BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka merupakan acuan utama dalam beberapa studi yang pernah dilakukan yang berkaitan dengan penelitian ini. Terdapat beberapa penelitian yang digunakan sebagai acuan dalam penelitian ini :

Dian Haryanto Budi (2017) pernah melakukan penelitian tentang System Informasi di Kabupaten Kebumen Berbasis Web. Penelitian ini mengahasilkan informasi tentang biaya masuk tempat wisata dan rute tempat wisata yang ada di Kabupaten Kebumen.

Devi Permatasari (2017) melakukan penelitian tentang System Informasi Geografis Pencarian Obyek Wisata Pantai di Kabupaten Gunungkidul berbasis Web. Penelitian ini mengasilkan informasi jarak dan rute menuju obyek wisata dan pengunjung dan pengunjung juga dapat melakukan pencarian berdasarkan pemilihan nama obyek, menampilkan informasi atm dan spbu terdekat dan jalur alternative.

Rustam Rusdi (2017) melakukan penelitian tentang Perancangan system Informasi Pariwisata dan Pemesanan Paket Wisata Berbasis Web di Pulau Lombok. Penelitian ini menghasilkan Pemesana paket wisata yang mencakup transpormasi dan penginapan di Pulau Lombok.

Rintha Prasetyo Nur Sukma Hendra Ningsih (2011) melakukan penelitian tentang Sistem Informasi Pariwisata Daerah Kabupaten Wonogiri Berbasis Web.

Penelitian ini menghasilkan Pemesana paket wisata yang mencakup transpormasi dan penginapan di Pulau Lombok. info obyek wisata alam, kuliner, sarana dan prasarana yang ada di Kabupaten Wonogiri.

Muhammad Haykal (2020) melakukan penelitian tentang Perancangan Dan Pembuatan Sistem Informasi Wisata Berbasis *Website* Di Dinas Pariwisata Dan Kebudayaan Kabupaten Pidie. Penelitian ini menghasilkan informasi wisata, informasi kuliner, informasi kebudayaan, dan informasi *event* yang ada di Kabupaten Pidie.

I Gede Handika (2018) telah melakukan penelitian tentang Pemanfaatan Framework Laravel Dalam Pembangunan Aplikasi *E-Travel* Berbasis *Website*. Penelitian ini menghasilkan pemesanan tiket pesawat, kereta, hotel. Restaurant dan tiket atraksi/*event*.

Untuk mempermudah perbandingan tinjauan pustaka, maka dapat dilihat pada Tabel 2.1 berikut :

Tabel 2. 1 Perbandingan Penelitian Sebelumnya

No	Penulis	Judul	Keterangan/Hasil
1	Dian Budi Haryanto (2017)	*System* Informasi di Kabupaten Kebumen Berbasis Web	Menambah informasi tentang biaya masuk tempat wisata dan rute tempat wisata
2	Devi Permatasari (2017)	*System* Informasi Geografis Pencarian Obyek Wisata Pantai di Kabupaten Gunungkidul berbasis Web	Menampilkan informasi jarak dan rute menuju obyek wisata dan pengunjung dan pengunjung juga dapat melakukan pencarian berdasarkan pemilihan

			nama obyek, menampilkan informasi atm dan spbu terdekat dan jalur alternative
3	Rustam Rusdi (2017)	Perancangan *system* Informasi Pariwisata dan Pemesanan Paket Wisata Berbasis Web di Pulau Lombok	Pemesana paket wisata yang mencakup transormasi dan penginapan
4	Rintha Prasetyo Nur Sukma Hendra Ningsih (2011)	Sistem Informasi Pariwisata Daerah Kabupaten Wonogiri Berbasis Web	Menampilkan info obyek wisata alam, kuliner, sarana dan prasarana yang ada di Kabupaten Wonogiri.
5	Muhammad Haykal (2020)	Perancangan Dan Pembuatan Sistem Informasi Wisata Berbasis *Website* Di Dinas Pariwisata Dan Kebudayaan Kabupaten Pidie	Menampilkan wisata, informasi kuliner, informasi kebudayaan, dan informasi event yang ada di Kabupaten Pidie.
6	I Gede Handika (2018)	Pemanfaatan *Framework Laravel* Dalam Pembangunan Aplikasi *E-Travel* Berbasis *Website*	Menampilkan pemesanan tiket pesawat, kereta, hotel. Restaurant dan tiket atraksi/event.

