# BAB 2

# TINJAUAN PUSTAKA

## 2. 1. Kepariwisataan

Pariwisata merupakan salah satu industri yang paling cepat perkembangannya, sektor pariwisata ini mampu memberikan kontribusi yang besar untuk perkembangan ekonomi, penyerapan tenaga kerja hingga pengurangan kemiskinan dan berbagai dampak positif lain. Berdasarkan Undang Undang Republik Indonesia Nomor 10 Tahun 2009 Tentang Kepariwisataan.

Kepariwisataan merupakan bagian integral dari pembangunan nasional yang dilakukan secara sistematis, terencana, terpadu, berkelanjutan dan bertanggung jawab dengan tetap memberikan perlindungan terhadap nilai-nilai agama, budaya yang hidup dalam masyarakat, kelestarian dan mutu lingkungan hidup serta kepentingan nasional.

Pembangunan kepariwisataan diperlukan untuk mendorong pemerataan kesempatan berusaha dan memperoleh manfaat serta mampu menghadapi tantangan perubahan kehidupan lokal, nasional dan global.Berikut ini penjabaran dari beberapa pengertian yang berkaitan dengan kepariwisataan berdasarkan Undang Undang Republik Indonesia Nomor 10 Tahun 2009 Tentang Kepariwisataan.

### 2.1.1. Pariwisata

Pariwisata adalah berbagai macam kegiatan wisata dan didukung berbagai fasilitas serta layanan yang disediakan oleh masyarakat, pengusaha, Pemerintah dan Pemerintah Daerah.

### 2.1.2. Wisata

Wisata adalah kegiatan perjalanan yang dilakukan oleh seseorang atau sekelompok orang dengan mengunjungi tempat tertentu untuk tujuan rekreasi, pengembangan priadi, atau mempelajari keunikan daya tarik wisata yang dikunjungi dalam jangka waktu sementara.

### 2.1.3. Destinasi Wisata

Daerah tujuan wisata yang selanjutnya disebut destinasi pariwisata adalah kawasan geografis yang berada dalam satu atau lebih wilayah administratif yang di dalamnya terdapat daya tarik wisata, fasilitas umum, fasilitas pariwisata, aksesibilitas, serta masyarakat yang saling terkait dan melengkapi terwujudnya kepariwisataan.

## 2. 2. *Database*

Secara umum, *database* berarti koleksi data yang saling terakit. Secara praktis , basis data dapat dianggap sebagai suatu penyusunan data yang terukur yang disimpan dalam media pengingat (*hard disk*) yang tujuannya adalah agar data tersebut dapat diakses dengan mudah dan cepat.

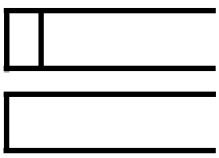
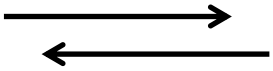
### 2.2.1. *Database Relational*

Ada beberapa macam *database*, antara lain yaitu *database* hierarkis, *database* jaringan, dan *database relasional*. *Database* relasional merupakan *database* yang popular saat ini dan telah diterapkan pada berbagai *platform*, dari PC hingga minikomputer atau tablet.Sebuah *database* relasional tersusun atas sejumlah tabel yang berkolerasi satu dengan lainnya.Sebagai contoh, *database* akademis misalnya mencakup tabel-tabel seperti dosen, mahasiswa, KRS, nilai dan lain sebagainya.

### 2.2.2. *Data Flow Diagram*

*Data Flow Diagram* (DFD) merupakan alat bantu yang dapat digunakan untuk menggambarkan aliran data informasi dan transformasi (proses) data dimulai dari pemasukan data sampai menghasilkan keluaran (*output*) data untuk memudahkan dalam membuat sebuah aplikasi.

Pada proses awal pengembangan sebuah aplikasi DFD merupakan hal yang sangat penting, gambaran konsep awal dari sebuah sitem dapat dibaca melalui DFD hingga pada akhirnya untuk bisa diterapkan pada proses pengkodean pengambangan sebuah aplikasi, Adapun simbol – simbol dari *Data Flow Diagram* seperti pada Gambar 1.

| Simbol | Keterangan |
|---|---|
|  | *External Entity*, merupakan kesatuan di lingkungan luar sistem yang bisa berupa orang atau sistem lain. |
|  | *Process*, merupakan proses seperti perhitungan aritmatika penulisan suatu formula atau pembuatan laporan |
|  | *DataStore* (Simpulan Data), dapat berupa suatu file atau database pada sistem komputer atau catatan manual |
|  | *Data Flow* (arus data), arus data ini mengalir diantara proses, simpan data dan kesatuan luar |

**Gambar 1.** Simbol *Data Flow Diagram*

### 2.2.3. *MySQL Database*

MySQL, merupakan aplikasi *database server* (*server* yang melayani permintaan terhadap *database*). Dalam perkembangannya disebut SQL yang merupakan kepanjangan dari *Structured Query Language*. Sebagaimana diketahui, SQL (*Structured Query Language)* merupakan bahasa standar dalam pengaksesan *database relasional*, pengatahuan akan SQL akan memudahkan siapa pun untuk menggunakan MySQL. MySQL dapat digunakan untuk membuat dan mengelola *database* beserta isinya. Kita dapat memanfaatkan MySQL untuk menambahkan, mengubah, dan menghapus data yang berada dalam *database*.

