BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian serupa terkait pemanfaatan *Laravel* untuk membangun aplikasi dan terkait aplikasi pemesanan lapangan futsal telah banyak dilakukan. Sari dan Wijanarko (2019) membuat sebuah aplikasi penyewaan kamera. Pembuatan aplikasi ini berlatar belakang masalah pada jadwal penyewaan dimana membuat pelanggan merasa kecewa karena adanya pelanggan lain yang memesan pada hari yang sama, sehingga jadwal pada hari tersebut menjadi berbenturan dikarenakan ketidaktahuan pelanggan. Sistem yang dibuat berbasis web dengan menggunakan teknologi *framework Laravel* dan *MySQL*.

Sinaga (2020) membuat sebuah aplikasi reservasi pada restoran. Pembuatan aplikasi ini bertujuan untuk dapat menangani pemesanan tempat dan menu bagi pelanggan, agar mempermudah pelanggan dalam memesan tempat dan menu yang mereka sukai. Aplikasi yang dibuat berbasis web dengan menggunakan *framework Laravel* dan *MySQL*.

Sari (2022) membuat sebuah aplikasi reservasi kamar hotel. Pembuatan aplikasi ini bertujuan agar dapat memberikan kemudahan dan memberikan informasi kamar yang dibutuhkan oleh pelanggan serta pihak pengelola hotel. Aplikasi yang dibuat berbasis web dengan menggunakan *framework Laravel*, *MySQL*, dan *library Bootstrap*.

Tarigan (2023) membuat sebuah aplikasi pemesanan lapangan futsal. Pembuatan aplikasi ini bertujuan untuk membantu meningkatkan efisien pemesanan lapangan dan pengolahan data. Aplikasi yang dibuat berbasis web dengan menggunakan bahasa pemrograman *PHP*, *MySQL* dan *library Bootstrap*.

Tabel 2 1 Perbandingan Penelitian

Penulis	Studi Kasus	Teknologi	Interface	Hasil
Sari dan Wijanarko (2019)	Sistem Informasi Penyewaan Kamera (Studi Kasus Di Rumah Kamera Semarang)	*Laravel, MySQL*	Website	- Penyewaan dan pengembalian kamera - Informasi jadwal penyewaan kamera - Laporan Pendapatan
Sinaga (2020)	Sistem Reservasi pada Restoran Cindelaras Kota Medan	*Laravel, MySQL*	Website	- Reservasi tempat dan pemesanan menu - Pembayaran transfer antar bank - Bukti booking
Sari, dkk (2022)	Pelayanan Reservasi Kamar Berbasis Web Di Renz Hotel Pangkalpinang.	*Laravel, MySQL, Bootstrap*	Website	- Reservasi kamar dan pemesanan menu - Nota reservasi
Tarigan (2023)	Aplikasi Pemesanan Lapangan Futsal Berbasis Web (Studi Kasus: Golden Goal Futsal)	*PHP, MySQL, Bootstrap*	Website	- Pemesanan lapangan futsal - Jadwal lapangan yang tersedia - Sistem pembayaran transfer antar bank
Yang diajukan (2023)	Aplikasi Pemesanan Lapangan Futsal Berbasis Web Di Pelle Futsal	*Laravel, MySQL, Bootstrap*	Website	- Pemesanan lapangan futsal - Melihat informasi jadwal lapangan - Sistem pembayaran transfer antar bank - Nota pembayaran

				- Laporan pendapatan - Aplikasi yang responsive

2.2 Dasar Teori

2.2.1 Bootstrap

Menurut Husein (2013) *Bootstrap* merupakan *framework* ataupun *tools* untuk membuat aplikasi website ataupun situs web *responsive* secara cepat, mudah dan gratis, karena website yang dibangun oleh peneliti merupakan website yang dapat diakses dalam perangkat mobile ataupun *personal computer*. *Bootstrap* terdiri dari *CSS* dan *HTML* untuk menghasilkan *Grid*, *Layout*, *Typography*, *Table*, *Form*, *Navigation*, dan lain lain. Di dalam *Bootstrap* juga sudah terdapat *jQuery plugins* untuk menghasilkan komponen *UI* yang cantik seperti *Transitions*, *Modal*, *Dropdown*, *Scrollspy*, *Tooltip*, *Tab*, *Popover*, *Alert*, *Button*, *Carousel*, dan lain lain.

