# BAB II
# LANDASAN TEORI

## 2.1 Tinjauan Pustaka

Adapun perbedaan penelitian dengan penelitian terdahulu dapat dilihat dibawah ini:

**Tabel 2.1** Tabel Literatur Review

| No | Judul | Author / Tahun | Metode | Hasil Penelitian |
|---|---|---|---|---|
| 1 | Implementasi *Prototype* Aplikasi Pengelolaan Inventaris Barang | Martono, 2019 | *Prototype* | Hasil dari penelitian ini adalah sebuah aplikasi yang dapat melakukan pengelolaan terhadap data inventaris barang yang dapat memberikan efisiensi dalam hal biaya dan waktu pemrosesan data. |
| 2 | Perancangan Sistem Informasi Persediaan Barang Pada Perusahaan KARYA CIPTA BUANA SENTOSA Berbasis *Web* Dengan Metode *Extreme Programming* | Ressa Priskila, 2018 | *Extreme Programming* (XP) | Hasil dari penelitian ini adalah Perancangan sistem informasi persediaan barang yang menggunakan *Extreme Programming* (XP) membuat pembuatan sistem menjadi lebih cepat dan sesuai dengan kebutuhan klien. |
| 3 | Perancangan Sistem Informasi *Inventory* Barang Menggunakan Metode *Waterfall*, Peranc. Sist. Inf. Invent. Barang Menggunakan Metode. *Waterfall* | D. P. Aji, Sopian, (2021) | *Waterfall* | hasil dari aplikasi sistem informasi *inventory* barang dala hal ini penulis menggunakan model *waterfall* berorientasi objek dengan melakukan perancangan sistem menggunakan *tools* desain UML dengan perancangan dan *tools* desain LRS menjadi lebih dipermudahkan dalam melakukan perancangan |

| | | | | |
|---|---|---|---|---|
| | | | | sistem dan dikemudian hari untuk dilakukan pengembangan lebih lanjut dengan melihat tahapan-tahapan yang harus dilakukan. |
| 4 | Rancang Bangun Aplikasi Inventaris Berbasis *Website* Pada Kelurahan Bantengan | TA Kinaswara, (2019) | *Waterfall* | 1.Aplikasi dapat menambah data barang, data kerusakan barang dan data pengajuan barang.<br>2.Barang dapat dicetak berdasarkan tanggal yang diinginkan sesuai data dari masing-masing barang.<br>3.Aplikasi inventarisasi barang berbasis *website* sudah dapat dijalankan dan memenuhi tujuan awal dari perancangan aplikasi |
| 5 | Rancang Bangun Aplikasi *Inventory Warehouse* Berbasis *Web* (Studi Kasus: TB. Mahkota Bangunan Desa Gandasari) | WN Hamidah, Suhendri (2021) | *Extreme Programming* (XP) | 1.Aplikasi ini dirancang menggunakan bahasa pemrograman PHP.<br>2.Aplikasi ini dibuat berbasis *web*site sehingga dapat diakses kapan saja dan dimana saja (*mobile*, tablet ataupun PC) asal ada *web browser* dan internet.<br>3.Berdasarkan pengujian *blackbox* aplikasi ini berjalan sesuai dengan rancangan dan fungsinya. |

### 2.1.1 Tinjauan Literatur 1

Martono (2019) meneliti tentang Implementasi *Prototype* Aplikasi Pengelolaan Inventaris Barang. Umumnya perusahaan akan menggunakan metode lama dengan memanfaatkan media kertas sebagai opsi utama dalam hal pengelolaan stok barang. Pemanfaatan media kertas ini di nilai kurang efektif dan efisien. Dari segi waktu, karyawan akan membutuhkan lebih banyak waktu dan tenaga untuk mencari suatu data barang. Dari aspek biaya, penyimpanan dengan

media kertas akan memakan lebih banyak ruang. Tingginya resiko kerusakan, kehilangan serta kurangnya akurasi data barang jika disimpan dalam media kertas menjadikan teknologi sebagai solusi atas permasalahan dari pemanfaatan media kertas sebagai media penyimpanan. Tujuan dari penelitian ini adalah untuk menghasilkan sebuah aplikasi pengelolaan inventaris barang yang telah dapat dipergunakan oleh administrator dan operator yang memiliki tugas yang berhubungan dalam pengelolaan inventaris barang sesuai dengan *prototype* aplikasi pengelolaan inventaris barang yang telah penulis buat pada penelitian sebelumnya. Metode penelitian yang digunakan yaitu model pengembangan sistem *waterfall* atau *Systems Development Life Cycle* (SDLC) dengan pemodelan *Unified Modeling Language* (UML) dimana alat yang digunakan untuk menggambarkan model sistem berupa *activity diagram*, *use case diagram*, *component diagram* dan *deployment diagram* serta dalam pembuatan basis data menggunakan *entity relationship digram* (ERD). Hasil dari penelitian ini adalah sebuah aplikasi yang dapat melakukan pengelolaan terhadap data inventaris barang yang dapat memberikan efisiensi dalam hal biaya dan waktu pemrosesan data, aplikasi ini juga dapat meningkatkan keakuratan data karena hanya dapat diakses oleh orang yang memiliki otoritas terhadap data inventaris barang dan mengurangi resiko kehilangan data inventaris barang karena telah disimpan dalam media digital.