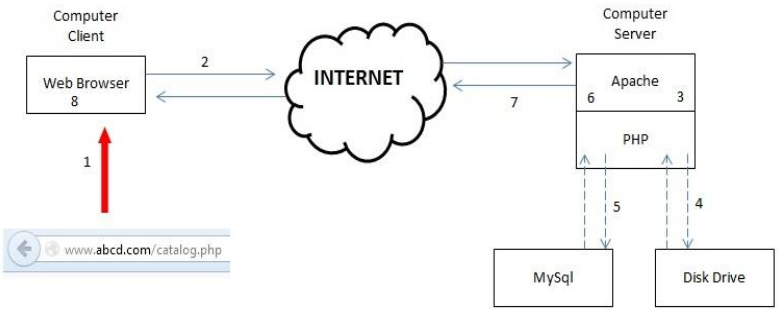
MySQL tersedia pada beberapa *platform* dan tergolong sebagai *database server* yang handal, dapat menangani *database* yang besar dengan

kecepatan tinggi, mendukung banyak sekali fungsi untuk mengakses *database*, dan sekaligus mudah untuk digunakan, bebagai *tool* pendukung juga tersedia. MySQL dapat menangani sebuah tabel yang berukuran *terabyte*, namun ukuran sesungguhnya sangat tergantung pada sistem operasi yang digunakan. MySQL mendukung pengamanan *database* dengan berbagai kriteria pengaksesan. Sebagai gambaran, dimungkinkan untuk mengatur *user* tertentu agar bisa mengakses data yang bersifat rahasia, sedangkan *user* lain tidak boleh.

MySQL juga mendukung konektivitas ke beberapa *software*, sebagai contoh, dengan menggunakan ODBC (*Open Database Connectivity*), *database* yang ditangani MySQL dapat diakses melalui program yang dibuat dengan *Visual Basic*, MySQL juga mendukung program klien yang berbasis *Java* melalui JDBC (*Java Database Connectivity*) selain itu MySQL juga bisa diakses melalui aplikasi berbasis *Web*, misalnya dengan menggunakan PHP.

## 2. 3. *Hypertext Preprocessor*

Bahasa pemrograman PHP merupakan bahasa yang umum digunakan untuk pengembangan aplikasi yang berbasis *Web* dan dapat disematkan didalam HTML serta bersifat *server-side scripting*. PHP memungkinkan kita untuk membuat halaman *web* yang bersifat dinamis. Sistem manajemen basis data yang umum dan sering digunakan bersama PHP adalah MySQL, namun PHP juga mendukung sistem manajement *database Oracle, Microsoft Access, Interbase, PostgreSQL*, dan sebagainya. Cara kerja aplikasi berbasis *web* yang dibuat dengan PHP bisa diilustrasikan seperti pada Gambar 2.




**Gambar 2.** Skema Cara Kerja PHP

Penjabaran dari skema cara kerja PHP pada Gambar 2 adalah sebagai berikut:

1. Pengguna menulis alamat ke dalam *address bar* dari *web browser* (*Internet Explorer, Google Chrome, Mozilla Firefox, Opera,* dan lainnya).
2. *Web browser* mengirimkan pesan diatas ke komputer *server* (www.abcd.com) melalui internet, meminta halaman yang *request.*
3. *Web server* (misal: *Apache*), program yang berjalan di komputer *server*, akan menangkap pesan tersebut, lalu meminta *interpreter* PHP (program lain yang juga berjalan di komputer *server*) untuk mencari *file* yang diminta dalam *disk drive.*
4. *Interpreter* PHP membaca *file* tersebut dari *disk.*
5. *Interpreter* PHP akan menjalankan perintah-perintah atau kode PHP yang ada di *file* tersebut. Jika kode dalam *file* tersebut melibatkan akses terhadap *database* (misal: MySQL) maka *interpreter* PHP juga akan berhubungan langsung dengan MySQL untuk melaksanakan perintah-perintah yang berkaitan dengan *database.*
6. *Interpreter* PHP mengirimkan halaman dalam bentuk HTML ke *Apache*.
7. Melalui *internet*, *Apache* mengirimkan halaman yang diperoleh dari *interperter* PHP ke komputer pengguna sebagai respon atas permintaan yang diberikan.
8. *Web browser* dalam komputer pengguna akan menampilkan halaman yang dikirim oleh *Apache.*

Pada perkembangannya PHP memiliki banyak komunitas pengembang, sehingga munculah beberapa *framework* yang bertujuan untuk memudahkan para pengembang aplikasi berbasis *Web.* PHP memiliki beberapa sintak dasar, bisa kita kategorikan menjadi empat sintak dasar yaitu S*eparator*, *Variable*, *Function*, *Comment.*

### 2.3.1. *Separator*

*Separator* atau pembataspada PHP biasa ditulis dengan sintak `<?PHP...?>.`

### 2.3.2. *Variable*

*Variable* pada PHP biasa diinisialisasi dengan simbol dolar (`$nama_variabel`**),** *variable* adalah tempat untuk menginisialisasi data

sementara atau konstanta di memori yang mempunyai nilai atau data yang dapat berubah – ubah selama proses pengkodean.

### 2.3.3. *Comment*

*Comment* atau komentar pada PHP ini paling umum digunakan untuk meninggalkan sebuah catatan pada kode yang kita buat, untuk penulisan komentar pada PHP memiliki tiga jenis sintak yaitu tanda blok`/*...*/`**,** komentar dua baris `//...` dan tanda pagar `#...` biasa digunakan untuk komentar satu baris.

### 2.3.4. *Function*

*Function* atau fungsi pada bahasa pemrograman PHP, fungsi ini sudah tersedia secara *default* ataupun ekstensi tambahan dari beberapa pengembang kode, dalam beberapa tingkat pengembangan fungsi ini memiliki berbagai konversi penamaan menyesuaikan kebutuhan daripada pegembangnya itu sendiri, sebagai contoh cara penulisan sintaknya seperti pada Gambar 3.

```
function nama_fungsi ($parameter1, $parameter2)
 {
   // kode program fungsi
   Return $nilai_akhir
 }
```

**Gambar 3.** Contoh sintak *function PHP*

Penjabaran dari baris kode Gambar 3 adalah sebagai berikut :

a. Kata `function`adalah instruksi kepada PHP bahwa kita akan membuat sebuah fungsi.
b. `nama_fungsi`adalah nama dari fungsi yang akan ditulis, ini disesuaikan pada kebutuhan kita dalam menuliskan kode program.
c. `$parameter1`**,** `$paramter2`adalah variabel perantara yang akan menyimpan masukan yag diperlukan dalam pemrosesan sebuah fungsi. Pada penggunaan parameter ini sangat menyesuaikan kebutuhan.
d. `return` adalah perintah khusus untuk fungsi, dimana `return` menginstruksikan kepada PHP bahwa pemrosesanfungsi telah selesai.

`return$nilai_akhir` berarti bahwa fungsi akan *"mengembalikan"* `$nilai_akhir` sebagai hasil dari fungsi tersebut.

## 2. 4. *Framework*

*Framework* adalah kerangka kerja, juga dapat diartikan sebagai kumpulan *source code (class* dan *function* atau bias juga disebut *libraries)* yang dapat membantu *programmer* dalam membuat aplikasi atau *website* terutama dalam menangani berbagai masalah seperti koneksi ke *database*, pemanggilan variabel, dan lainya, sehingga akan lebih mempercepat penyelesaian sebuah aplikasi.

### 2.4.1. *Phalcon Framework*

*Phalcon* adalah *framework open source* PHP yang berlisensi dibawah ketentuan Lisensi BSD dan ditulisdalam C-*extension* yang dioptimalkan untuk membuat sebuah aplikasi dengan performa tinggi. *Phalcon* juga menawarkan konsep OOP (*Object Oriented Programming*) atau pemrograman berorientasi objek yang diperlukan untuk mengimplementasikan arsitektur MVC *(Model – View - Controller)* pada aplikasi yang akan dibuat, pola desain seperti ini juga banyak dipakai oleh *framework web* dan aplikasi *desktop* lainnya. Fitur dasar Phalcon *framework* ini diantaranya adalah :

1. ***Low Overhead***yaitu konsumsi memori yang rendah pada CPU jika dibandingkan dengan kerangka kerja tradisional.
   a. Ekstensi Zephir/C dimuat secara bersamaan dengan PHP satu kali pada proses mulai daemon web server
   b. Ekstensi Kelas dan fungsi yang disediakan sudah siap digunakan untuk aplikasi apapun
   c. Kode telah dikompilasi ke platform dan proses tertentu

2. **MVC dan HMVC**, *Phlacon* menggunakan struktur file, skema, dan pola yang sangat mudah untuk dipahami pada kosep berorientasi objek, baik dengan *single module* ataupun dengan *multiple module. Phalcon* tidak memaksakan struktur *file* tertentu untuk pengembangan aplikasi, pengembang dapat menerapkan aplikasi berbasis *phalcon* dengan struktur *file* paling nyaman untuk pengembang gunakan

```
Single/
├── apps
│   ├── controllers
│   │   ├── IndexController.php
│   │   └── ProductsController.php
│   ├── models
│   │   └── Products.php
│   └── views
│       ├── index.phtml
│       └── products
│           ├── index.phtml
│           └── test.phtml
└── public
    └── index.php
```

**Gambar 4.** Contoh Struktur *Directory Single File Phalcon*

```
multiple/
├── apps
│   ├── backend
│   │   ├── Module.php
│   │   ├── controllers
│   │   │   ├── IndexController.php
│   │   │   ├── LoginController.php
│   │   │   └── ProductsController.php
│   │   ├── models
│   │   │   └── Products.php
│   │   └── views
│   │       ├── login
│   │       │   └── index.phtml
│   │       └── products
│   │           └── index.phtml
│   └── frontend
│       ├── Module.php
│       ├── controllers
│       │   ├── IndexController.php
│       │   ├── ProductsController.php
│       │   └── UsersController.php
│       ├── models
│       │   └── Products.php
│       └── views
│           ├── index
│           │   └── index.phtml
│           └── products
│               └── index.phtml
└── public
    └── index.php
```

**Gambar 5.**Contoh Struktur *Directory Multiple File Phalcon*

3. ***Dependency Injection***, *Phalcon* dibagun untuk menjadi *powerful framework*tetapi mudah dipahami dan menggunakan pola yang disebut dengan "*Dependency Injection*". Menginialisasi dan menggunakan hampir dimana saja selama aplikasi berlangsung.

```
//Create the Dependency Injector Container

$di = new Phalcon\DI();

//Register Class, functions, components

$di->set(“request”, new Phalcon\Http\Request());

..

//Use Anywhere else in code

$request = $di->getShared(‘Request’);
```