Untuk mendapatkan gambaran yang lebih baik tentang cara menggunakan *Bootstrap*, berikut ini contoh penggunaan *Bootstrap*.

```
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge"
/>
<meta name="viewport" content="width=device-width,
initial-scale=1" />
<title>Bootstrap Template</title>
<link href="css/bootstrap.min.css" rel="stylesheet"
/>
</head>
<body>
<h1>Hello, world!</h1>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.
11.3/jquery.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

Gambar 2.1 Contoh Penggunaan Bootstrap

`meta charset="utf-8"`

Menyatakan set karakter yang digunakan untuk menulis website. Di sini, UTF-8 mengacu pada *Unicode.*

`meta http-equiv="X-UA-Compatible"`

Menentukan versi *Internet Explorer* yang akan merender halaman. Menggunakan mode *Edge*, konfigurasinya menetapkan mode paling tinggi yang tersedia.

`meta name="viewport"`

Memastikan bahwa halaman memiliki rasio 1:1 dengan ukuran *viewport* (area pandang).

`link href="css/bootstrap.min.css" rel="stylesheet"`

Ini adalah area untuk menambahkan *CSS* inti *Bootstrap*.

```
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.
3/jquery.min.js"
```

Memuat *jQuery* melalui *CDN* Google. Akan lebih baik untuk memuatnya dari *CDN* melalui *HTTP* karena file bisa disimpan selama setahun dalam cache.