Penenlitian yang akan dibuat dengan judul "Informasi Pariwisata di Kabupaten Pangandaran Berbasis Web Menggunakan *Framework Laravel*" menghasihkan informasi tentang destinasi wisata yang dimana terdapat beberapa

kategori destinasi wisata. Ada juga informasi hotel penginapan yang bisa dijadikan tempat referensi untuk menginap, oleh-oleh unik dan khas asli dari Pangandaran. Info kuliner yang bisa dimakan saat mengunjungi Pangandaran. Dan menampilkan info terkait kebudayaan yang ada, mencakup tradisi-tradisi dan kesenian daerah.

2.2 Dasar Teori

Dasar teori digunakan untuk memahami definisi,pengertian dasar dan istilah yang digunakan dalam penelitian ini.

2.2.1 Framework Laravel

Menurut Siena, (2009) *Framework* adalah sekumpulan library yang diorganisasikan pada sebuah rancangan arsitektur untuk memberikan kecepatan, ketepatan, kemudahan dan konsistensi di dalam pengembangan aplikasi dari definisi tersebut". *Framework* terdiri dari:

1. *Model*

   Model mencakup semua proses yang terkait dengan pemanggilan struktur databaik berupa pemanggilan fungsi, input processing atau mencetak output ke dalam browser.

2. *View*

   View mencakup semua proses yang terkait layout output. Bisa dibilang untuk menaruh template interface website atau aplikasi.

3. *Controller*

   Controller mencakup semua proses yang terkait dengan pemanggilan database dan kapsulisasi proses proses utama. Jadi semisal dibagian ini ada file bernamamember.php, maka semua proses yang terkait dengan member

akan dikapsulisasi/ dikelompokan dalam file ini.

Beberapa alasan dari digunakannya *framework* dalam membuat aplikasi adalah sebagai berikut.

a. Aplikasi akan memiliki standar pemograman yang universal.
b. Menghindari *repetitive work.*
c. Memudahkan dalam *team work.*
d. Memudahkan dalam *maintenance* dan pengembangan aplikasi dimasa mendatang.
e. Hemat waktu dan biaya

Laravel merupakan *Framework* PHP yang menekankan pada kesederhanaan dan fleksibilitas pada desainnya. *Laravel* dirilis dibawah lisensi MIT dengan sumber kode yang disediakan di Github. Sama seperti framework PHP lainnya, Laravel dibangun dengan basis MVC (*Model-View-Controller*). Laravel dilengkapi *command line* tool yang bernama "*Artisan*" yang bisa digunakan untuk *packging bundle* dan instalasi *bundle*. *Framework Laravel* dibuat oleh *Taylor Otwell*, proyek *Laravel* dimulai pada April 2011. Awal mula proyek ini dibuat karena *Otwell* sendiri tidak menemukan *framework* yang *up- to-date* dengan versi PHP. Mengembangkan *framewrok* yang sudah ada juga bukan merupakan ide yang bagus karena keterbatasan sumber daya. Dikarenakan beberapa keterbatasan tersebut, *Otwell* membuat sendiri *framework* dengan nama *Laravel*. Oleh karena itu *Laravel* mensyaratkan PHP versi 5.3 keatas. (Rohman, 2014).

Framewok Laravel juga memiliki beberapa keunggulan sebagai berikut:

a. Menggunakan *Command Line Interface* (CLI) *Artisan*.

b. Menggunakan *package manager* PHP *Composer*.

c. Penulisan kode program lebih singkat, mudah dimengerti, dan ekspresif.

Kemudian untuk cara instalasi *framework Laravel* dapat dilakukan dengan 3 cara yaitu:

Melalui *Installer Laravel*.

a. Menggunakan *Composer* dengan mengetikkan perintah *create-project*.

b. Download *source code Laravel* secara lengkap melalui GitHub dengan alamat https://github.com/laravel/laravel/.