**Gambar 6.** Contoh Penulisan *Dependency Injection*

4. **REST** (*Representational State Transfer*), merupakan standar arsitektur komunikasi berbasis *web* yang sering digunakan dan diterapkan dalam pegembangan layanan berbasis *web*. Umunya menggunakan HTTP sebagai protokol untuk komunikasi data.

```
Use Phalcon\Mvc\Micro;

$app = new Micro();

//Returning data in JSON

$app->get(
       ‘/check/status’,
       function()
    {
      return $this->response->setJsonContent(
      [
        ‘status’ => ‘important’
      ]);
    }
  );
```

**Gambar 7.** Contoh Penulisan *REST*

5. ***Autoloader***, merupakan fungsi yang dipanggil secara otomatis oleh PHP ketika fitur *autoloading* diaktifkan.

```
Use Phalcon\Loader;

//Create the autoloader

$loader = new Loader();

//Register some namespaces

$loader->RegisterNamespaces(
       [
```

```
        'Example\Base'    => 'vendor/example/base/',
        'Example\Adapter' => 'vendor/example/adapter/',
        'Example'         => 'vendor/example/',
      ]
);

//Register autoloader

$loader->register();
```

**Gambar 8.** Contoh Penulisan *Autoloader*

6. ***Router***, sebagai pengatur pintu masuk yang berupa *request* pada apalikasi , mereka memilah dan mengolah *request url* untuk kemudian diproses sesuai dengan tujuan akhir url tersebut.

```
//Create the router

$router = new Phalcon\Mvc\Router();

//Define a route

$router->add(
      '/admin/users/my-profile',
      [
        'controller' => 'user',
        'action'     => 'profiles',
      ]
);
```

**Gambar 9.** Contoh Penulisan *Router*

Gambar 4 dan Gambar 5 merupakan struktur *directory file* dasar yang sangat sederhana untuk *framework phalcon. Phalcon* juga memiliki beberapa versi yang *compatible* dengan lintas *platform*diantaranya *Windows*, *Linux*, *Mac*, *FreeBSD*.

Beberapa keunggulan *Phalcon* apabila dibandingkan dengan *framework* lainnya adalah :

a. Penggunaan *framework* ini telah menjadi suatu kebutuhan dalam pengembangan *web* profesional dengan PHP, *framework* ini menawarkan filosofi terstruktur yang dengan mudah mempertahankan sedikit koding dan membuat pekerjaan lebih menyenangkan.
b. *Open Source*, dengan *Phalcon*, anda bebas untuk menggunakan *framework* secara keseluruhan ataupun hanya beberapa sebagai komponen tambahan.

c. *Low Resource*, optimasi kinerja *database* menjadi ringan dengan kinerja maksimum dengan menggunakan ORM C untuk aplikasi berbasi MVC
d. *High Perfomance*, *Phalcon* menawarkan*framework* yang *powerfull*, sangat cepat dimana fitur ini memungkinkan pengembang untuk berkonsentrasi pada pembuatan aplikasi mereka dengan tepat, *Phalcon* dapat memberikan lebih banyak fungsi dengan konsumsi memori yang irit.

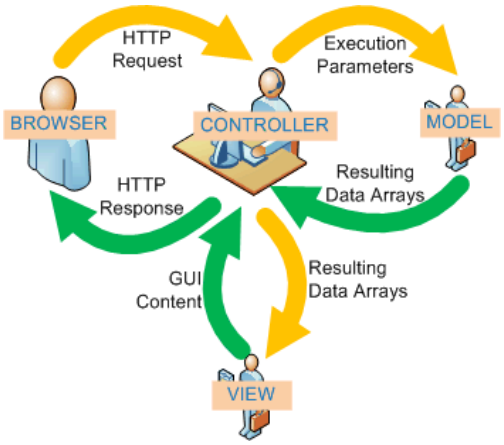
### 2.4.2. MVC *(Model – View - Controller)*

MVC, merupakan kepanjangan dari *Model, View, Controller.* Sebagian *framework* PHP memiliki konsep dasar ini, dimana hal ini akan memudahkan para pengembang aplikasi untuk menyelesaikan sebuah *project.* Konsep MVC pada penerapannya akan memisahkan komponen utama dalam membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna dan bagian yang menjadi kontrol utama aplikasi.

***Model***, merupakan fungsi – fungsi yang berhubungan langsung dengan *database* untuk memanipulasi data, seperti memasukkan data, pembaharuan data, hapus data, dan lain-lain, namun tidak dapat berhubungan langsung dengan bagian *View.*

***View***, merupakan bagian yang berfungsi untuk mengatur tampilan ke pengguna. *View* berfungsi untuk menerima dan merepresentasikan data kepada pengguna, pada suatu aplikasi berbasis *website* bagian ini biasanya berupa *file* HTML yang diatur oleh bagian *controller*, bagian *view* tidak memiliki akases langsung pada bagian *model.*

***Controller***, merupakan bagian yang mengatur keseluruhan hubungan antara bagian *model* hingga *view*, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses sebuah aplikasi.




**Gambar 10.** Cara Kerja Konsep MVC

Manfaat MVC diantaranya adalah meliputi pemisahan *business logic* dari *user interface* dan lapisan *database*, ini membuatnya semakin jelas memudahkan dalam perbaikan dan penyelesaian dalam pengkodean sebuah aplikasi. Sebuah aplikasi yang menggunakan konsep MVC dapat terdiri dari beberapa *triads* yang masing-masing bertanggung jawab atas elemen UI (*user interface*) yang berbeda. Cara kerja konsep MVC bisa terlihat pada Gambar 10.