```
src="js/bootstrap.min.js
```

Menambahkan *JavaScript* inti *Bootstrap*. Syntax ini harus berada di bawah syntax *jQuery* agar berfungsi dengan baik.

2.2.2 *Web Responsive*

Istilah *Responsive* web desain awalnya dicetuskan oleh Ethan Marcotte dalam sebuah artikelnya di List Apart. Ia mengulas tiga teknik yang telah ada, yakni *flexible grid layout*, *flexible images*, dan *media queries* ke dalam satu pendekatan yang dinamakan *responsive* design. Marcotte dan beberapa ahli lainnya berargumen bahwa metodologi *responsive* yang sebenarnya adalah tidak hanya cukup melakukan perubahan layout sesuai ukuran browser yang mengaksesnya akan tetapi melakukan perubahan total secara keseluruhan terhadap pendekatan yang biasanya dipakai dalam mendesain sebuah web. Dari pada memulai desain pada ukuran layar desktop yang fixed kemudian mengecilkan dan mengatur isinya guna keperluan ukuran yang lebih kecil, maka sebaiknya desain dilakukan pada ukuran viewport yang terkecil terlebih dahulu dan dilanjutkan pada ukuran *viewport* yang lebih besar (Alatas, 2013).

Responsive Web Design (RWD) adalah sebuah metode atau pendekatan sistem web yang bertujuan memberikan pengalaman berselancar yang optimal dalam berbagai perangkat, baik mobile maupun komputer. Ciri-ciri web responsive:

1. Tampilan web di layar *(user interface)* beradaptasi di berbagai perangkat berbeda.
2. Ukuran huruf, tata letak dan gambarnya beradaptasi di berbagai ukuran layar yang berbeda.
3. Umumnya menggunakan *scrolling* dan *swipe* untuk navigasi di perangkat mobile.

2.2.3 *Highcharts*

Highcharts merupakan layanan gratis untuk dapat membuat bagan dan grafik interaktif. *Highcharts* bekerja lintas platform dan dijalankan dari sisi klien, sehingga tidak membutuhkan konfigurasi dari sisi server. Antarmuka yang dibangun sangat interaktif dan mudah untuk dikostumisasi untuk mempresentasikan data dalam berbagai cara (Hønsi, 2013).

2.2.4 *Framework Laravel*

Menurut Aminudin (2015) *Laravel* adalah sebuah *Framework PHP* dirilis dibawah lisensi *MIT* dengan kode sumber yang sudah disediakan oleh *Github*, sama seperti *framework-framework* yang lain, *Laravel* dibangun dengan konsep *MVC (Model-Controller-View)*, kemudian *Laravel* dilengkapi juga *command line tool* yang bernama *Artisan* yang bisa digunakan untuk *packaging bundle* dan instalasi *bundle* melalui *command prompt*.

Berikut ini beberapa fitur yang dimiliki oleh *framework Laravel* menurut Aminudin (2015):

1. *Bundles*

   Bundles yaitu sebuah fitur dengan sistem pengemasan modular dan berbagai *bundle* telah tersedia untuk digunakan dalam aplikasi Anda.

2. *Eloquent ORM*

   Eloquent ORM merupakan penerapan *PHP* lanjutan dari pola *active record* menyediakan metode internal untuk mengatasi kendala hubungan antara objek *database*. Pembangun *query Laravel Fluent* didukung *Eloquent*.

3. *Application Logic*

   Application Logic merupakan bagian dari aplikasi yang dikembangkan, baik menggunakan *Controllers* maupun sebagai bagian dari deklarasi *Route*. Sintaks yang digunakan untuk mendefinisikannya mirip dengan yang digunakan oleh *framework Sinatra*.

4. *Reverse Routing*

   Reverse Routing mendefinisikan hubungan antara link dan *route*, sehingga jika suatu saat ada perubahan pada route secara otomatis akan tersambung dengan link yang relevan. Ketika link yang dibuat dengan menggunakan nama-nama dari *route* yang ada, secara otomatis laravel akan membuat *URI* yang sesuai.

5. *Restful Controllers*

   Restful Controllers memberikan sebuah option (pilihan) untuk memisahkan logika dalam melayani *HTTP GET* dan permintaan *POST*.

6. *Class Auto Loading*

Class Auto Loading menyediakan otomatis loading untuk *class-class PHP*, tanpa membutuhkan pemeriksaan manual terhadap jalur masuknya. Fitur ini mencegah loading yang tidak perlu.

7. *View Composers*

   View Composers adalah kode unit logical yang dapat dijalankan ketika sebuah *view* di load.

8. *IoC Container*

   IoC Container memungkinkan untuk objek baru yang dihasilkan dengan mengikuti prinsip kontrol pembalik, dengan pilihan contoh dan referensi dari objek baru sebagai *Singletons*.

9. *Migrations*

   Migrations menyediakan versi sistem kontrol untuk skema *database*, sehingga memungkinkan untuk menghubungkan perubahan adalah basis kode aplikasi dan keperluan yang dibutuhkan dalam merubah tata letak *database*. Mempermudah dalam penempatan dan memperbarui aplikasi.

10. *Unit Testing*

    Unit Testing mempunyai peran penting dalam *framework Laravel*, dimana unit testing ini mempunyai banyak tes untuk mendeteksi dan mencegah regresi. *Unit testing* dapat dijalankan melalui fitur *artisan command-line*.

11. *Automatic Pagination*

    Automatic Pagination menyederhanakan tugas dari penerapan halaman, menggantikan penerapan yang manual dengan metode otomatis yang terintegrasi ke *Laravel*.

2.2.5 *Model View Controller (MVC)*

Menurut Daqiqil (2011) *MVC* adalah singkatan dari *Model View Controller*. *MVC* sebenarnya adalah sebuah pattern/teknik pemograman yang memisahkan bisnis logic (alur pikir), *datalogic* (penyimpanan data) dan presentation logic (antarmuka aplikasi) atau secara sederhana adalah memisahkan antara desain, data dan proses. Adapun komponen-komponen *MVC* antara lain:

1. *Model*

   Model berhubungan dengan data dan interaksi ke *database* atau web service. *Model* juga merepresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks, file *XML* maupun web service. Biasanya di dalam model akan berisi class dan fungsi untuk mengambil, melakukan update dan menghapus data website. Sebuah aplikasi web biasanya menggunakan basis data dalam menyimpan data, maka pada bagian *Model* biasanya akan berhubungan dengan perintah-perintah *query SQL*.

2. *View*

   View berhubungan dengan segala sesuatu yang akan ditampilkan ke end-user. Bisa berupa halaman web, rss, javascript dan lain-lain. Kita harus menghindari adanya logika atau pemrosesan data di *view*. Di dalam *view* hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. *View* dapat dikatakan sebagai halaman website yang dibuat dengan menggunakan HTML dan bantuan CSS atau JavaScript. Di dalam *view* jangan pernah ada kode untuk

melakukan koneksi ke *basis data*. *View* hanya dikhususkan untuk menampilkan data-data hasil dari *model* dan *controller*.

3. *Controller*

Controller bertindak sebagai penghubung data dan *view*. Di dalam *Controller* inilah terdapat class-class dan fungsi-fungsi yang memproses permintaan dari *View* ke dalam struktur data di dalam *Model*. *Controller* juga tidak boleh berisi kode untuk mengakses basis data karena tugas mengakses data telah diserahkan kepada *model*. Tugas *controller* adalah menyediakan berbagai variabel yang akan ditampilkan di view, memanggil *model* untuk melakukan akses ke basis data, menyediakan penanganan kesalahan/error, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap input