiri suatu program
2.		*Process*	Proses perhitungan atau proses pengolahan data
3.		*Predefined Process* (sub program)	Permulaan sub program atau proses pengolahan data
4.		*Preparation*	Proses inisialisasi atau pemberian harga awal
5.		*Input – Output*	Memasukan data maupun menunjukan hasil dari suatu *process* tanpa tergantung dengan jenis peralatannya
6		*Decision*	Memilih proses berdasarkan kondisi yang ada
7.		*Flow*	Menghubungkan antara simbol satu dengan simbol yang lain atau menyatakan jalannya arus dalam suatu proses. Simbol arus ini sering disebut juga dengan *connecting line*

No.	Simbol	Nama	Keterangan
8.		*Manual Operation*	Digunakan untuk menyatakan suatu proses yang tidak dilakukan komputer
9.		*Manual Input Symbol*	Digunakan untuk menginputkan data secara manual dengan *keyboard*

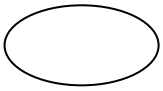
2.1.6 UML (*Unified Modeling Language*)

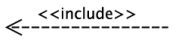
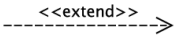
Unified Modeling Language (UML) adalah bahasa yang digunakan untuk pemodelan perangkat lunak yang telah terstandarisasi dan digunakan untuk memvisualisasikan, menspesifikasikan, membangun dan mendokumentasikan berbagai bagian dari sistem perangkat lunak. UML juga menjadi sebuah media komunikasi dengan menggambarkan diagram dan teks-teks pendukung di dalamnya[11]. Pada UML terdapat banyak diagram beserta penjelasannya seperti berikut :

a. *Use Case Diagram*

Use Case Diagram merupakan diagram yang menggambarkan hubungan antara pelaku dan sistem. *Use Case Diagram* menjelaskan bagaimana interaksi antara satu atau lebih pelaku dengan sistem yang akan kembangkan[12]. Simbol-simbol yang terdapat pada *Use Case Diagram* sebagai berikut :

Tabel 2. 2 Simbol-simbol *Use Case*

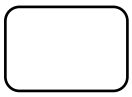
No.	Simbol	Nama	Keterangan
1.		*Use case*	Deskripsi dari urutan langkah-langkah yang dilakukan oleh sistem yang menghasilkan hasil yang dapat diukur bagi seorang *actor*
2.		*Actor*	Menspesifikasikan kumpulan peran yang dimainkan oleh pengguna saat berinteraksi dengan kasus pengguna
3.		*Association*	Menghubungkan objek satu dengan objek lainnya

No.	Simbol	Nama	Keterangan
4.		*Generalization*	Hubungan generalisasi dan spesialisasi antara dua *use case* yang menunjukkan bahwa satu *use case* memiliki fungsi yang lebih umum daripada yang lainnya
5.		*Include*	Menambahkan relasi pada sebuah *use case* jika terdapat kebutuhan untuk menjalankan suatu fungsi tertentu
6.		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari kasus pengguna sumber pada titik tertentu

b. *Activity Diagram*

Activity Diagram adalah diagram yang memodelkan berbagai aliran aktivitas dalam sistem yang sedang dikembangkan. Diagram ini menunjukkan bagaimana setiap aliran dimulai dari keputusan-keputusan yang mungkin terjadi dan bagaimana setiap aktivitas berakhir. Diagram ini juga memvisualisasikan urutan aliran kerja, pengambilan keputusan, percabangan dan interaksi antar aktivitas hingga mencapai titik akhir dalam sistem[13]. Simbol-simbol yang terdapat pada *Activity Diagram* yaitu :

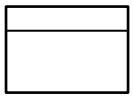
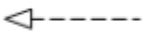
Tabel 2. 3 Simbol-simbol *Activity Diagram*

No.	Simbol	Nama	Keterangan
1.		*Activity*	Menunjukkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2.		*Action*	*State* dari sistem yang mengambarkan eksekusi dari suatu aksi
3.		*Initial Node*	Bagaimana objek dibentuk atau diawali
4.		*Activity Final Node*	Bagaimana objek dibentuk dan diakhiri
5.		*Decision*	Menggambarkan suatu keputusan atau tindakan yang harus diambil pada kondisi tertentu
6.		*Line Connector*	Menghubungkan satu simbol dengan simbol yang lain

c. *Class Diagram*

Class Diagram digunakan untuk menampilkan kelas-kelas dan paket-paket dalam sistem. Selain itu, C*lass Diagram* juga dapat memberikan ilustrasi sistem secara statis serta hubungan antar kelas-kelas tersebut. Umumnya, beberapa *Class Diagram* dirancang untuk satu sistem tunggal. Beberapa diagram akan menampilkan subset dari kelas-kelas dan relasinya. Beberapa diagram dapat dibuat sesuai kebutuhan untuk mendapatkan gambaran lengkap terhadap sistem yang dibangun[14]. Simbol-simbol yang terdapat pada *Class Diagram* seperti berikut :