## 2. 5. *Bootstrap*

*Bootstrap* adalah sebuah alat bantu untuk membuat sebuah tampilan halaman *website* yang dapat mempercepat pekerjaan seorang pengembang *website* ataupun pendesain halaman *website*. Sesuai namanya, *website* yang dibuat dengan alat bantu ini memiliki tampilan halaman yang sama atau mirip dengan tampilan halaman *twitter* atau desainer juga dapat mengubah tampilan halaman *website* sesuai dengan kebutuhan.

*Bootstrap* dibangun dengan teknologi HTML dan CSS yang dapat membuat *layout* halaman *website*, tabel, tombol, *form*, navigasi, dan komponen lainnya dalam sebuah *website* hanya dengan memanggil fungsi CSS (*class*) dalam berkasHTML yang telah didefinisikan. Selain itu juga terdapat komponen-komponen lainnya yang dibangun menggunakan *JavaScript*.

### 2.5.1. *Hyper Text Markup Language* (HTML)

HTML, kependekan dari *Hyper Text Markup Language*. Dokumen HTML adalah *file* teks murni yang dapat dibuat dengan *text editor* apapun, dokumen ini dikenal sebagai *web page*. HTML umumnya berisi informasi atau *interface* aplikasi dalam internet, HTML merupakan dokumen yang disajikan dalam *browser web surfer.*

Dalam penulisan sintak HTML ada tiga istilah utama yang diantaranya adalah *Element, Tag,* dan *Attribute.* HTML memiliki struktur dasar dalam penulisannya yang harus dipatuhi, pada Gambar 6 adalah contoh struktur dasar dari aturan penulisan sintak HTML.

```
<html>
  <head>
     <title>...</title>
  </head>
  <body>
    ...
  </body>
</html>
```

**Gambar 11.** Struktur Penulisan dasar HTML

Penjabaran dari baris kode diatas adalah sebagai berikut :

a. ***Element DOCTYPE***, digunakan untuk memberikan informasi kepada *browser* mengenai versi HTML yang digunakan oleh dokumen.
b. ***Element* HTML**, ini mengandung keseluruhan dokumen HTML, yang berarti *Tag* pembuka *Element* HTML yang merupakan tanda awal dokumen HTML.
c. ***Element Head,*** pada dokumen HTML digunakan untuk menguraikan meta data (informasi yang berkaitan dengan dokumen HTML), judul dokumen, tautan dokumen ke berkas-berkas eksternal. Berbagai data yang ada di dalam *Elemen head* tidak akan tampak pada halaman *website* hasil tampilan *browser.*
d. ***Element Title,*** berfungsi untuk memberi judul pada dokumen.
e. ***Element Body,*** merupakan penampung dari isi konten dokumen yang akan ditampilkan kepada pengguna.

#### 2.5.1.1. *Element*

*Element* HTML, merupakan komponen yang menetapkan peran sebuah objek dalam dokumen, termasuk struktur dan konten objek tersebut, beberapa contoh elemen HTML diantaranya `p, b, h1, div, spam, em, strong,` dan masih banyak yang lainnya.

#### 2.5.1.2. *Tag*

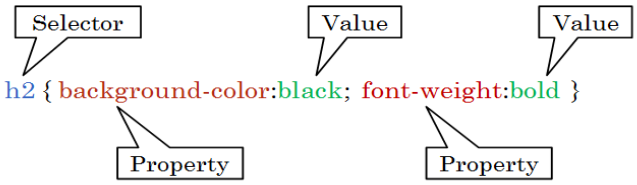
*Tag*, sebuah *Element* HTML biasanya direpresentasikan oleh *Tag*. *Tag* pembuka menandakan *Element* tersebut dimulai, dan *Tag* penutup menandakan akhir dari sebuah *Element.Tag* pembuka direpresentasikan dengann nama *element* yang diapait simbol `<...>,`contoh `<p>` , *Tag* penutup dituliskan dengan menambahkan garis miring `/` setelah simbol `<`, contoh `<p>...</p>` .

#### 2.5.1.3. *Attribute*

*Attribute* merupakan informasi tambahan yang dapat kita berikan untuk sebuah *Element*. Setiap *Element* HTML memiliki *Attribute* yang berbeda-beda. *Attribute* standar yang dimiliki oleh semua *Element* sendiri merupakan *Attribute* yang umumnya dapat diimplementasikan oleh semua *Element*, misalnya *Attribute* `‘id’` untuk indetifikasi *Elemet*, atau `‘class’` untuk klasifikasi *Element*, contoh `<a href=``http://www.detik.com``> link detikcom </a>` .Kode tersebut memberi contoh *Attribute*`href` yang dimiliki oleh *Element* `a.` *Attribute* ini berguna untuk memberikan referensi *hyperlink* dari sebuah *Element*.

### 2.5.2. *Cascading Style Sheet* (CSS)

CSS merupakan kepanjangan dari *Cascading Style Sheet* adalah baris kode yang dibuat untuk mendesain serta mempercantik tampilan HTML menjadi semakin dinamis. CSS sendiri merupakan bahasa yang deklaratif dan sangat komplek. Ada tiga istilah dasar yang yang digunakan dalam CSS, yaitu *Selector*, *Property*, dan *Value*. Sebagai gambaran dasar penerapannya bisa dilihat pada gambar berikut.