Untuk menggunakan *Laravel* versi 5.3 atau diatasnya maka komputer atau server yang digunakan harus memenuhi persyaratan sebagai berikut:

a. PHP >= 5.6.4

b. OpenSSL PHP *Extension*

c. PDO PHP *Extension*

d. *Mbstring* PHP *Extension*

e. *Tokenizer* PHP *Extension*

f. XML PHP *Extension*

Fitur *framework Laravel* yang ditekankan pada penelitian ini adalah *Blade, Migration, Eloquent ORM, Authentification,* dan *Resource Controller*. Berikut adalah penjelasan mengenai lima fitur tersebut:

a. Blade

Blade adalah *template engine*. Pada dasarnya *Blade* adalah *view* namun dengan menggunakan *Blade* akan mempermudah untuk mengatur tampilan *website* dan menampilkan data. Cara untuk membuat file *view* menjadi file *Blade* adalah dengan menambahkan ekstensi .blade.php pada file view. Dan cara untuk memanggil file *Blade* sama dengan cara untuk memanggil file *view* biasa. Contoh program berikut adalah perbandingan antara file *view* biasa dengan file *Blade*.

mahasiswa.php

```
<div id="Mahasiswa">
    <h1>Mahasiswa</h1>
    <?php if(!empty($mahasiswa)): ?>
        <ul>
            <?phpforeach($mahasiswa as $mhs): ?>
            <li><?= $mhs ?></li>
            <?phpendforeach ?>
        </ul>
    <?php else: ?>
    <p>Tidakada data Mahasiswa.</p>
    <?phpendif ?>
</div>
```

mahasiswa.blade.php

```
<div id="Mahasiswa">
    <h1>Mahasiswa</h1>
    @if(!empty($mahasiswa))
        <ul>
            @foreach($mahasiswa as $mhs)
            <li> {{ $mhs }} </li>
            @endforeach
        </ul>
    @else
    <p>Tidakada data Mahasiswa.</p>
    @endif
</div>
```

Dari kedua contoh program diatas dapat disimpulkan jika dengan menggunakan file *Blade* maka penulisan program akan menjadi lebih singkat dan rapi.

b. Migration

Migration adalah fitur yang menyediakan cara baru untuk membuat *database*. Dengan menggunakan *migration* cara membuat *database* melalui

Command Line Interface (CLI) database atau dengan menggunakan aplikasi *database manager* digantikan dengan menggunakan class. Tahapan menggunakan *migration* adalah membuat class kemudian melakukan perintah *migrate* melalui *Command Line Interface* (CLI) *artisan.*

Keuntungan menggunakan *migration* adalalah class yang dibuat bisa dipakai untuk membuat *database* pada berbagai macam *Relation Database Management System* (RDBMS) yang didukung oleh *Laravel*. Sebagai contoh misalnya aplikasi yang digunakan selama ini menggunakan *database MySQL*, kemudian karena alasan pengembangan aplikasi maka akan dilakukan penggantian database ke *PostgreSQL*. Dalam proses penggantian tersebut tidak perlu membuat class lagi, tinggal melakukan perintah migrate melalui *Command Line Interface* (CLI) *artisan.*

Keuntungan lain dari menggunakan migration adalah semua perubahan yang dilakukan pada database akan disimpan pada suatu tabel. Sehingga bisa dilakukan pembatalan (*roolback*) pada *database* jika melakukan perubahan yang tidak benar.

c. Eloquent ORM

Eloquent ORM adalah implementasi dari *ActiveRecord* yang digunakan untuk mengatur relasi antar tabel di *database*. Pada *Eloquent* ORM tabel direpresentasikan dalam bentuk kelas dan data yang tersimpan didalam tabel direpresentasikan dalam bentuk objek. Relasi yang dapat diatur menggunakan *Eloquent* ORM adalah sebagai berikut:

1) *One-to-One* yaitu relasi satu ke satu. Pada relasi ini digunakan *method hasOne* dan *belongsTo*.

2) *One-to-Many* yaitu relasi satu ke banyak. Pada relasi ini digunakan *method hasMany* dan *belongsTo*.
3) *Many-to-One* yaitu relasi banyak ke satu. Pada relasi ini digunakan *method belongsTo* dan *hasMany*.
4) *Many-to-Many* yaitu relasi banyak ke banyak. Pada relasi ini digunakan *method belongsToMany*.

d. Resource Controller

Resource Controller adalah fitur yang digunakan untuk mempercepat pembuatan *controller*. Sebagai contoh misalnya ada *controller* yang menangani semua HTTP *request* terhadap data dosen, untuk membuat *controller* tersebut hanya perlu mengetikkan perintah berikut contoh.

```
php artisan make:controller DosenController –resource
```

Perintah diatas akan menghasilkan *controller* DosenController.php yang disimpan pada folder app/Http/Controllers. Tabel 2.2 dibawah ini adalah daftar action yang dapat dilakukan oleh controller DosenController.php.

Tabel 2. 2 Daftar Action DosenController.php

No	Verb	URI	Action	Route Name
1	GET	/dosen	Index	dosen.index
2	GET	/dosen/create	Create	dosen.create
3	POST	/dosen	Store	dosen.store
4	GET	/dosen/{dosen}	Show	dosen.show
5	GET	/dosen/{dosen}/edit	Edit	dosen.edit
6	PUT/PATCH	/dosen/{dosen}	Update	dosen.update
7	DELETE	/dosen/{dosen}	destroy	dosen.destroy

Pada Table 2.2 diatas menunjukan contoh daftar action pada DosenController.pho yang terdiri dari *Verb, Uri, Action, dan Route Name.*

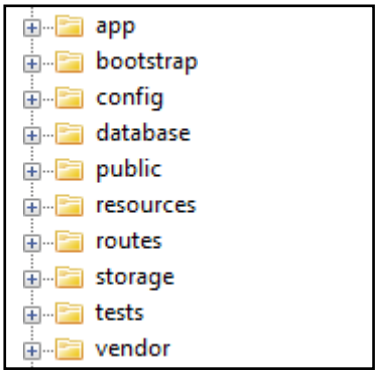
Setelah membuat controller DosenController.php hal yang harus dilakukan selanjutnya adalah membuat satu baris kode program pada route.

```
Route::resource('dosen', 'DosenController');
```

Satu baris kode program pada *route* diatas akan menangani semua *route* untuk melihat, menambah, mengedit, dan menghapus data dosen.

Jadi dapat disimpulkan dengan menggunakan fitur *Resource Controller* dapat mempercepat pembuatan *controller* serta dapat menyederhanakan *route* untuk *controller*.

Struktur folder dari *framework Laravel* 5.3 yang masih default dapat dilihat pada Gambar 2.1 dibawah ini:



Gambar 2. 1 Struktur Folder Laravel

Berikut adalah keterangan pada Gambar 2.1 diatas

1) *Folder app* adalah *folder* yang berisi kode program inti dari aplikasi yang akan dibuat. *Model* dan *controller* tersimpan pada folder ini.
2) *Folder bootstrap* adalah *folder* yang berisi *konfigurasi autoloading* dan terdapat juga *folder cache* yang menyimpan file-file yang dihasilkan secara otomatis oleh *Laravel* untuk mengoptimasi kinerja dari sistem yang dihasilkan.

3) *Folder config* adalah *folder* yang berisi semua *file konfigurasi* aplikasi.
4) *Folder database* adalah *folder* yang berisi *file database migration* dan *seeds.*
5) *Folder public* adalah folder yang berisi file index.php. File tersebut digunakan sebagai *entry point* untuk menangani semua *request* yang masuk ke aplikasi. Pada folder ini juga dapat disimpan beberapa aset dari aplikasi seperti gambar, *JavaScript*, dan *CSS*.
6) *Folder resources* adalah folder yang berisi file *view* dari aplikasi yang dibuat. Selain itu terdapat juga file *language* yang digunakan aplikasi.
7) *Folder routes* adalah folder yang berisi file yang digunakan untuk mendefinisikan semua *route* ke aplikasi. Secara *default* ada tiga file *route* yang disediakan Laravel yaitu api.php, console.php, dan web.php.
8) *Folder storage* adalah folder yang berisi *template Blade* yang dikompilasi, file *session*, file *cache*, dan file lainnya yang dihasilkan secara otomatis oleh *Laravel.*
9) *Folder test* adalah folder yang berisi semua file test yang dibuat untuk aplikasi.
10) *Folder vendor* adalah *folder* yang menyimpan semua *library* yang digunakan.

2.2.2 Unifield Modelling Language (UML)

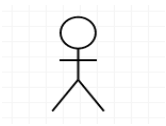
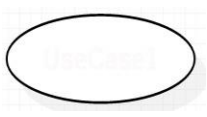
Menurut Adi Nugroho (2010), UML (*Unified Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Pemodelan (*modeling*) sesungguhnya digunakan untuk

menyederhanakan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipahami dan dipelajari. UML juga menyediakan standar notasi ataupun diagram yang digunakan untuk pemodelan *system.*

2.2.2.1 Use Case Diagram

Menurut Adi Nugroho (2010), *Use Case Diagram* digunakan untuk memodelkan fungsionalitas - fungsionalitas sistem/perangkat lunak dilihat dari pengguna yang ada di luar sistem (yang sering dinamakan sebagai aktor). *Use Case* pada dasarnya merupakan unit fungsionalitas koheren yang diekspresikan sebagai transaksi-transaksi yang terjadi antara aktor dan sistem. Kegunaan dari *use case diagram* adalah untuk mendaftarkan aktor-aktor dan *use case – use case* dan memperlihatkan aktor-aktor mana yang berpartisipasi dalam masing-masing *use case.* Notasi-notasi yang digunakan dalam *use case* diagram yang terdapat pada Tabel 2.3 berikut :

Tabel 2. 3 Notasi-notasi Use Case Diagram

NOTASI	KEGUNAAN	SIMBOL
Actor	Menggambarkan semua objek diluar sistem (bukan hanya pengguna sistem/ perangkat lunak) yang berinteraksi dengan sistem yang dikembangkan.	
Use Case	Menggambarkan fungsionalitas yang dimiliki sistem.	

Collaboration	Interaksi aturan-aturan dan elemen lain yang bekerjasama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).	
Assosiation	Lintasan komunikasi antara *actor* dengan *use case*.	
Include	Penambahan perilaku kesuatu *use case* dasar yang secara eksplisit mendeskripsikan penambahan tersebut.	<<include>>
Extend	Penambahan perilaku kesuatu *use case* dasar.	<<extend>>
Generalization	Relasi antara pengklasifikasi yang memiliki deskripsi yang bersifat lebih umum dengan berbagai pengklasifikasi yang lebih spesifik, digunakan dalam struktur pewarisan.	
Dependency	Relasi antar dua elemen model.	

2.2.2.2 Class Diagram

Menurut Adi Nugroho (2010), dalam notasi UML, himpunan kelas- kelas beserta hubungan / relasi / asosiasi antar kelas biasanya digambarkan menggunakan sebuah diagram UML yang dinamakan diagram kelas (*class diagram*). Jika kita perhatikan lebih jauh, sesungguhnya diagram kelas memiliki dua kegunaan / fungsi yang sangat penting, yaitu:

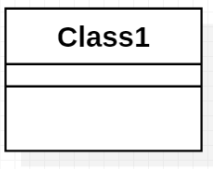
a. Mempresentasikan keadaan statis kelas-kelas yang terlibat dalam sistem.

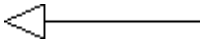
Kelas-kelas ini bisa saja merupakan kelas-kelas dalam bahasa pemrograman dan kelas-kelas persisten yang hadir dalam bentuk tabel-tabel yang ada di sistem basis data relasional.

b. Hubungan antar kelas dalam sistem atau perangkat lunak yang sedang kita kembangkan dapat terlihat dengan mudah.

Notasi-notasi yang digunakan dalam *class diagram UML* terdapat pada Tabel 2.4 berikut :

Tabel 2. 4 Notasi-Notasi Class Diagram

NOTASI	KETERANGAN	SIMBOL
Class	*Class* adalah balok-balok pembangun pada pemrograman berorientasi objek. Sebuah *class* digambarkan sebagai sebuah kotak yang terbagi menjadi 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengahmendefinisikan atribut *class*. Bagian bawah mendefinisikan *method* dari sebuah *class*.	
Composition	Jika sebuah *class* tidak bisa berdiri sendiri dan harus menjadi bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *composition* terhadap *class* tempatnya bergantung tersebut.	

Assosiation	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 *class*, dan dilambangkan oleh sebuah garis yang menghubungkan antar 2 *class*. Garis ini dapat melambangkan tipe – tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship*.	1..n owned by 1
Dependency	Kadang kala class menggunakan *class* yang lain. Hal ini disebut *dependecy*. Umumnya *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain.	
Generalization	Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah kearah "*parent*"-nya / induknya.	

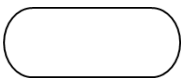
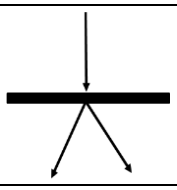
2.2.2.3 Activity Diagram

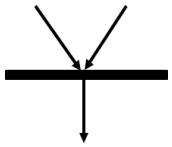
Menurut Adi Nugroho (2010), diagram aktivitas (*activity diagram*) sesungguhnya merupakan bentuk khusus dari *state machine* yang bertujuan untuk memodelkan komputasi-komputasi dan aliran - aliran kerja yang terjadi dalam sistem / perangkat lunak yang sedang dikembangkan. *State* pada diagram aktivitas merepresentasikan *state* dari komputasi yang dieksekusi, bukan *state* dari suatu objek biasa.

Biasanya, suatu diagram aktivitas mengasumsikan komputasi - komputasi dilaksanakan tanpa adanya interupsi-interupsi eksternal berbasis *event* terjadi padanya. Suatu diagram aktivitas memuat di dalamnya *activity state* dimana suatu *activity state* merepresentasikan eksekusi pernyataan dalam suatu prosedur atau kinerja suatu aktivitas dalam suatu aliran kerja. Alih-alih menunggu selesainya atau event seperti yang terjadi pada *state* tunggu, *activity state* menunggu selesainya komputasi. Saat suatu aktivitas selesai maka akan berlanjut ke *activity state* berikutnya yang terlihat pada diagram aktivitas. Penyelesaian transisi dalam suatu diagram aktivitas biasanya akan terpicu saat aktivitas sebelumnya selesai.

Berikut adalah notasi - notasi / simbol-simbol yang digunakan pada *activity diagram* yang terdapat pada Tabel 2.5 dibawah ini :

Tabel 2. 5 Notasi-notasi Activity Diagram

NOTASI	KETERANGAN	SIMBOL
Initial	Titik awal untuk memulai suatu aktivitas.	
Final	Titik akhir untuk mengakhiri aktivitas.	
Activity	Menandakan sebuah aktivitas.	
Decision	Pilihan untuk menggambil keputusan.	
Fork	Menunjukkan kegiatan yang dilakukan secara paralel.	

Join	Untuk menggabungkan beberapa kegiatan secara paralel menjadi satu.	

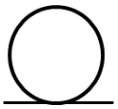
2.2.2.4 Sequence Diagram

Menurut Adi Nugroho (2010), *Sequence Diagram* memperlihatkan interaksi sebagai diagram dua matra (dimensi). Matra vertikal adalah sumbu waktu, waktu bertambah dari atas ke bawah. Matra horizontal memperlihatkan peran pengklasifikasian yang merepresentasikan objek - objek mandiri yang terlibat dalam kolaborasi. Masing - masing pengklasifikasian direpresentasikan sebagai kolom-kolom vertikal dalam sequence diagram yang sering disebut sebagai garis waktu (life line). Selama objek ada, peran digambarkan menggunakan garis tegas. Selama aktivitas prosedur pada objek aktif, garis waktu digambarkan sebagai garis ganda. Pesan-pesan digambarkan sebagai suatu tanda panah dari garis waktu suatu objek ke garis waktu objek lainnya.

Panah-panah menggambarkan aliran pesan antar peran pengklasifikasian digambarkan dalam urutan waktu kejadiannya dari atas ke bawah. Berikut selengkapnya notasi-notasi yang digunakan dalam sequence diagram yang terdapat pada Tabel 2.6 dibawah ini.

Tabel 2. 6 Notasi-notasi Sequence Diagram

NOTASI	KETERANGAN	SIMBOL
Object (Partisipan)	Objek atau biasa disebut partisipan merupakan *intance* dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah *class* (kotak) dengan nama objek di dalamnya yang diawali dengan titikkoma.	
Actor	*Actor* juga dapat bekomunikasi dengan objek, maka *actor* juga dapat diurutkan sebagai kolom.	
Life Line	*Life line* mengindikasikan keberadaan sebuah *object* dalam baris waktu. Notasi untuk *life line* adalah garis putus-putus vertikal yang ditarik dari sebuah *object*.	
Activation	Activation dinotasikan sebagai sebuah kotak persegi empat yang digambarkan pada sebuah life line. Activation mengindikasikan sebuah object yang akan melakukan sebuah aktivasi.	
Self Message	Self message mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.	

Control	Control berhubungan dengan fungsionalitas seperti pemanfaatan sumber daya, pemrosesan terdistribusi, atau penanganan kesalahan.	
Entity	Entity digunakan untuk menangani informasi yang mungkin akan disimpan secara permanen. Entity bisa juga merupakan sebuah tabel pada struktur basis data.	
Message	Message, digambarkan dengan anak panah horizontal antar activation. Message mengindikasikan komunikasi antara objek-objek.	

2.2.3 Pariwisata

Menurut Muljadi, (2012:7) istilah Pariwisata baru muncul di masyrakat kira-kira pada abad ke 18, khususnya sesudah Revolusi Industri di Inggris. Istilah tersebut berasal dari dilaksanakannya kegiatan wisata yaitu suatu aktivitas perubahan tempat tinggal sementara daru seseorang diluar tempat tinggal seharihari dengan suatu alasan apapun selain melakukan kegiatan yang bisa menghasilkan upah atau gaji. Pariwisata merupakan aktivitas pelayanan dan produk industri pariwisata yang mampu menciptakan pengalaman perjalanan bagi wisatawan. Kata Pariwisata berasal dari dua suku kata, yaitu pari dan wisata. Pari yang berarti banyak, berkali-kali, berputar-putar, sedangkan wisata berarti perjalanan atau

berpergian. Jadi pariwisata berarti perjalanan atau berpergian yang dilakukan secara berkali-kali atau berkeliling.

Dalam pengertian kepariwisataan terdapat empat faktor yang harus ada dalam batasan suatu definisi pariwisata. Faktor-faktor tersebut adalah perjalanan itu dilakukan dari satu tempat ke tempat lain, perjalanan itu harus dikaitkan dengan orang-orang yang melakukan perjalanan wisata semata-semata sebagai pengunjung tempat wisata tersebut.

Sedangkan menurut Kusmayadi dan Sugiarto, (2000) Pariwisata merupakan ekspedisi berupa perjalanan yang bertujuan untuk rekreasi, liburan atau bisnis. Sedangkan orang yang melakukan perjalanan rekreasi atau liburan ke tempat-tempat di luar daerah yang sering mereka tempati untuk sementara waktu disebut wisatawan.

2.2.4 Wisata

Menurut Undang-undang Nomor 10 Tahun 2009 tentang Kepariwisataan Bab 1 Pasal 1 dinyatakan bahwa wisata adalah :

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

Wisata berdasarkan jenis-jenisnya dapat dibagi kedalam dua kategori, yaitu:

a. Wisata Alam, yang terdiri dari:
    1) Wisata pantai (*Marine tourism*), merupakan kegiatan wisata yang

ditunjang oleh sarana dan prasarana untuk berenang, memancing, menyelam, dan olahraga air lainnya, termasuk sarana dan prasarana akomodasi, makan dan minum.