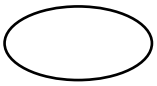
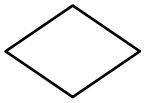
T**abel 2. 4** Simbol-simbol *Class Diagram*

No.	Simbol	Nama	Keterangan
1.		*Generalization*	Hubungan di mana objek anak mewarisi perilaku dan struktur data dari objek induk di atasnya
2.		*Nary Association*	Digunakan untuk menghindari asosiasi dengan lebih dari dua objek
3.		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek
6.		*Dependency*	Hubungan dimana perubahan yang terjaddi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7.		*Association*	Menghubungkan antara objek satu dengan objek yang lain

2.1.7 ERD (*Entity Relationship Diagram)*

Entity Relationship Diagram (ERD) adalah representasi visual yang digunakan untuk merancang struktur *database*. ERD membantu dalam mendeskripsikan data yang akan disimpan dalam sistem serta batasannya. Komponen kunci yang terdapat dalam ERD meliputi set entitas, set relasi dan juga kendala atau *constraints*[15].

Tabel 2. 5 Simbol-simbol *Entity Relationship Diagram*

No.	Simbol	Nama	Keterangan
1.		Entitas	Data utama yang disimpan pada suatu tabel basis data. Penamaan entitas umumnya berbentuk kata benda dan belum merupakan nama tabel
2.		Atribut	*Field* atau kolom data yang perlu disimpan dalam sebuah entitas
3.		Relasi	Relasi menunjukkan adanya hubungan antara sejumlah entitas dari himpunan entitas yang berbeda
4.		*Link*	Menghubungkan antara himpunan entitas dan himpunan entitas dengan atributnya

2.1.8 *Database*

Database atau basis data adalah kumpulan data yang terstruktur dan saling terhubung yang disusun agar dapat diakses dengan cepat dan mudah. Data dalam *database* disimpan dalam format seperti *file*, tabel atau arsip elektronik dan tersimpan dalam media penyimpanan elektronis. Tujuan utama dari pengaturan, penyortiran, pengelompokan dan pengorganisasian data dalam *database* adalah untuk memudahkan pencarian dan penggunaan sesuai dengan kebutuhan yang ditentukan. *Database* memiliki 8 fungsi utama, yaitu :

1. *Create database*
2. *Drop database*
3. *Create table*
4. *Drop table*
5. *Insert*
6. *Read*
7. *Update*
8. *Delete*

Dalam sebuah *database*, informasi disimpan dalam tabel yang terdiri dari baris dan kolom. Data dalam tabel mewakili fakta atau angka. Setiap baris dalam tabel yang berisi informasi disebut *record*, sementara setiap kolom yang menggambarkan atribut umum untuk semua *record* disebut kolom[16].

a. DBMS (*Database Management System*)

DBMS (*Data Base Management System*) merupakan *software* yang dirancang untuk mengelola basis data dengan efisien, terutama dalam mengatur dan mengelola kumpulan data besar. Tujuannya adalah memfasilitasi manipulasi data secara efektif dan memudahkan pengelolaan data secara keseluruhan[17]. Di dalam DBMS terdapat dua jenis bahasa yang umum digunakan seperti berikut[18]:

1. DDL (*Data Definition Language*)

   DDL digunakan untuk mendefinisikan struktur dan skema dari *database*. Termasuk pembuatan, modifikasi dan penghapusan tabel, kolom, indeks dan segala definisi lainnya yang berkaitan dengan struktur *database*. Perintah-perintah di dalam DDL meliputi :

   - *CREATE* : berfungsi untuk membuat membuat *database* dan tabel baru.
   - *ALTER* : berfungsi untuk struktur tabel yang ada.
   - *DROP* : berfungsi untuk menghapus *database* dan tabel.

2. DML (*Data Manipulation Language)*

   DML merupakan komponen dari bahasa kueri *database* yang bertujuan untuk mengelola data di dalam tabel. DML dapat digunakan untuk mengubah nilai-nilai dalam tabel, mengambil subset data yang diperlukan dan menjalankan berbagai operasi lain yang berkaitan dengan manipulasi data. Perintah-perintah di dalam DML meliputi :

   - *INSERT* : berfungsi untuk menambahkan data baru ke dalam tabel.
   - *UPDATE* : berfungsi untuk mengubah data yang sudah ada dalam tabel.
   - *DELETE* : berfungsi untuk menghapus data dari tabel.
   - *SELECT* : berfungsi untuk menampilkan data tertentu dari tabel berdasarkan kriteria yang ditentukan.

b. SQL (*Structured Query Language)*

Structured Query Language (SQL) adalah bahasa yang digunakan untuk mengakses data dalam basis data relasional. Semua server basis data mendukung penggunaan SQL untuk mengelola dan mengambil data. SQL adalah bahasa pemrograman khusus yang fokus pada manajemen data dalam Sistem Manajemen Basis Data Relasional (RDBMS). Instruksi SQL umumnya berupa perintah sederhana yang digunakan untuk manipulasi dan pengambilan data dari *database* relasional atau struktural. Istilah "*query*" sering digunakan sebagai penyebutan untuk perintah SQL[19].

2.1.9 PHP (*Hypertext Preprocessor*)

PHP (Hypertext Preprocessor) adalah bahasa pemrograman yang berjalan di *web* server dan mengolah data di server. Data dari pengguna client diolah dan disimpan di *database*, lalu dapat

ditampilkan kembali. Untuk menjalankan kode PHP, file harus diunggah ke server melalui proses transfer data dari komputer *client*[20].

Proses kerja PHP dimulai ketika *browser* mengirimkan permintaan untuk membuka halaman *web*. Berdasarkan alamat internet URL *(Uniform Resource Locator*), *browser* mengidentifikasi alamat yang diminta dan mengirimkan informasi yang diperlukan ke *web* server[21].

2.1.10 *Framework*

Framework adalah serangkaian instruksi yang terorganisir dalam bentuk *class* dan *function*, yang memiliki tujuan untuk membantu pengembang dalam menggunakan kembali kode tanpa perlu menulis ulang *syntax* yang sama berulang kali. Dengan demikian, penggunaan *framework* dapat menghemat waktu dan usaha dalam pengembangan perangkat lunak[22]. Di dalam *framework* meliputi hal-hal seperti :

A. MVC (*Model View Controller*)

Model View Controller (MVC) adalah metode pengembangan aplikasi yang memisahkan data (model), tampilan (*view*) dan pengaturannya (*controller*). MVC memisahkan pengembangan aplikasi berdasarkan komponen utama seperti manipulasi data, antarmuka pengguna dan kontrol dalam aplikasi *web*. Dalam pengembangan sistem informasi, MVC digunakan untuk model merepresentasikan struktur data dan mengelola database, sementara *view* menampilkan *output* kepada pengguna dan *controller* menghubungkan *model* dan *view*, memproses data dan mengirimkannya ke halaman *web*[23].

B. *Framework* Laravel

Laravel adalah sebuah *framework web* berbasis PHP yang sumbernya dapat diakses oleh siapapun dan tidak berbayar Laravel digunakan untuk pengembangan aplikasi *web* dengan pola MVC (*Model View Controller*). Di dalamnya terdapat mekanisme *routing* yang menghubungkan permintaan pengguna dengan pengontrol sehingga pengontrol tidak langsung menerima permintaan tersebut. Ada lima konsep arsitektur dalam Laravel yang memiliki fungsi masing-masing, yaitu :

a. *Routes,* untuk memberikan akses pada setiap permintaan sesuai alur yang ditentukan sebelumnya.
b. *Controller,* untuk menghubungkan *model* dan tampilan dengan perintah-perintah untuk mengatur cara data ditampilkan dari *model* ke tampilan atau sebaliknya.
c. *Model,* digunakan untuk menyimpan dan mengelola data dalam *database*. Struktur model Laravel mencakup nama tabel, *primary key* dan *fillable*. Tabel berisi nama tabel, *primary key* adalah kunci utama dan *fillable* adalah atribut-atribut yang dapat diisi.

d. *View*, berisi kode HTML (*HyperText Markup Language*) untuk menampilkan data ke *browser*, dengan format tampilan menggunakan istilah "*blade*", seperti contoh "*view.blade.php*".
e. *Migrations*, adalah proses desain tabel dalam *database* yang berfungsi sebagai panduan atau sistem kontrol untuk skema *database*.

Keunggulan Laravel meliputi performa yang tinggi, stabilitas *reload* data, keamanan, fitur *Blade*, ketersediaan *library* yang siap pakai dan fitur pengelolaan *migrations* untuk skema tabel *database*.

2.1.11 *Web*

Web adalah kumpulan halaman yang menampilkan informasi dalam bentuk teks, gambar, animasi, audio, video dan kombinasi lainnya, baik statis maupun dinamis, yang terhubung ke internet. Secara umum, *web* adalah perangkat lunak yang menampilkan dokumen di internet dan memungkinkan pengguna untuk mengakses internet melalui perangkat lunak dengan koneksi internet[24].