**Gambar 12.** Contoh penggunaan CSS

### 2.5.2.1. *Selector*

*Selector*, adalah sebuah *Tag* HTML yang digunakan pada CSS agar *Element* tersebut dapat dimanipulasi atau ditambahkan gaya pada CSS, *Selector* dapat berupa *Tag* HTML itu sendiri, atau bisa juga berupa nilai dari *Attribute Class* atau *Id* pada *Tag* HTML. Pada Gambar 12, yang dikatakan *Selector* adalah *Element* `h2`, sebuah selector merupakan seluruh kode *Element* yang berada sebelum `{...}`.

*Selector Class*, pada CSS biasa ditulis dengan *prefix* `.` dan pada pemanggilan di sintak HTML biasa ditulis `class=”namaClass”`, dalam satu *Element* HTML bisa memanggil beberapa *Class* dengan spasi sebagai pemisah, pada satu halaman *Selector Class* ini dapat dipanggil berkali-kali.

```
.container {
        Width = 960px;
}
.center-content {
        Text-align: center;
        Color: snow;
        Font-family: Consolas;
}
<div class=”container center-content”> Contoh Selector Class </div>
```

**Gambar 13.** *Selector Class*

*Selector Id*,sama halnya dengan *Selector Class* hanya saja untuk *Selector Id* ini hanya dapat dipanggil sekali pada satu halaman, jika ada lebih dari satu *Selector Id* pada satu halaman makan yang akan dipanggil adalah *Selector Id* yang pertama. Pada CSS biasa ditulis dengan *prefix* `#` dan untuk pemanggilan pada HTML biasa menggunakan sintak `id=”namaID”`.

```
#content {
        Text-align: center;
        Color: snow;
}

<p id=”content”>Contoh Selector Id</p>
```

**Gambar 14.** *Selector Id*

#### 2.5.2.2. *Property*

*Property*, adalah jenis *Style* atau nama untuk menentukan *Style* pada *Tag* HTML, dengan penggunaan *Property* lah sebuah Tag HTML dapat diubah menjadi sifat CSS, misalnya seperti menambahkan *Property* pada `color, background, text-align, font,` dan lain sebagainya. Pada Gambar 12, yang dikatan *Property* adalah kode atau *Element* sebelum tanda `:` (titik dua) yaitu `background-color` dan `font-weight.`

#### 2.5.2.3. *Value*

*Value* pada CSS adalah nilai dari *Property*. Pada Gambar 12, yang dikatakan *Value* adalah `black` dan `bold` atau pada beberapa baris kode lainnya *Value* ini biasa diinisialisasikan dengan berupa kode *hexa* dan ditutup dengan *separator* `;` untuk menandakan bahwa baris kode telah berakhir dan bisa ditambah *Property* dan *Value* baru.

### 2.5.3. *JavaScript*

*JavaScript* adalah bahasa pemrograman yang dijalankan oleh kebanyakan *browser modern*, *JavaScript* mendukung pemrograman berorientasi objek dan pemrograman procedural, *JavaScript* dapat digunakan untuk mengendalikan halaman *web* dari sisi pengguna (*client-side), server-side programs*(dari sisi *server*), dan bahkan aplikasi berbasis *mobile (Mobile Apps)*, bahasa ini biasa digunakan dikombinasi dengan HTML, CSS, dan AJAX.

*JavaScript*adalah teknologi kompetitf untuk *VBScript*, berbeda dengan *Java* yang dikembangkan oleh *Sun Microsystems*, *JavaScript* memiliki teknologi yang sama sekali berbeda, sedagkan *VBScript* dikemangkan oleh *Microsoft* dan perbedaannya adalah *VBScript* bekerja hanya pada *browser Internet Explorer (IE)* sedangkan *JavaScript*

mendukung untuk penggunaan *browser* lain juga, ini menjadikan *JavaScript* sebagai bahsa pilihan untuk aplikasi global dan akhirnya menyingkirkan *VBScript* dari pasar pengembangan *Web.*

Tujuan dari *JavaScript*, bila digunakan dalam pengembangan web adalah memanipulasi elemen halaman *website* dan membuatnya lebih dinamis. *JavaScript*disimpan dalam bahasa *markup* HTML, meskipun memiliki beberapa pilihan tentang cara menyematkan bahasa, itu harus direferensikan dengan cara tertentu. *JavaScript* bisa ditulis langsung di samping *Tag* HTML dengan *prefix* `<script language = “JavaScript”>`, *JavaScript* juga dapat dihubungkan dengan *file* eksternal atau situs *web* eksternal menggunakan sintak `<script src= “my_external_file.js”>`.

## 2. 6. *Recommender Systems* (RSs)

Sistem perekomendasi (*Recommender Systems)* adalah sebuah alat dan teknologi yang menyediakan rekomendasi terkait suatu hal untuk dapat dimanfaatkan oleh *user*. *Recommender System* atau sistem perekomendasi membantu kita dalam mengatasi masalah berupa *information overload* dengan menyediakan saran-saran yang bersifat personal berdasarkan pada *history* perilaku pengguna sebelumnya. Akurasi dari rekomendasi yang dihasilkan oleh sistem tergantung pada algoritma yang digunakan.Tingakt efektifitas dalam sebuah sistem rekomendasi tergantung pada pengenalan kepada pengguna mengenai informasi dari *item* - *item* yang membuat pengguna merasa tertarik dan meyakinkan pengguna untuk mencoba *item* tersebut. Ada beberapa pendekatan dalam membangung dan merancang sebuah sistem perekomendasi yaitu,

*a.* *content-based,* merekomendasikan suatu *item* dengan cara mencari tingkat kesamaan atara *item*yang sebelumnya pernah diberi *like* dengan *item* lain
*b.* *collaborative-filtering,* yaitu merekomendasikan suatu *item*berdasarkan *rate*yang diberikan oleh pengguna lainnya
*c.* *demographic*, yaitu memberikan rekomendasi *item* berdasarkan data sebaran pengguna pada wilayah tertentu
*d.* *knowledge-based*, yaitu suantu sistem rekomendasi berdasarkan domain *expert* untuk menentukan kualitas data-datanya

*e. community-based,* yaitu merekomendasikan suatu *item* berdasarkan preferensi yang dimiliki oleh teman-teman dilingkungannya

*f. hybrid,* yaitu sistem rekomendasi ini mengkombinasikan teknik-teknik yang sudah disebutkan sebelumnya.

Sistem perekomendasi mempunyai karakteristik sebagai salah satu jenis dari *customer decision support system* (DSS) dikarenakan sistem perekomendasi merupakan sebuah sistem informasi yang digunakan dalam pengambilan keputusan dan digunakan untuk mendukung bukan menggantikan manusia dalam pengambilan keputusannya.

Sama seperti jenis DSS yang lain, pada sistem perekomendasi, *customer* menyediakan masukan berupa karakteristik produk yang diinginkan atau berupa *rating* terhadap produk, yang kemudian digunakan oleh sistem untuk membuat rekomendasi bagi *customer* yang bersangkutan. Meskipun demikian, sistem perekomendasi berbeda dengan DSS dalam hal penggunaannya. Pengguna DSS biasanya adalah *level general manager*, atau *analyst* yang memanfaatkan sistem untuk membantu pekerjaan-pekerjaan seperti perencanaan pemasaran, perencanaan logistik, atau perencanaan keuangan, sementara pengguna sistem perekomendasi adalah *customer* yang menghadapi masalah yang disebut sebagai *preferential choice problem*.

### 2.6.1. *Collaborative Filtering* (CF)

*Collaborative Filtering* (CF) merupakan proses penyaringan atau pengevaluasian *item* menggunakan opini dari orang lain. *Collaborative Filteing* (CF) murni menggunakan matriks yang berisi *user-item rating* sebagai satu-satunya *input*, sedangkan *output* yang dihasilkan ada dua jenis yaitu prediksi yang mengindikasikan seberapa besar tingkat kesukaan seorang pengguna terhadap sebuh *item* dan yang kedua adalah sebuah daftar yang berisi *n-item* yang direkomendasikan. Istilah pengguna dalam CF mengacu kepada mereka yang memberi penilaian terhadap *item-item* di dalam sistem, sekaligus nantinya menerima rekomendasi dari sistem.

**Table 1.** Contoh Representasi Matriks *user-item rating*

| User | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 1 | 1 | 1 | 1 | 0 |
| User1 | 1 | 1 | 1 | 0 | 1 |
| User2 | 1 | 1 | 0 | 1 | 0 |
| User3 | 0 | 0 | 1 | 1 | 1 |

*Collaborative Filtering* memiliki dua pendekatan utama yang sangat sering dipakai yaitu :

a. *User-based collaborative filtering* (UCF)
Pada pendekatan *user-based* ini memiliki asumsi bahwa setiap *user* mempunyai kesamaan kritiria dengan *user* lainya. Hal yang paling mendasar dari penggunaan algoritma ini adalah hasil dari setiap rekomendasi disusun berdasarkan *item-item* yang disukai oleh *user*, maka *item* yang direkomendasikan oleh sistem adalah *item-item* menurut apa yang disukai oleh *user* lainnya. Bisa dikatakan bahwa *user* yang memiki kesamaan atribut akan tertarik terhadap *item* yang sama.Algoritma yang sering digunakan antara lain adalah Algoritma Pearson *Correlation Coefficient* (PCC) dan Algoritma *Vector Space Similarity* (VSS).

b. *Item-based collaborative filtering* (ICF)
Pendekatan ini bersandar pada relasi antar *item*, rekomendasi didasarkan atas fakta bahwa seorang *user* cenderung memilih *item* yang sama dengan *item-item* yang telah dipilihnya pada masa lampau.

### 2.6.2. *Pearson Correlation Coefficient*(PCC)

Pendekataan pada *Collaborative Filtering* secara umum baik *user-based* maupun *item-based*, menggunakan secara keseluruhan atau sampel dari basis data *user-item* untuk membangkitkan prediksi. *Pearson correlation coefficient* atau dalam bahasa Indonesia sering kita sebut dengan korelasi *pearson* adalah salah satu rumus yang digunakan untuk mencari hubungan antara dua variabel dan dilambangkan dengan *r*,

perhitungan korelasi ini dapat menghasilkan angka positif (+) dan negatif (-).

Kekuatan hubungan korelasi memiliki nilai 0 sampai dengan 1. Apabila dalam proses perhitungan menghasilkan angka 0 maka menunjukan bahwa tidak ada korelasi antar variabel, sedangkan jika menghasilkan angka 1 maka menunjukan bahwa ada korelasi yang positif atau identik antar variabel. Pada beberapa kasus perhitungan, jika mengasilkan angka yang berinterval antara 0 sampai 1 maka akan memiliki inisiasi korelasi sebagai berikut :

| | | | |
|---|---|---|---|
| 0 | = Tidak Ada Korelasi | 0.50 – 0.75 | = Korelasi Kuat |
| 0.00 – 0.25 | = Korelasi Sangat Lemah | 0.75 – 0.99 | = Korelasi Sangat Kuat |
| 0.25 – 0.50 | = Korelasi Cukup | 1 | = Korelasi Sempurna |

Berikut ini merupakan persamaan dari *Pearson Correlation Coefficient* (PCC) :

$$\boldsymbol{r} = \frac{\sum \boldsymbol{xy} - \frac{(\sum \boldsymbol{x})(\sum \boldsymbol{y})}{\boldsymbol{n}}}{\sqrt{\left(\sum \boldsymbol{x^2} - \frac{(\sum \boldsymbol{x})^2}{\boldsymbol{n}}\right)\left(\sum \boldsymbol{y^2} - \frac{(\sum \boldsymbol{y})^2}{\boldsymbol{n}}\right)}} \qquad \textbf{( 1 )}$$

Keterangan :

$\sum \boldsymbol{xy}$ = Sumasi dari perkalian *rating* x dan y
$\sum \boldsymbol{x}$ = Sumasi dari *ratingitem* x
$\sum \boldsymbol{y}$ = Sumasi dari rating y
*n* = Jumlah *user* yang me-*ratingitem* x & y
*r* = *Pearson Correlation Coefficient*

### 2.6.3. Perhitungan Prediksi

Metode untuk menghitung prediksi pada *user u* untuk *item i* dengan persamaan (2), yang mana sebelum menggunakan persamaan ini kita diharuskan memilih *user* dengan *rating* tertinggi dari masing-masing *item* berdasarkan *user rating.* Setelah menentukan *item-item* tertinggi bersadasarkan *user rating* setelah itu lakukan perhitungan prediksi berdasarkan persamaan (2).

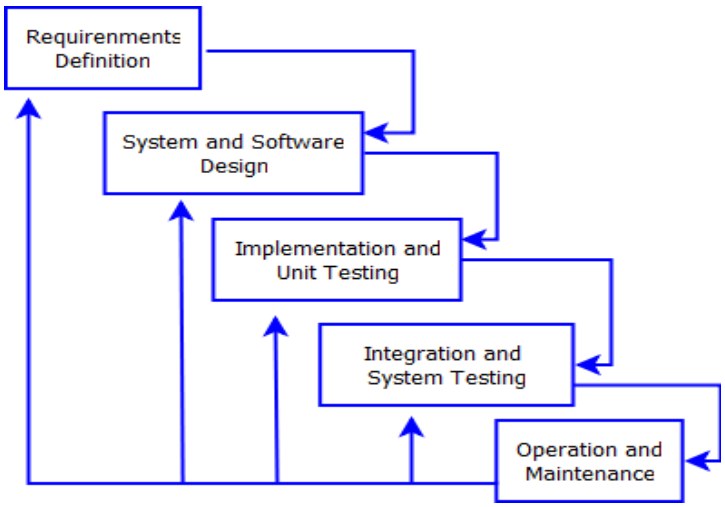
$$P_{(u\,i)} = \frac{\sum_u^n r_{u,i} * W_{u1,u2}}{\sum_u^n W_{u1,u2}} \quad (2)$$

Keterangan :

$P_{(u,i)}$ = Prediksi *rating item i* oleh *user u*
$\sum_u^n r_{u,i}$ = Sumasi *rating user u* terhadap *item j*
$W_{u1,u2}$ = Bobot korelasi antara *user u* dan *user* lainnya
$\sum_u^n W_{u1,u2}$ = Sumasi bobot korelasi antar *user u* dan *user* lainnya

## 2. 7. Metode *Waterfall Process Model*

Model *Waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini sebenarnya "*Classic Life Cycle*" atau "*Sequential Linear Model*". Model ini teramasuk kedalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar Tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai didalam *softawre enginnering (SE).* Disebut dengan waterfall karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan.




**Gambar 15.** *Waterfall Model*

*Waterfall* adalah suatu metodologi pengembangan perangkat lunak yang mengusulkan pendekatan kepada perangkat lunak sistematik dan

sekuensial yang mulai pada tingkat kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Langkah yang harus dilakukan pada metodologi *waterfall* teradapat pada Gambar 15.

### 2. 7. 1. Analisis Kebutuhan perangkat lunak

Pekerjaan dimulai dari pembentukan kebutuhan-kebutuhan untuk seluruh elemen sistem dan kemudian memilah mana yang untuk pengembangan sistem.

### 2. 7. 2. Desain (Pemodelan Sistem)

Proses ini digunakan untuk mengubah kebutuhan kebutuhan diatas menjadi representasi ke dalam bentuk ” blueprint ” *software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya.

### 2. 7. 3. Generasi Kode (Pembuatan Program)

Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain yang telah dibuat harus diubah kedalam bentuk yang dapat dimengerti oleh mesin, yaitu kedalam bahasa pemrograman melalui proses *coding*. Tahap inilah yang mnantinya dikerjakan oleh *programmer.*

### 2. 7. 4. Pengujian (implementasi)

Pada tahap ini, semua fungssi – fungsi *software* harus diujikan, agara *software* bebas dari *error* dan hasilnya benar-benar sesuai dengan kebutuhan yang telah didefinisikan.

### 2. 7. 5. Pemeliharaan

Pemeliharaan suatu *software* diperlukan, termasuk didalamnya adalah pengembangan karena *software* tidak selamanya begitu. Ketika dijalankan kemungkinan terdapat *error* kecil yang tak ditemukan sebelumnya, termasuk penambahan fitur yang belum ada pada *software* tersebut.