### 2.1.2 Tinjauan Literatur 2

Ressa Priskila (2018) meneliti tentang perancangan sistem informasi persediaan barang pada perusahaan Karya Cipta Buana Sentosa berbasis *web* dengan metode *Extreme Programming* (XP). Permasalahan dalam perusahaan ini

masih menggunakan program *Microsoft Excel* dalam pencatatan persediaan barang seperti data barang masuk dan keluar, ketersediaan barang di gudang dan juga dalam penyajian laporan. Masalahnya adalah ketika membutuhkan informasi ketersediaan (stok) dan laporan harus membuka *file* atau tabel satu persatu. Hal ini dirasa tidak efektif dan efisien. Karena itu dibutuhkan suatu sistem informasi yang dapat menunjang kebutuhan informasi perusahaan yang lebih efektif dan efisien dalam pengelolaan persediaan. Sistem informasi tersebut adalah sistem informasi persediaan barang. Pengembangan sebuah perangkat lunak/sistem informasi membutuhkan tahapan-tahapan yang tepat untuk memenuhi kebutuhan pengguna. Penelitian ini menggunakan pendekatan *Agile Software Development* yaitu *Extreme Programming (*XP*)*. Metode *Extreme Programming (*XP*)* dipilih karena pembangunan sistem yang lebih cepat dan sangat fleksibel dengan perubahan yang terjadi pada proses pembangunan perangkat lunak. Artinya selama pembangunan sistem berjalan klien diberikan kesempatan untuk menambahkan atau merubah proses bisnis, sehingga perangkat lunak yang dikembangkan dapat berhasil dan sesuai dengan keinginan pengguna. Tujuan penelitian ini adalah merancang sebuah sistem informasi persediaan (stok) barang berbasis *web* menggunakan metode *Extreme Programming (*XP) untuk mengatasi permasalahan pengelolaan persediaan barang yang ada di perusahaan. Diharapkan dengan adanya sistem informasi ini, pengelolaan persediaan barang menjadi lebih efektif dan efisien, pencarian informasi persediaan/stok dan laporan juga lebih akurat dan cepat.

### 2.1.3 Tinjauan Literatur 3

D. P. Aji, Sopian (2021) meneliti tentang Sistem Informasi *Inventory* Barang Menggunakan Metode *waterfall*. Masalah yang dihadapi adalah pada inventori barang yang diteliti oleh penulis ini dimana proses pencatatan pengeluaran dan pemasukan barang yang dilakukan masih manual dengan masih menggunakan *Microsoft Office Excel* karena karena aplikasi tersebut tidak bisa menyimpan data dan informasi secara terpusat pada suatu *database*. Tujuan untuk mengatasi permasalahan tersebut penulis menggunakan proses yang sebelumnya manual menjadi terkomputerisasi sehingga pencatatan pengeluaran dan pemasukan barang informasi yang dihasilkan lebih tepat dan akurat dengan data ditampilkan secara terpusat sehingga lebih efektif dan efisien. Metode Dalam pengembangan sistem yaitu *Software Development Life Cycle* (SDLC) dengan metode *Waterfall* yang berorientasi objek dan berbasis *web*, perancangan Sistem menggunakan *tools* desain *United Modelling languange* (UML) dengan perancangan *use case diagram*, *activity diagram* dan *sequence diagram*, sedangkan perancangan *database* menggunakan *tools* desain *Logical Record Structure* (LRS).

### 2.1.4 Tinjauan Literatur 4

Pencatatan barang atau inventarisasi memiliki banyak data barang yang perlu di catat, maka dengan adanya aplikasi inventaris berbasis *web* dapat membantu setiap pendataan barang di kantor kelurahan Bantengan. Penelitian menggunakan metode *waterfall* dimana peneliti melakukan analisis kebutuhan, desain sistem, pembuatan sistem, pengujian sistem dan pemeliharaan. Penulisan laporan tugas akhir ini bertujuan untuk menemukan solusi dari permasalahan yang

ada di tempat penelitian. Tujuan dan hasil dari penelitian ini adalah untuk memberikan kemudahan dalam mengelola data inventaris di Kelurahan Bantengan dan bertujuan untuk terciptanya sebuah Aplikasi Inventarisasi Barang.