2) Wisata Etnik *(Etnik tourism),* merupakan perjalanan untuk mengamati perwujudan kebudayaan dan gaya hidup masyarakat yang dianggap menarik.
3) Wisata Cagar Alam *(Ecotourism),* merupakan wisata yang banyak dikaitkan dengan kegemaran akan keindahan alam, Kesegaran hawa di pegunungan, keajaiban hidup binatang (margasatwa) yang langka, serta tumbuh-tumbuhan yang jarang terdapat di tempat-tempat lain.
4) Wisata Buru, merupakan wisata yang dilakukan di negara-negara yang memang memiliki daerah atau hutan tempat berburu yang dibenarkan oleh pemerintah dan digalakan oleh berbagai agen atau biro perjalanan.
5) Wisata Agro, merupakan jenis wisata yang mengorganisasikan perjalanan ke proyek-proyek pertanian, perkebunan, dan ladang pembibitan di mana wisata rombongan dapat mengadakan kunjungan peninjauan untuk tujuan studi maupun menikmati segarnya tanaman di sekitarnya.

b. Wisata Sosial-Budaya, yang terdiri dari:

1) Peninggalan sejarah kepurbakalaan dan monumen, wisata ini termasuk golongan budaya, monumen nasional, gedung bersejarah, kota, desa, bangunan-bangunan keagamaan, serta tempat-tempat bersejarah lainnya seperti bekas pertempuran *(battle fields)* yang merupakan daya tarik

wisata utama di banyak negara.

2) Museum dan fasilitas budaya lainnya, merupakan wisata yang berhubungan dengan aspek alam dan kebudayaan di suatu kawasan atau daerah tertentu. Museum dapat dikembangkan berdasarkan pada temanya, anatara lain museum arkeologi, sejarah, entologi, sejarah alam, seni dan kerajinan, ilmu pengetahuan dan teknologi, industri, ataupun dengan tema khusus lainnya.

2.2.5 Bootstrap

Menurut Husein Alatas, (2013) *Bootstrap* merupakan *framework* untuk membangun desain web secara *responsif.* Artinya, tampilan web yang dibuat oleh bootstrap akan menyesuaikan ukuran *layer* dan *browser* yang kita gunakan baik dideskto*p*, tablet ataupun *mobile device*. Dengan *bootstrap* kita juga bisa membangun web dinamis ataupun statis.

2.2.6 Website

Menurut Adelheid, Andrea, (2015), *Website* merupakan komponen atau kumpulan komponen yang terdiri dari teks, gambar, suara animasi sehingga lebih merupakan media informasi yang menarik untuk dikunjungi. *Website* adalah halaman informasi yang disediakan melalui jalur internet sehingga bisa diakses di seluruh dunia selama terkoneksi dengan jaringan internet. Secara garis besar, website bisa digolongkan menjadi 2 bagian yaitu website statis dan website dinamis.

2.2.7 MySQL

Menurut Nugroho dalam buku *Database Relasional* dengan MySQL disebutkan bahwa MySQL ialah *software* sistem pengelolaan atau manajemen basis data yang berupa (*Database Management System*-DBMS) yang tersedia secara *open source* (gratis), *software* ini sangat disukai bahkan sangat popular dikalangan pemograman web. *Software* ini dapat memproses data yang berjumlah besar. MySQL merupakan *software* yang bersifat *multi user* dan juga mampu mengirim serta menerima data dengan sangat cepat.

2.2.8 PHP

Menurut Raharjo, Budi, (2009), PHP adalah salah satu bahasa pemrograman skrip yang dirancang untuk membangun aplikasi web. Aplikasi web adalah aplikasi yang disimpan dan dieksekusi (oleh PHP *Engine*) di lingkungan *web server*. Setiap permintaan yang dilakukan oleh *user* melalui aplikasi akan dikembalikan lagi ke hadapan *user*. Dengan aplikasi web, halaman yang tampil di layar web *browser* dapat bersifat dinamis, tergantung dari nilai data atau parameter yang dikirimkan oleh *user* ke *web server.*