

## **BAB III**

### **LANDASAN TEORI**

#### **3.1 Pengertian Rancang Bangun**

Perancangan sistem merupakan suatu aktifitas/proses yang dilakukan untuk menggambarkan bagaimana proses bisnis berjalan dengan membuat *diagram* seperti *use case diagram*. Salah satu proses perancangan atau proses pengembangan sistem yang sudah ada dan banyak diketahui oleh pengembang adalah *System Development Life Cycle (SDLC) Waterfall*, langkah yang ada dalam *SDLC Waterfall* ini terdiri dari 5 bagian yaitu *Requirement, Design, Implementation, Integration & Testing* dan *Operation & Maintenance* (Taufiq, R., Ummah, R. R., Nasrullah, I., & Permana, A. A. 2019).

Perancangan didefinisikan sebagai pengembangan perencanaan dan pembuatan sketsa dari beberapa elemen yang terpisah. Pembangunan adalah kegiatan menciptakan atau memperbaiki sistem yang telah ada secara keseluruhan (Aprilman, D., & Widodo, S. 2022).

Rancang bangun adalah kegiatan menerjemahkan hasil analisa kedalam bentuk paket perangkat lunak, kemudian menciptakan sistem atau memperbaiki sistem yang sudah ada (Jh, A. R., & Prastowo, A. T. 2021).

#### **3.2 Pengertian Administrasi**

Secara etimologi kata Administrasi berasal dari bahasa Latin, yaitu Ad yang memiliki arti intensif dan ministrare yang memiliki arti melayani, membantu, dan memenuhi. Dalam bahasa Inggris “administration”. Menurut KBBI administrasi adalah usaha dan kekuatan meliputi penetapan tujuan serta penetapan cara-cara penyelenggaraan pembinaan organisasi; usaha dan kegiatan yang berkaitan dengan penyelenggaraan pemerintahan; kegiatan kantor dan tata usaha. Administrasi adalah perencanaan, pengendalian, dan pengorganisasian pekerjaan perkantoran, serta pergerakan mereka yang melaksanakannya agar mencapai tujuan yang telah ditetapkan (Sutha, D. W. 2018).

Beberapa pengertian Administrasi menurut para ahli yang berpendapat (Kosassy, S. O. 2021), antara lain :

- a. George R. Terry, pengertian administrasi adalah kegiatan perencanaan, pengendalian, dan pengorganisasian pekerjaan perkantoran, serta pergerakan mereka yang melaksana-kannya agar mencapai tujuan yang telah ditetapkan.
- b. Menurut Arthur Grager, pengertian administrasi adalah fungsi tata penyelenggaraan terhadap komunikasi dan pelayanan warkat suatu organisasi.
- c. Menurut Sondang P. Siagian, arti administrasi adalah segala bentuk dari proses kerjasama antara dua individu atau lebih atas dasar rasionalitas terpilih untuk mencapai tujuan yang ditentukan sebelumnya.

Administrasi dapat disimpulkan sebagai segala sesuatu yang mengandung unsur pokok yang sama, yaitu adanya kegiatan tertentu, adanya manusia yang melakukan kerjasama serta mencapai tujuan yang telah ditentukan sebelumnya (Sibuea, N., & Tampubolon, M. 2022).

### 3.3 Pengertian Agenda

Pada dasarnya agenda mencakup pengurutan aktivitas, pengalokasian aktivitas pada fasilitas dan pemetaan aktivitas menurut urutan waktu. Tujuan agenda adalah meningkatkan efektivitas dan efisiensi waktu, mengurangi terjadinya keterlambatan (Raharja, Lutfiani, & Wardana, 2018).

Berikut ini adalah Arti, Makna, Pengertian, Definisi dari kata "agenda" menurut kamus besar bahasa Indonesia (KBBI) dan menurut para ahli bahasa. Arti kata agenda-agen-da/ agénda/ n 1. buku catatan yg bertanggal untuk satu tahun: acara rapat itu telah dicatat dl --;2. acara (yg akan dibicarakan dl rapat) (Setiawan, Y., Nurwanto, A., & Erlansari, A. 2019).

Menurut (Ramadani, 2019), agenda adalah suatu buku yang dipergunakan untuk mencatat surat-surat masuk dan keluar dalam satu tahun.

### 3.4 Pengertian Website

*Website* atau situs dapat diartikan sebagai kumpulan halaman yang menampilkan informasi data teks, data gambar diam atau gerak, data animasi, suara, video dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*).

Bersifat statis apabila isi informasi *website* tetap, jarang berubah, dan isi informasinya searah hanya dari pemilik *website*. Bersifat dinamis apabila isi informasi *website* selalu berubah-ubah, dan isi informasinya interaktif dua arah berasal dari pemilik serta pengguna *website* (Ibrahim, A., & Ambarita, A. 2018).

*Website* atau disingkat *web*, dapat diartikan sekumpulan halaman yang terdiri dari beberapa laman yang berisi informasi dalam bentuk data digital baik berupa text, gambar, video, audio dan animasi lainnya yang disediakan melalui jalur koneksi *internet*. *Website* adalah apa yang anda lihat *via browser*, sedangkan yang disebut *web* sebenarnya adalah sebuah aplikasi *web*, karena melakukan *action* tertentu dan membantu anda melakukan kegiatan tertentu (Ahmat Josi, 2017).

*Website* adalah keseluruhan halaman-halaman *web* yang terdapat dalam sebuah *domain* yang mengandung informasi (Fahrudin et al., 2011).

### 3.5 Pengertian *Framework*

*Framework* atau kerangka kerja adalah komponen siap pakai yang digunakan developer untuk menangani berbagai permasalahan dalam pemrograman, seperti pemanggilan variabel, file, koneksi ke *database* dan sebagainya. Supaya lebih fokus dan mampu menyelesaikan software menjadi lebih cepat dan efektif (Jh, A. R., & Prastowo, A. T. 2021).

*Framework* adalah kumpulan intruksi-intruksi yang yang dikumpulkan dalam *class* dan *function-function* dengan fungsi masing-masing untuk memudahkan *developer* dalam memanggilnya tanpa harus menuliskan *syntax* program yang sama berulang-ulang serta dapat menghemat waktu (Sallaby & Kanedi, 2020).

*Framework* juga bisa diartikan sebagai komponen – komponen pemrograman yang sudah jadi dan siap untuk digunakan kapan saja, sehingga pengembang aplikasi tidak perlu lagi membuat *scrip* yang sama untuk tugas – tugas yang sama (Kristianto, G. A. 2019).

### 3.6 Pengertian *CodeIgniter*

Menurut A Budiman, S Sunariyo, J Jupriyadi (2021) mengatakan bahwa *CodeIgniter* adalah sebuah *framework* yang dibuat menggunakan bahasa pemrograman *PHP* bertujuan untuk memudahkan para programmer *web* untuk

membuat atau mengembangkan aplikasi berbasis *web*. *CodeIgniter* memiliki eksekusi tercepat dibandingkan dengan framework lainnya. *CodeIgniter* bersifat open source dan menggunakan model basis MVC (*Model View Controller*), yang merupakan model konsep modern saat ini. *CodeIgniter* juga menawarkan banyak *library* yang dapat digunakan.

*Codeigniter* adalah sebuah *framework* php yang bersifat *open source* dan menggunakan metode MVC (*Model, View, Controller*) untuk memudahkan *developer* atau *programmer* dalam membangun sebuah aplikasi berbasis *web* tanpa harus membuatnya dari awal. Dengan menggunakan PHP *Codeigniter* diharapkan dapat memudahkan *developer* atau *programmer* untuk membuat aplikasi *web* dengan cepat dan mudah dibandingkan dengan membuat sebuah *web* dari awal (Destiningrum & Adrian, 2017).

*CodeIgniter* adalah Sebuah *framework* php yang bersifat *open source* dan menggunakan metode MVC (*Model, View, Controller*) untuk memudahkan *developer* atau *programmer* dalam membangun sebuah aplikasi berbasis *web* tanpa harus membuatnya dari awal (Sallaby & Kanedi, 2020).

### 3.7 Pengertian PHP (*Personal Home Page*)

PHP (*Personal Home Page*) adalah bahasa pemrograman yang bersifat *open source* dimana pengguna dapat mengembangkan kode didalamnya digunakan untuk membuat *web* dinamis. Bahasa ini sampai sekarang masih digunakan baik bagi programmer atau pemula karena penggunaanya yang mudah (Destiningrum & Adrian, 2017).

Kemudahan dan kepopuleran *PHP* menjadikan Bahasa pemrograman ini digemari oleh banyak orang. Selain kemudahan dan kepopulerannya, yang menjadi daya tarik pengguna yaitu *PHP* dapat digunakan dengan gratis dan bersifat terbuka. *PHP* sering di sebut pemrograman *server side*, karena dalam pemrosesan *PHP* terjadi pada komputer *server* (Andre, 2019).

PHP merupakan bahasa pemrograman yang berfungsi sebagai pengolahan data pada sebuah *server* yang berjalan dalam sebuah *web server*. PHP adalah bahasa *server-side-scripting* yang menyatu dengan HTML untuk membuat halaman *web* yang *dinamis*. Karena PHP merupakan *server-side-scripting* maka *sintaks* dan perintah-perintah PHP akan diesksekusi di-*server* kemudian hasilnya

akan dikirimkan ke *browser* dengan format HTML. Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh *user* sehingga keamanan halaman *web* lebih terjamin. PHP dirancang untuk membuat halaman *web* yang *dinamis*, yaitu halaman *web* yang dapat membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman *web* (Sari et al., 2019).

### 3.8 Pengertian MySQL (*My Structure Query Language*)

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis. Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis (Lestanti & Susana, 2016).

*Database MySQL* merupakan salah satu perangkat lunak yang berbentuk *Relational Database Management System* (RDBMS). Selain RDBMS *database MySQL* juga merupakan *Database Management System* (DBMS) (Saputro, 2012).

Menurut Nugraha, MySQL merupakan *database* yang mampu menangani data yang sangat besar dan bersifat *opensource*, daya tampung ukuran *database* ini hingga *Giga Byte* sehingga sangat cocok digunakan untuk penampungan data pada perusahaan kecil maupun perusahaan besar. Kelebihan MySQL adalah karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *databasenya* sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan yang berskala kecil sampai menengah, MySQL juga bersifat *open source* (Sari et al., 2019).

### 3.9 Alat Bantu Perencanaan UML (*Unified Modeling Language*)

*Unified Modelling Language* (UML) adalah suatu Bahasa yang digunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan alat pengembangan sistem dan juga metodologi pengembangan sistem berbasis orientasi objek (Suendri, 2018).

*Unified Modeling Language* (UML) adalah bahasa pemodelan yang berorientasi objek yang di gunakan merancang sistem yang di mulai dari kelas-kelas dan komponen-komponen sistem. UML merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan *diagram* dan teks-teks pendukung. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggamarkan, membangun dan dokumentasi dari sistem perangkat lunak (Rossa A. S dan M. Salahudin, 2014).

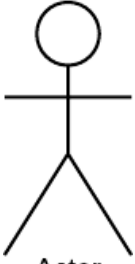
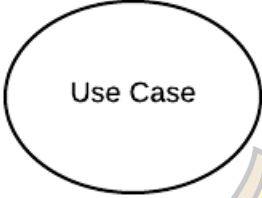



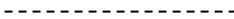
*Unified Modeling Language* (UML) adalah kumpulan notasi grafis yang didukung oleh sebuah meta-model tunggal, yang membantu dalam menjelaskan dan merancang sistem perangkat lunak, khususnya sistem perangkat lunak dibangun menggunakan gaya berorientasi objek. UML terdiri atas banyak elemenelemen grafis yang digabungkan membentuk *diagram*. Tujuan representasi elemen-elemen grafis ke dalam *diagram* adalah untuk menyajikan beragam sudut pandang dari sebuah sistem berdasarkan fungsi masing-masing *diagram* tersebut. Kumpulan dari beragam sudut pandang inilah yang kita sebut sebuah model (Utomo, 2013). UML biasanya di sajikan dalam bentuk *diagram* atau gambar yang meliputi *class* beserta *atribut* dan operasinya, serta hubungan antar *class*.

Pada metode pemodelan UML memiliki beberapa pendukung dalam pembuatan perancangan berorientasi objek diantaranya adalah *use case diagram*, *activity diagram*, *sequence diagram*, dan *class diagram*. Pada pembuatan sistem ini, penulis menggunakan 4 jenis *diagram* diantara lain:

#### a. *Diagram Use Case*

*Diagram Use Case* menggambarkan aktivitas yang dilakukan suatu sistem dari sudut pandang pengamatan luar. Sebuah *use case* yang mempresentasikan sebuah interaksi antara aktor dengan sistem. Simbol-simbol pada *use case* ditunjukkan pada Tabel 3.1.

Tabel 3. 1 Simbol *Use Case Diagram*

Simbol	Deskripsi
 <p>Actor</p>	<p><b>Actor</b></p> <p>Merupakan pengguna sistem yang sedang dijalankan. Sistem yang sedang terintegrasi dengan sistem lain.</p>
 <p>Use Case</p>	<p><b>Use Case</b></p> <p>Merupakan sebuah pekerjaan tertentu, dalam hal ini <i>use case</i> menggambarkan proses dari sistem, contohnya mengkreasi sebuah daftar belanja atau lain sebagainya yang disediakan oleh sistem.</p>
	<p><b>Asosiasi</b></p> <p>Merupakan mengindikasikan apa dan siapa yang meminta interaksi secara langsung.</p>
	<p><b>Generalisasi</b></p> <p>Generalisasi antara <i>actor</i> dan <i>use case</i> yang lebih umum dan <i>use case</i> yang lebih spesifik.</p>
	<p><b>Extend</b></p> <p>Perluasan dari <i>use case</i> lain secara optional menggunakan fungsional yang disediakan.</p>
	<p><b>Include</b></p> <p>Pemanggilan <i>use case</i> oleh <i>use case</i> lain berdasarkan fungsionalitas.</p>




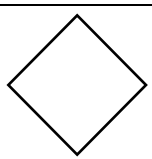


Pada Tabel 3.1 menjelaskan tentang simbol *use case diagram*, simbol-simbol tersebut diantaranya ada *Actor*, *Use Case*, *Asosiasi*, *Generalisasi*, *Extend*, *Include*.

b. *Diagram Activity*

*Diagram Activity* menggambarkan aktivitas pengguna terhadap semua menu yang ada pada sebuah sistem (Suendri, 2018). Berikut simbol-simbol yang ada pada *diagram activity* yang ditunjukkan pada Tabel 3.2.

Tabel 3. 2 Simbol *Activity Diagram*

Simbol	Deskripsi
	<b>Start Point</b> Menggambarkan status awal sistem.
	<b>Stop Point</b> Status akhir yang dilakukan oleh sistem.
	<b>Aktivitas / Activity</b> Menggambarkan suatu proses atau kegiatan bisnis.
	<b>Penggabungan / Join</b> Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	<b>Decision</b> Aktivitas yang tidak dapat dilakukan secara bersamaan.
	<b>Swimlane</b> Digunakan untuk membedakan apa yang dikerjakan oleh <i>actor</i> dan apa yang dikerjakan oleh sistem.







Pada Tabel 3.2 menjelaskan tentang simbol *diagram activity*, simbol-simbol tersebut diantaranya ada *Start Point*, *Stop Point*, *Activity*, *Join*, *Decision*, *Swimlane*.




c. *Diagram Sequence*

*Diagram Sequence* menggambarkan interaksi bagaimana suatu operasi tersebut dilakukan. *Diagram* ini diatur dengan waktu, jadi tau pesan apa yang dikirim dan kapan pelaksanaannya. *Diagram Sequence* terdiri atas dimensi *vertikal* (waktu) dan *horizontal* (objek yang terkait) (Suendri, 2018). Berikut notasi yang ada pada *Diagram Sequence* yang ditunjukkan pada Tabel 3.3.

Tabel 3. 3 Simbol *Sequence Diagram*

Simbol	Deskripsi
	<b>Actor</b> Orang yang sedang berinteraksi dengan sistem.
	<b>Entity Class</b> Kumpulan kelas berupa entitas yang berisi gambaran awal sistem. Biasanya berupa gambaran dari tabel.
	<b>Boundary Class</b> Kumpulan kelas yang menjadi interaksi antara <i>actor</i> dengan sistem. Biasanya berupa tampilan form/report.
	<b>Control Class</b> Menggambarkan proses yang dijalankan oleh sistem.
	<b>Message</b> Simbol untuk pengiriman pesan.
	<b>Activation</b> Eksekusi operasi dari objek.

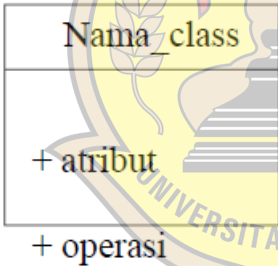
	<p><b><i>Lifeline</i></b></p> <p>Garis titik-titik yang terhubung dengan objek, sepanjang <i>lifeline</i> terdapat <i>activation</i>.</p>
---	---

Pada Tabel 3.3 menjelaskan tentang simbol *sequence diagram*, simbol-simbol tersebut diantaranya ada *Actor*, *Entity Class*, *Boundary Class*, *Control Class*, *Message*, *Activation*, *Lifeline*.

d. *Diagram Class*

*Diagram Class* menggambarkan struktur program dan deskripsi serta objek yang memiliki hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Berikut merupakan simbol-simbol pada *Diagram Class* yang ditunjukkan pada Tabel 3.4.

Tabel 3. 4 Simbol *Class Diagram*

Simbol	Deskripsi
	<p><b>Kelas</b></p> <p>Kelas pada struktur sistem beserta atribut dan operasi yang akan digunakan.</p>

Pada Tabel 3.4 menjelaskan tentang simbol *class diagram*, simbol-simbol tersebut diantaranya ada *Class*, *Attribute*, *Operation*.

*Multiplicity* menentukan/mendefinisikan banyaknya objek yang terhubung dalam suatu relasi. Berikut adalah tabel hubungan antar *class* (*multiplicity*) yang ditunjukkan dengan Tabel 3.5.

Tabel 3. 5 *Multiplicity Class Diagram*

<b><i>Multiplicity</i></b>	<b>Penjelasan</b>
1	Satu dan hanya satu.
0..*	Boleh tidak ada atau satu atau lebih.
1..*	1 atau lebih.
0..1	Boleh tidak ada, maksimal 1.
n..n	Batasan antara. Contoh 2.4.

Pada Tabel 3.5 menjelaskan tentang simbol relasi *multiplicity class diagram*, simbol hubungan relasi *class* tersebut diantaranya ada 1, 0..\*, 1..\*, 0..1, n..n.

### 3.10 Pengujian atau Unit Testing

Pengujian sistem adalah pengujian yang dilakukan pada sistem komputer (*computer-based system*) secara keseluruhan. Pengujian berdasarkan spesifikasi kebutuhan perangkat lunak. Pengujian ini biasanya dilakukan berdasarkan spesifikasi yang dianalisa secara informal dan manual. Pengujian ini juga tidak memiliki metode dan kriteria formal sehingga hasil pengujiannya bisa menjadi tidak konsisten dan rancu. Dukungan alat bantu untuk pengujian ini jarang ditemukan (Hermanto, B. 2019). Aktivitas pengujian terdiri dari satu set langkah untuk dapat menempatkan kasus uji secara spesifik. Kualitas sebuah perangkat lunak dilihat dari kualitas perangkat lunak itu sendiri dan melihat kepuasan pelanggan (Cholifah, Yulianingsih, & Sagita, 2018).

Dalam melakukan pengujian sistem terdapat beberapa teknik pengujian sistem, yaitu :

#### 1. White-Box-Testing (Pengujian Kotak Putih)

*White-Box Testing* yaitu pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur *control* dari desain program secara *procedural* untuk membagi pengujian kedalam beberapa kasus pengujian (MZ, M. K., 2016). Pengujian *Whitebox* dirancang guna mendapatkan kesalahan pada persyaratan fungsional tanpa mengabaikan kerja bagian dalam dari suatu *software* (Ardiansyah, 2017). Untuk mengetahui kesalahan dan kompleksitas pada kode program maka dibutuhkan pengujian *Whitebox* (Ijudin, A., & Saifudin, A., 2020).

Berikut tujuan pengujian *white box*, antara lain sebagai berikut :

- a. Untuk mengetahui cara kerja suatu perangkat lunak secara *internal*.
- b. Meningkatkan ketelitian dalam implementasikan perangkat lunak.
- c. Untuk menjamin operasi *internal* sesuai dengan spesifikasi yang telah ditetapkan dengan menggunakan struktur kendali dari prosedur yang di rancangan.

$$\text{Rumus } V(G) = E - N + 2$$

## 2. *Black-Box-Testing* (Pengujian Kotak Hitam)

*Black-Box Testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan (Setiyani, L. 2019).

Pendekatan pengujian *Black-Box* adalah metode pengujian di mana data tes berasal dari persyaratan fungsional yang ditentukan tanpa memperhatikan struktur program akhir. Karena hanya fungsi dari modul perangkat lunak yang menjadi perhatian, pengujian *Black-Box* juga mengacu pada uji fungsional, metode pengujian menekankan pada menjalankan fungsi dan pemeriksaan inputan dan data output (MZ, M. K., 2016).

Pengujian kotak hitam adalah rencana percobaan yang memperhatikan detail sistem serat aspek dari fungsinya, mengenali jenis-jenis kecacatan fungsi antarmuka, kecacatan di model data serta kecacatan di jalan masuk ke dalam asal data disimpan (Rossa, A. S., & Shalahuddin, M., 2013).

Pengujian kotak hitam (*Black-Box Testing*) berkaitan dengan pengujian – pengujian yang dilakukan pada antarmuka perangkat lunak. Pengujian kotak hitam mengkaji beberapa aspek *fundamental* dari suatu sistem / perangkat lunak dengan sedikit memperhatikan struktur logis internal dari perangkat lunak (Setiyani, L. 2019).

Tujuan *Black-Box Testing* adalah untuk mengetahui kesalahan – kesalahan sistem, antara lain sebagai berikut :

- a. Fungsi yang salah atau hilang.
- b. Kesalahan antarmuka.
- c. Kesalahan dalam struktur data atau akses basis data eksternal.
- d. Kesalahan perilaku atau kinerja.
- e. Kesalahan inisialisasi dan penghentian.