### 2.1.5 Tinjauan Literatur 5

Sistem yang berjalan di perusahaan bidang *inventory* TB. Mahkota Bangunan masih ada banyak kendala, masih banyak kekurangan dan kelemahan yang dihadapi, salah satunya saat mengirim barang. Data pengiriman barang masuk dan barang keluar hanya ditulis dalam kertas, lalu oleh admin akan disalin kembali ke dalam buku. Penulisan secara manual ini menyebabkan kinerja perusahaan menjadi terhambat dan kurang terkontrol, apabila terjadi kerusakan buku-buku yang berisi data tersebut, dapat berakibat kesalahan dalam laporan. Penelitian ini menggunakan metode *Extreme Programming* (XP) yang terdiri dari 4 tahap *Planning, Design, Coding* dan *Testing*. Metode pengambilan data yang digunakan adalah metode observasi dan metode literatur. Hasil dari penelitian ini berupa Aplikasi *Inventory* barang yang dapat menunjang aktifitas jual beli pada TB. Mahkota Bangunan. Hasil dari pengujian yang dilakukan, aplikasi ini dibuat menggunakan bahasa pemrograman PHP *MySQL* berbasis *web* dan aplikasi ini dapat menyajikan informasi *stock* barang serta laporan data barang yang akurat.

## 2.2 Pengertian Aplikasi

Pengertian aplikasi menurut Jogiyanto (1999:12) adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian sehingga komputer dapat memproses Input menjadi *output*.

.

## 2.3 Pengertian Inventarisasi

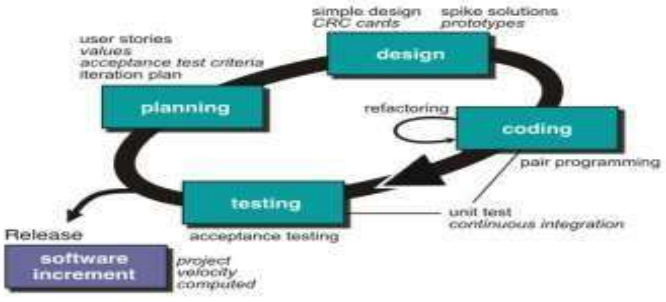
Inventarisasi merupakan suatu kegiatan yang dilakukan untuk mencatat suatu barang keluar dan masuk dan menyusunnya dengan benar yang telah sesuai dengan peraturan yang telah diterapkan (Siswanto dan Khambali, 2018:45).

Budiono (dalam Oktarina, 2015:23) mengatakan bahwa inventarisasi adalah suatu cara dalam mencatat barang yang dilakukan untuk mendaftar setiap barang yang dimiliki kantor untuk dapat dipakai dalam melaksanakan setiap tugas.

## 2.4 Metode Pengembangan *Extreme Programming*

XP adalah metodologi pengembangan perangkat lunak yang ditujukan untuk meningkatkan kualitas perangkat lunak dan tanggap terhadap perubahan kebutuhan pelanggan. Jenis pengembangan perangkat lunak semacam ini dimaksudkan untuk meningkatkan produktivitas dan memperkenalkan pos pemeriksaan di mana persyaratan pelanggan baru dapat diadopsi (Pressman, 2012).

Metode ini membawa unsur-unsur yang menguntungkan dari praktek rekayasa perangkat lunak tradisional ke tingkat "ekstrem", sehingga metode ini dinamai Extreme Programming. Unsur-unsur yang menjadi karakteristik metodologi adalah kesederhanaan, komunikasi, umpan balik, dan keberanian. Gambar tahapan XP dapat dilihat pada gambar 2.1 :




Gambar 2.1 Tahapan Extreme Programming

Tahapan-tahapan dari Extreme Programming terdiri dari :

1. *Planning*

Pada Planning berfokus untuk mendapatkan gambaran fitur dan fungsi dari perangkat lunak yang akan dibangun. Aktivitas *planning* dimulai dengan membuat kumpulan gambaran atau cerita yang telah diberikan oleh klien yang akan menjadi gambaran dasar dari perangkat lunak tersebut. Kumpulan gambaran atau cerita tersebut akan dikumpulkan dalam sebuah indeks dimana setiap poin memiliki prioritasnya masing-masing. Tim pengembang aplikasi juga akan menentukan perkiraaan waktu serta biaya yang dibutuhkan untuk masing-masing indeks.

Setelah semua kebutuhan terpenuhi, tim XP akan menentukan alur dari pengembangan aplikasi sebelum memulai pengembangan tugas.

2. *Design*

Aktivitas *design* dalam pengembangan aplikasi ini, bertujuan untuk mengatur pola logika dalam sistem. Sebuah desain aplikasi yang baik adalah desain yang dapat mengurangi ketergantungan antar setiap proses pada sebuah sistem. Jika salah satu fitur pada sistem mengalami kerusakan, maka hal tersebut tidak akan mempengaruhi sistem secara keseluruhan.

Tahap *Design* pada model proses *Extreme Programming* merupakan panduan dalam membangun perangkat lunak yang didasari dari cerita klien sebelumnya yang telah dikumpulkan pada tahap *planning*. Dalam XP, proses *design* terjadi sebelum dan sesudah aktivitas *coding* berlangsung. Artinya, aktivitas design terjadi secara terus-menerus selama proses pengembangan aplikasi berlangsung.

3. *Coding*

Setelah menyelesaikan gambaran dasar perangkat lunak dan menyelesaikan design untuk aplikasi secara keseluruhan, XP lebih merekomendasikan tim untuk membuat modul unit tes terlebih dahulu yang bertujuan untuk melakukan uji coba setiap cerita dan gambaran yang diberikan oleh klien.

Setelah berbagai unit tes selesai dibangun, tim barulah melanjutkan aktivitasnya kepenulisan *coding* aplikasi. XP menerapkan konsep *Pair Programming* dimana setiap tugas sebuah modul dikembangkan oleh dua orang programmer. XP beranggapan, 2 orang akan lebih cepat dan baik dalam menyelesaikan sebuah masalah. Selanjutnya, modul aplikasi yang sudah selesai dibangun akan digabungkan dengan aplikasi utama.

4. *Testing*

Walaupun tahapan uji coba sudah dilakukan pada tahapan *coding*, XP juga akan melakukan pengujian sistem yang sudah sempurna. Pada tahap *coding*, XP akan terus mengecek dan memperbaiki semua masalah-masalah yang terjadi walaupun hanya masalah kecil. Setiap modul yang sedang dikembangkan, akan diuji terlebih dahulu dengan modul unit tes yang telah dibuat sebelumnya.

Setelah semua modul selesai dan dikumpulkan ke dalam sebuah sistem yang sempurna, maka tim XP akan melakukan pengujian penerimaan atau *acceptance test*. Pada tahap ini, aplikasi akan langsung diuji coba oleh *user* dan klien agar mendapat tanggapan langsung mengenai penerapan gambaran dan cerita yang telah dilakukan sebelumnya.

## 2.5 Pengertian *Website dan Database*

*Website* adalah sebuah *software* yang berfungsi untuk menampilkan dokumen pada suatu *web* yang membuat pengguna dapat mengakses internet melalui

*software* yang terkoneksi dengan internet. (Mara Destiningrum, Q. All, 2017).

Basis data (*database*) adalah suatu kumpulan data yang disusun dalam bentuk tabel-tabel yang saling berkaitan maupun berdiri sendiri dan disimpan secara bersama-sama pada suatu media. Basis data dapat digunakan oleh satu atau lebih program aplikasi secara optimal, data disimpan tanpa mengalami ketergantungan pada program yang akan menggunakannya (Riadi.Muchlisin, 2015).

### 2.6 *Unifield Modeling Language* (UML)

Menurut Rosa dan Shalahuddin (2016) mendefinisakan *Unified Modeling Language* (UML) adalah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek.

#### 2.6.1 *Use Case Diagram*

Menurut Rosa dan Shalahuddin (2016) mendefinisikan *diagram usecase* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

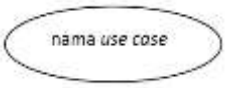
Syarat penamaan pada *usecase* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami (Rosa dan Shalahuddin, 2016). Ada dua hal utama pada *usecase* yaitu pendefinisian apa yang disebut aktor dan *usecase*:

1) Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2) *Usecase* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Rosa dan Shalahuddin (2016), menjelaskan simbol-simbol *usecase* atau *diagram usecase* yang ditampilkan pada tabel 2.2 berikut.

**Tabel 2.2** Simbol *Use Case Diagram*

| No. | Simbol | Keterangan |
|---|---|---|
| 1. | *Usecase*<br>  | 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 *usecase*. |
| 2. | Aktor /*actor*<br>  <br>nama aktor | Orang, proses, atau sistem lain yang Berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, 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. | Asosiasi /<br>*association*<br> | Komunikasi antar aktor dan *usecase* yang berpartisipasi pada *usecase* atau *usecase* memiliki interaksi dengan aktor |
| 4. | Ekstensi /<br><< *extend* >><br>  | Relasi *usecase* tambahan ke sebuah *usecase* dimana *usecase* yang ditambahkan dapat berdiri sendiri walau tanpa *usecase* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *usecase* tambahan memiliki nama depan yang sama dengan *usecase* yang ditambahkan, misal arah panah mengarah pada *usecase* yang ditambahkan; biasanya *usecase* yang menjadi extend-nya merupakan jenis yang sama dengan *usecase* yang menjadi induknya. |
| 5. | Generalisasi/<br>*generalization*<br>  | Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *usecase* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya: Arah panah mengarah pada *usecase* yang menjadi generalisasinya (umum) |
| 6. | Ekstensi /<br><< *include* >><br>  | Relasi *usecase* tambahan ke sebuah *usecase* di mana *usecase* yang ditambahkan memerlukan *usecase* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *usecase*. |

(Rosa dan Shalahuddin, 2016)

### 2.6.2 *Activity Diagram*

*Diagram* aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak (Rosa dan Shalahuddin, 2013). Rosa dan Shalahuddin (2016) menjelaskan simbol-simbol *diagram* aktivitas yang ditampilkan pada tabel 2.3 berikut.

**Tabel 2.3** Simbol *Activity Diagram*

| No. | Simbol | Keterangan |
|---|---|---|
| 1. | Status awal  | Status awal aktivitas sistem, sebuah *diagram* aktivitas memiliki sebuah status awal. |
| 2. | Aktivitas   | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja. |
| 3. | Percabangan / *decision*  | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| 4. | Penggabungan / *join*  | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
| 5. | Status akhir  | Status akhir yang dilakukan sistem, sebuah *diagram* aktivitas memiliki sebuah status akhir. |
| 6. | *Swimlane*   | *Swimlane* memisahkan organisasi bisnis yang bertanggungjawab terhadap aktivitas yang terjadi. |

(Rosa dan Shalahuddin, 2013)

### 2.6.3 *Class Diagram*

Menurut Rosa dan Shalahuddin (2016) mendefinisikan *diagram* kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.

2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Susunan struktur kelas yang baik pada *diagram* kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main, Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem *(view),* Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *usecase (controller),* Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *usecase,* kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. 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.

Rosa dan Shalahuddin (2016) menjelaskan simbol-simbol *diagram* kelas yang ditampilkan pada tabel 2.4 berikut.

**Tabel 2.4** Simbol *Class Diagram*

| No. | Simbol | Keterangan |
|---|---|---|
| 1. | Kelas   | Kelas pada struktur sistem |
| 2. | Antarmuka / *interface*  nama_interface | Sama dengan konsep *interface* dalam pemrograman berorientasi objek |
| 3. | Asosiasi / *association*  | Asosiasi / *association* Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity* |

**Tabel 2.4** Simbol *Class Diagram*

| NO. | Simbol | Keterangan |
|---|---|---|
| 4. | Asosiasi berarah / *directed association*  | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity. |
| 5. | Generalisasi  | Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus). |
| 6. | Kebergantungan / *dependency*  | Relasi antar kelas dengan makna kebergantungan antar kelas. |
| 7. | Agregasi / *Aggregation*  | Relasi antar kelas dengan makna semua bagian *(whole-part).* |

(Rosa dan Shalahuddin, 2013)

## 2.7 Alat Pengembangan *Web*

### 2.7.1 XAMPP

Menurut Buana (2014) XAMPP merupakan perangkat lunak *open source* yang diunggah secara gratis dan bisa dijalankan di semua operasi seperti *windows*, *linux*, *solaris* dan *mac*. XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache* HTTP *Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl.

### *2.7.2 MySQL*

Menurut Kadir (2013) *MySQL* adalah nama *database server*. *Database server* adalah *server* yang berfungsi untuk menangani *database*. *Database* adalah suatu pengorganisasian data dengan tujuan memudahkan penyimpanan dan pengaksesan data. Dengan menggunakan *MySQL*, kita bisa menyimpan data dan kemudian data

bisa diakses dengan dengan cara mudah dan cepat. *MySQL* tergolong sebagai *database* relasional. pada model ini, data dinyatakan dalam bentuk dua dimensi yaitu secara khusus dinamakan tabel, tabel tersusun atas baris dan kolom.

### 2.7.3 *Web Server Apache*

Menurut Nugroho (2013) *Apache* merupakan aplikasi *web server*. Tugas utama apache adalah menghasilkan halaman *web* yang benar kepada *user* berdasarkan kode PHP yang dituliskan oleh pembuat halaman *web*.

### 2.7.4 HTML

Menurut Nugroho (2013) HTML adalah bahasa dasar yang digunakan untuk menyusun halaman *web*. Keberadaannya tetap diperlukan walaupun muncul bahasa seperti PHP ataupun JSP. PHP dan HTML dipakai secara bersama-sama. Dalam hal ini, posisi skrip PHP adalah melekat pada dokumen HTML. Dengan demikian, didokumen HTML bisa disisipkan skrip PHP. Namun, konsekuensinya, dokumen HTML harus disimpan dengan ektensi berupa (dot/.)php.

Berikut Struktur dasar dokumen HTML adalah sebagai berikut:

```
<!doctype html>
<!DOCTYPE html>
<html>
	<head>
		<title> ..... </title>
	</head>
	<body>.....</body>
</html>
```

Di dalam dokumen HTML, tanda < > menyatakan *tag*. *Tag* menyatakan elemen dalam dokumen HTML. Umumnya *tag* berpasangan. Contoh, <head>

berpasangan dengan </head>, dan <body> dengan </body>. Namun, ada pula *tag* yang tidak berpasangan. Sebagai contoh, <br> dn <hr> tidak memiliki pasangan. Penjelasan untuk *tag-tag* HTML pada contoh sebagai berikut:

1) Pasangan <html>...</html> menyatakan awal dokumen HTML.
2) Di dalam <html>...</html> terdapat pasangan <head>...</head> dan <body>...</body>.
3) Pasangan <head>...</head> menyatakan bagian judul dokumen HTML. Isinya paling tidak berupa pasangan <title>...</title>.
4) Isian yang berada pada <title>...</title> menentukan judul dalam *browser*.
5) Pasangan <body>...</body> menyatakan bagian tubuh dokumen. Bagian ini bisa berbagai tag misalnya ... atau <h1>...</h1>.

### 2.7.5 PHP

Menurut Nugroho (2013) mendefinisikan PHP (*Hypertext Preprocessor*) itu bahasa pemograman berbasis *web*. Jadi, PHP itu adalah bahasa program yang digunakan untuk membuat aplikasi berbasis *web*. PHP termasuk bahasa program yang bisa bisa berjalan di sisi *server*, atau sering disebut *Side Server Language*. Jadi, program yang dibuat dengan kode PHP tidak bisa berjalan kecuali dia dijalankan pada *server web*, tanpa adanya *server web* yang terus berjalan dia tidak akan bisa dijalankan.

Sedangkan, Menurut Kadir (2013) mendefinisikan PHP merupakan bahasa pemrograman yang ditunjuk untuk membuat aplikasi *web*. Ditinjau dari pemrosesannya, PHP tergolong berbasis *server side*. Artinya, pemrosesan dilakukan di *server*. Hal ini berkebalikan dengan bahasa seperti *JavaScript*, yang pemrosesannya dilakukan di sisi klient (*client side*). PHP sering dikatakan bahasa

untuk membangun aplikasi *web* dinamis. Pengertian dinamis di sini adalah memungkinkan untuk menampilkan data yang tersimpan dalam *database*. Dengan demikian, halaman *web* akan menyesuaikan dengan isi *database*.

### 2.7.6 *JavaScript*

Menurut Kadir (2013) mendefinisikan *JavaScript* merupakan bahasa pemrogaman yang penggunaannya dilekatkan di dokumen HTML. Kode ditulis di dalam pasangan *tag* <*script*> </*script* >. Berbeda dengan kode PHP yang diproses di *server*, kode *JavaScript* diproses di sisi klient. Karena sifat tersebut, *JavaScript* dapat digunakan misalnya untuk melakukan validasi data di formulir di sisi klien sehingga tidak terjadi komunikasi bolak-balik dengan *server*. Di samping itu, *JavaScript* melalui pendekatan AJAX dapat digunakan untuk mengubah isi sebagian area di halaman *web* berdasarkan data yang diperoleh dari *server*.

Apabila sudah terbiasa dengan PHP, mempelajari *JavaScript* tidaklah sulit karena ada kemiripan-kemiripan. Akan tetapi, hal yang terpenting untuk disadari, jangan sampai mencampuradukkan kode PHP dan *JavaScript* karena keduanya merupakan bahasa yang berbeda. *JavaScript* tidak memerlukan peranti khusus. *Browser* telah menyediakan pemroses *JavaScript*. Oleh karena itu, pengguna dapat langsung membuat kode *JavaScript* tanpa memerlukan *software* tambahan (Kadir, 2013).

### 2.7.7 CSS

Menurut Kadir (2013) *Cascading Style Sheets* (CSS) digunakan untuk mengatur tampilah halaman *web*. Banyak hal yang bisa ditangani oleh CSS, dari mengatur bingkai elemen HTML, penawaran latar belakang yang bergradasi, pembuatan bayangan pada elemen HTML, pengaturan teks, hingga pembuatan

menu. Akan tetapi, halaman *web* yang menarik tentu saja tidak sekedar dibentuk dengan CSS, namun juga dipadu dengan kode *JavaScript* atau *jQuery* untuk mendapatkan efek-efek tertentu. Dapat dikatakan bahwa hampir semua halaman *web* turut melibatkan CSS. Oleh karena itu, memahami CSS perlu dilakukan bagi pengembang *website*.

### 2.7.8 *Framework CodeIgniter*

Menurut (Sallaby & Kanedi, 2020) mengatakan bahwa *CodeIgniter* adalah sebuah framework yang dibuat menggunakan bahasa pemrograman PHP yang 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 *Model-View-Controller* (MVC), yang merupakan model konsep modern saat ini.

#### 2.7.8.1 *Model-View-Controller* (MVC)

Metode *Model-View-Controller* (MVC) terdapat tiga komponen menurut (Yesputra, Rolly, Marpaung Nasrun, 2018) , yaitu :

1. *Model*, mengelola basis data (RDBMS) seperti MySQL ataupun Oracle RDMS. Model berhubungan dengan database sehingga biasanya dalam model akan berisi class ataupun fungsi untuk membuat (*create*), melakukan pembaruan (*update*), menghapus data (*delete*), mencari data (*search*), dan mengambil data (*select*) pada *database*. Selain itu juga model akan berhubungan dengan perintah-perintah query sebagai tindak lanjut dari fungsi-fungsi (*create, update, delete, select*).

2. *View*, bagian *User Interface* atau bagian yang nantinya merupakan tampilan untuk *end-user*. *View* bisa berupa halaman HTML, CSS, Javascript, JQuery dan AJAX, karena metode yang dipakai merupakan MVC sehingga *view* tidak boleh terdapat pemrosesan data ataupun pengaksesan yang berhubungan dengan *database*, sehingga *view* hanya menampilkan data-data hasil dari *Model* dan *Controller*.
3. *Controller*, penghubung antara *view* dan *model*, maksudnya ialah karena *model* tidak dapat berhubungan langsung dengan *view* ataupun sebaliknya, jadi *controller* inilah yang digunakan sebagai jembatan keduanya. Sehingga tugas *controller* ialah sebagai pemrosesan data atau Alur *Logic* Program, menyediakan *variable* yang akan ditampilkan di *view*, pemanggilan *model* sehingga *model* dapat mengakses *database*, *error handling* validasi atau *check* terhadap suatu *input* data.

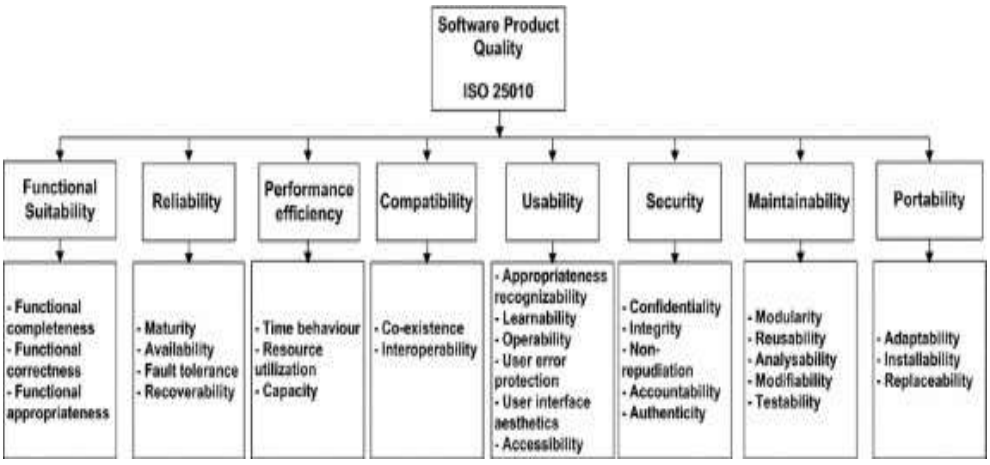
Kesimpulan dari pengertian di atas bahwa *CodeIgniter* adalah *Framework* PHP yang di dalamnya terdapat fitur lengkap aplikasi *web* yang sudah dikemas menjadi satu.

## 2.8 Pengujian Sistem

### 2.8.1 Pengujian *ISO 25010*

Model *ISO-25010* merupakan bagian dari *Software product Quality Requirements and Evaluation* (*SQuaRE*), yang merupakan pengembangan dari model kualitas perangkat lunak sebelumnya yaitu *ISO-9126*. Dalam model *ISO-25010* ini digunakan untuk melihat kualitas suatu perangkat lunak yang digunakan oleh perusahaan, instansi ataupun organisasi. Metode *ISO 25010* ini dapat

digunakan untuk mengevaluasi kualitas sistem perangkat lunak secara spesifik berdasarkan dua dimensi umum, yaitu dimensi *product quality*, dimana prosesnya mengacu pada karakteristik intrinsik dari sebuah produk perangkat lunak, memiliki beberapa elemen antara lain meliputi *functional suitability, reliability, operability, performance efficiency, security, compatibility, maintainability* dan *transferability. Quality in use* dan *product quality*. Sedangkan pada *dimensiquality in use*, terdapat beberapa karakteristik relatif yang ditinjau dari perspektif *user* antara lain *Usability in use, Flexibility in use,* dan *Safety* Adapun untuk mengetahui gambaran kualitas *system*, penulis melakukan analisis berdasarkan model *ISO-25010* yang terdiri dari dua dimensi umum, yaitu dimensi *product quality* dan dimensi *quality in use* (Abran et al., 2008) Adapun dimensi yang pertama terdapat beberapa faktor elemen diantaranya :




**Gambar 2.2** Model kualitas produk *ISO/IEC 25010*Sumber: (Kurniawan, Arifianto, and Muharom 2018)

1. *Functionality* (Fungsionalitas). Kemampuan perangkat lunak untuk Merupakan tingakatan dimana perangkat lunak dapat menyediakan fungsionalitas yang dibutuhkan ketika perangkat lunak digunakan pada kondisi spesifik tertentu dalam hal ini perangkat lunak dapat memenuhi kelayakan dari sebuah fungsi

untuk melakukan pekerjaan yang spesifik bagi pengguna dan dapat memberikan hasil yang tepat dan ketelitian terhadap tingkatan kebutuhan pengguna.

2. *Reliability* Merupakan tingkatan dimana perangkat lunak dapat bertahan pada tingkatan tertentu ketika digunakan oleh pengguna pada kondisi yang spesifik dalam hal ini perangkat lunak dapat beroperasi dan siap ketika dibutuhkan untuk digunakan dan juga dapat bertahan pada tingkat kemampuan tertentu terhadap kegagalan, kesalahan serta perangkat lunak kembali pada tingkat tertentu dalam mengembalikan pengembalian data yang disebabkan kegagalan atau kesalahan pada perangkat lunak.

3. *Performance efficiency* Merupakan tingkatan dimana perangkat lunak dapat memberikan kinerja terhadap sejumlah sumber daya yang digunakan pada kondisi tertentu dalam hal ini *performance efficiency* dapat memberikan reaksi dan waktu yang dibutuhkan ketika melakukan aksi dari sebuah fungsi dan perangkat lunak dapat menggunakan sejumlah sumber daya ketika melakukan aksi dari sebuah fungsi.

4. *Maintainability* Merupakan tingkat dimana sebuah perangkat lunak dapat dimodifikasi. Dalam hal ini modifikasi adalah pebaikan, perubahan atau penyesuaian perangkat lunak untuk dapat berubah pada lingkungan, kebutuhan dan fungsionalitas yang spesifik. Selain itu perangkat lunak dapat dianalisi untuk mengetahui apa yang menyebabkan kegagalan pada perangkat lunak untuk mengidentifikasi bagian yang dapat dimodifikasi.

**2.9 Skala Pengukuran**

Skala pengukuran yang digunakan adalah skala *Likert*, skala yang didasarkan pada penjumlahan sikap responden dalam merespon pernyataan berkaitan indikator-indikator suatu konsep atau variable yang sedang diukur (Sanusi 2012).

Skala *Likert* umumnya menggunakan lima titik dengan label netral pada posisi tengah (ketiga).Skala *Likert* apat dilihat pada Tabel 2.5.

**Tabel 2.5** Skala *Likert*

| Jawaban | Skor |
|---|---|
| Sangat Setuju | 5 |
| Setuju | 4 |
| Netral | 3 |
| Tidak Setuju | 2 |
| Sangat Tidak Setuju | 1 |

*Sumber :* ***(Sanusi, 2012)***

Hasil penilaian responden akan dihitung *persentase* kelayakannya dengan menggunakan perhitungan, dapat dilihat dibawah ini.

$$\text{Persentase} = \frac{\text{Skor Aktual (f)}}{\text{Skor Ideal (n)}} \text{ X } 100\%$$

Persentase kelayakan yang diperoleh kemudian dibandingkan dengan Tabel konversi yang berpedoman pada acuan konversi nilai, dapat dilihat pada Tabel 2.6.

**Tabel 2.6** Skala Konversi Nilai

| Persentase Pencapaian (%) | Interpretasi |
|---|---|
| 90 - 100 | Sangat Baik |
| 80 - 89 | Baik |
| 60 - 79 | Cukup |
| 30 - 59 | Kurang |
| 0 – 29 | Sangat Kurang |

Sumber : (Sanusi, 2012)