BAB II
KAJIAN LITERATUR

2.1 Tinjauan Pustaka

Pada bab ini berisikan pembahasan teori *framework* MVC, teori model sistem menggunakan UML, pembahasan mengenai Laravel, metode pengembangan *waterfall* dan metode pengujian *Black Box Testing*.

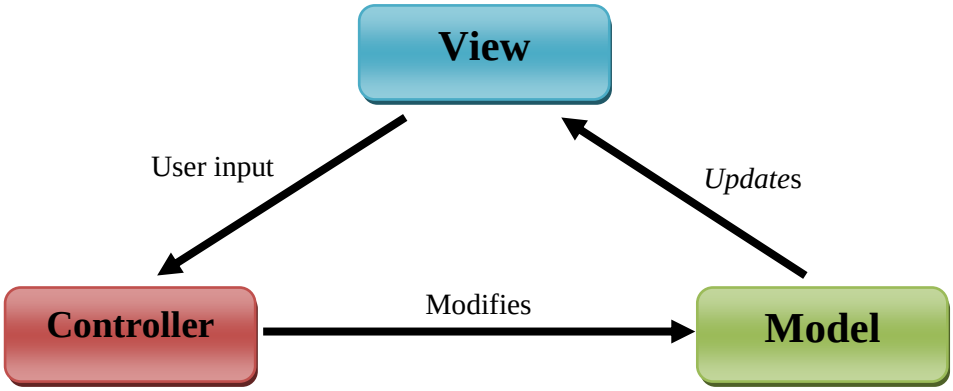
2.1.1 *Framework*

Bedasarkan kamus bahasa Inggris — Indonesia yang di buat oleh Yohanes Aristianto *framework* adalah rangka atau kerangka kerja, disebut kerangka kerja sebab semua hal yang programer lakukan untuk membangun sebuah aplikasi harus sesuai dengan aturan yang ada pada kerangka kerja dan tidak boleh keluar dari kerangka kerja tersebut (S.Sabti, 2012). Menurut Ralph E. Johnson, ketua *UIUC parttens/Software Architecture Group* dan koordinator program proyek senior di *Department of Computer Science pada University of lllinois*, menyatakan bahwa *framework* adalah desain yang *reuseable* yang dinyatakan sebagai suatu set abstraksi *Class* yang mengatur bagaimana *Class* saling terhubung (Johnson, 1991) . *Framework* merupakan kerangka kerja yang bertujuan untuk memudahkan dalam membuat sebuah aplikasi agar dapat dilakukan perubahan dengan cepat dan dapat digunakan kembali dengan aplikasi lainnya yang sejenis. Berdasakan penjelasan tersebut dengan adanya *framework* memungkinkan programer untuk membangun sebuah aplikasi dengan lebih mudah dan efisien dengan *source code* yang di hasilkan lebih rapih untuk pengembangan di kemudian hari (Shalahuddin & Rosa, 2011).

STT-NF

2.1.2 MVC

MVC(*Model View Controller)* merupakan pola atau teknik untuk mengembangkan aplikasi yang bertujuan untuk menghasilkan program yang disiplin dengan cara membaginya menjadi 3 bagian *model, view, dan controller* (Sidik & Betha, 2012), seperti Gambar 1.



Gambar 2.1 Arsitektur MVC

Pada MVC terdapat bagian yang pertama yang disebut dengan *models*, *models* ini berfungsi untuk memanipulasi data *(insert, Update, Delete, search*) berserta aturan bisnisnya (*business process*) seperti proses validasi dan relasi (Warsito & Muhammad, 2014). *Models* juga merupakan bagian dari aplikasi yang mengimplementasi logika untuk data aplikasi, Sehingga dengan adanya *models* maka aplikasi yang dihasilkan akan terstruktur dan data yang tersimpan akan lebih aman karena harus melalui validasi data terlebih dahulu.

Bagian kedua yang disebut dengan *views*, di bagian ini merupakan komponen yang berisikan keseluruhan detail dari tampilan antar muka (*User interface*) untuk pengguna aplikasi seperti *admin*, *members* dan *guest*. Dengan *views* semua proses internal aplikasi akan ditampilkan kepada *User* dan menuntun alur interaksi *User* terhadap aplikasi. *Views* saling terhubung dengan *models* dan me-render isinya ke dalam permukaan layar, apabila terjadi perubahan pada *models* maka secara otomatis *views* akan menggambar ulang bagian layar yang terkena perubahan untuk menunjukan perubahan tersebut(Hidayat & Surarso, 2012).

Bagian terakhir yang disebut dengan *controller*, pada bagian ini terjadi proses inputan dari pengguna dan menginstruksikan *models* dan *views* untuk melakukan aksi berdasarkan masukan tersebut. Sehingga, *controller* bertanggung jawab atas penampungan *event* yang dilakukan oleh *User* (Hidayat & Surarso, 2012). Dicontohkan dalam sebuah kasus pada saat *User* mengklik tombol 'Kirim" dalam sebuah halaman maka *controller* akan menampilakan tampilan yang telah ditentukan oleh *views* dan data yang dihasilkan akan tersimpan di dalam *models*.

Konsep *Model View Controller* (MVC) bertujuan agar sebuah aplikasi dapat mudah diplihara oleh orang-orang di dalam tim pengembangan yangberbeda spesifikasi pekerjaan, misalnya *database administrator* (DBA) untuk mengurusi masalah basisdata, blok *controller* untuk programmer, dan blok *view* untuk desainer antar muka (Sidik & Betha, 2012)

2.1.3 *Software Development Life Cycle* (SDLC)

Software Development Life Cycle (SDLC) merupakan kerangka kerja yang menyediakan urutan pekerjaan dalam membuat perangkat lunak. Tahapan pada proses SDLC pada umumnya adalah persyaratan fungsional perangkat lunak (*User requirement*), perancangan, pengujian, implementasi dan perawatan(Tuteja & Gaurav, 2012). Dengan SDLC membuat pengembangan aplikasi menjadi terstruktur dan metodis karena harus dilakukan secara berurutan. Salah satu tipe SDLC yang paling banyak digunakan adalah *Waterfall*.

Waterfall method merupakan model pengembangan sistem informasi yang sistematik dan sckuensial dengan tujuan yang berbeda untuk setiap fase pengembangannya(Pressman & R.S., 2002). Setelah salah satu fase selesai, maka berlanjut ke fase berikutnya dan tidak bisa melakukan revisi ke fase yang sudah dilakukan. Metode *Waterfall* memiliki tahapan-tahapan sebagai berikut (Sommerville, 2011):

l) *Requirements analysis*

Seluruh kebutuhan *software* harus terkumpul pada fase ini, serta kegunaan *software* yang diharapkan pengguna dan batasan *software* yang didapat dari hasil konsultasi dengan pengguna dan berfungsi sebagai spesifikasi sistem.

2) *Software design*

Tahapan ini dilakukan sebelum programer malakukan *coding.* Tahapan ini bertujuan untuk perancangan sistem mengalokasikan kebutuhan-kebutuhan sistem baik dari perangkat keras serta perangkat lunak dengan membentuk arsitektur secara keseluruhan melalui penggambaran abstraksi sistem dan hubungannya menggunakan *mockup*.

3) *Implementation*

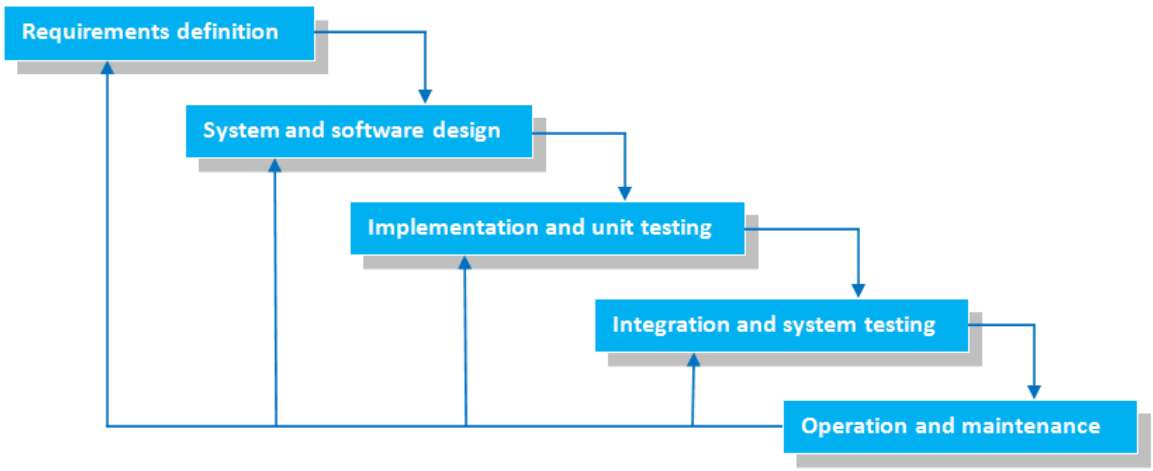
Dalam tahap ini programer melakukan pembuatan aplikasi. Pembuatan aplikasi dipecah menjadi serangkaian program atau unit program yang nantinya akan digabungan dalam tahap selanjutnya. Selain itu dalam taham ini juga dilakukan verifikasi bahwa setiap unit memenuhi spesifikasinya.

4) *Integation and testing*

Di tahapan ini unit-unit program digabungkan dan dilakukan pengujian sebagai sebuah sistem tunggal untuk memastikan apakah aplikasi yang dibuat telah sesuai dengan kebutuhanya atau tidak.

5) *Operation and maintenance*

Ini adalah tahapan terakhir dalam metode *waterfall* dan biasanya (walau tidak selalu) tahapan ini merupakan tahapan yang paling panjang. Aplikasi dijalankan serta dilakukan *miantenance* secara berkala melibatkan perbaikan kesalahan yang tidak ditemukan pada tahap-tahap sebelumnya, meningkatkan implementasi dari unit-unit program serta layanan program sebagai kebutuhan baru.



Gambar 2.2 Metode *Waterfall*

2.1.4 Laravel

Laravel adalah sebuah MVC *web development framework* untuk sebuah pengembangan aplikasi yang diharapkan dapat meningkatkan kualitas aplikasi yang dihasilkan, dengan mengurangi biaya pengembangan dan perbaiakan serta menghasilkan *source code* yang rapih dan fungsional yang dapat mengefesiensikan untuk implementasinya(Widodo & Purnomo, 2016). Laravel merupakan *framework PHP* yang menekankan pada kesederhanaan dan fleksibilitas pada desainnya(Naista, 2016). Laravel merupakan *framework* PHP yang selalu *up-to-date* karena laravel bersifat *open-source* dan dikembangan secara bersama. Laravel memberikan kemudahan untuk berinteraksi dengan *database* yang di sebut *migration*. Dengan adanya *migration*, programer sangat mudah untuk melakukan modifikasi sebuah *database* secara independen karena implementasi skema *database* dipresentasikan dalam sebuah *Class*. Ada beberapa basis data yang mendukung *migration* di laravel seperti (*MySQL, PostgreSQL, MSSQL,* dan *SQLITE*) dan untuk menggunakan *framework* laravel sebelumnya programer harus menguasai metode dalam membuat program dengan istilah OOP(*Object Oriented Programming*).

2.1.4 UML (*Unified Modeling Language*)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem (Dharwiyanti, 2003).

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. UML mendefinisikan diagram-diagram sebagai berikut:

a) *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sangat ditekankan pada "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

b) *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *Class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

c) *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan state diagram khusus, di mana sebagian besar *State* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *State* sebelumnya (*internal processing*). Oleh karena itu *Activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

d) *Sequence Diagram*

Sequence Diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence Diagram* terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence Diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

2.1.6 *Black Box Testing*

Black Box Testing adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan". Kasus uji yang dibuat untuk melakukan pengujian *Black Box Testing* harus dibuat dengan kasus benar dan kasus salah.

Dalam pengujian *Black Box* ini ada beberapa usaha untuk menemukan kesalahan dalam kategori sebagai berikut(Mustaqbal M. S., 2015) :

1. Fungsi-fungsi yang salah atau hilang.
2. Kesalahan *interface.*
3. Kesalahan pada struktur data dan akses basis data.
4. Kesalahan performa.
5. Kesalahan inisialisasi dan terminasi.

2.1.7 *User Acceptance Testing*

User Acceptance Testing (UAT) merupakan sekumpulan urutan langkah pengujian sebuah aplikasi di sisi pengguna, menggunakan format yang telah disepakati bersama, dengan tujuan untuk mengetahui sejauh mana pemahaman pengguna terhadap aplikasi yang disajikan, serta apakah aplikasi telah cukup mampu memenuhi kebutuhan pengguna dan menyelesaikan permasalahan yang terjadi, dengan hasil akhir sebuah dokumen pelengkap pengembangan aplikasi(Rouli, Paulus, & Ridi, 2015).

2.1.8 *Skala Likert*

Skala likert adalah pengukuran skala yang menggunakan beberapa butir pertanyaan untuk mengukur perilaku individu dengan merespon 5 titik pilihan pada setiap butir pertanyaan, yaitu sangat setuju, setuju, tidak memutuskan, tidak setuju, dan sangat tidak setuju (Likert, 1932)

2.1.9 Metode PIECES

Untuk mengidentifikasi masalah, harus dilakukan analisis terhadap kineja, informasi, ekonomi, keamanan, efisiensi, dan pelayanan. Metode ini dikenal dengan PIECES analisis (Performance, Information, Economic, Control, Efeciensy, Service). Dengan metode analisis PIECES ini akan mendapatkan beberapa masalah dan akhirnya dapat ditentukan masalah utamanya.(Fatta, 2007)

2.2 Penelitian Terkait

Pada bagian ini peneliti akan memaparkan penelitian terkait yang memiliki persamaan dengan penelitian yang dilakukan baik dari segi penggunaan teknologi, landasan teori, konsep dan segmentasi pengguna

Table 1 Penelitian Terkait

Nama Peneliti	Judul	Tahun	Kesimpulan
Wahyu Setiadi	Perancangan Sistem Pemesanan *Online* Penggunaan Lapangan Futsal Di Kota Yogyakarta	2013	Bertujuan untuk merancang sistem pemesanan penggunaan lapangan futsal yang mencakup informasi tempat-tempat futsal yang dapat diakses secara mempermudah penyewa dalam melakukan sewa lapangan, mengetahui info jadwal, daftar lapangan futsal, pemesanan lapangan dengan mudah dan cepat. Penelitian ini juga bertujuan untuk membantu pemilik/pengelola futsal dalam mengelola dan mempublikasikan informasi terkait tempat futsalnya.
Hendry dan Teddy Marcus Zakaria	Aplikasi *E-Commerce* Sebagai Jembatan Perancang dan *User* Pada *T-shirt Design*	2017	*T-Shirt Design E-Commerce* mempunyai keunikan, karena produk yang dijual hanya kaos desain dan penjual sekaligus perancang akan menjual kaos dengan motif/gambar yang dibuat sendiri. *T-Shirt Design* adalah aplikasi *website E-Commerce* untuk produk kaos yang menghubungkan perancang sekaligus penjual, dengan *User*.
Beny Bond Banjarnahor dan Kristoko Dwi	Implementasi *Framework* Laravel Pada Sistem Informasi	2017	Bertujuan untuk mengatasi masalah pemesanan lapangan futsal yang dilakukan secara konvensional oleh karena itu di buat aplikasi untuk

Hartomo, M.kom	Pemesanan Penggunaan Lapangan Futsal Berbasis WEB di ZONA6 FUTSAL SEMARANG		membantu petugas dan penyewa dalam melakukan pemesanan lapangan futsal. Penelitian ini menggunakan bahasa pemrograman PHP, dengan *database management* sistem MySQL. Penggunaan bahasa pemrograman PHP 5 didukung dengan *framework* Laravel .
Alvino Gartner	Rancang Bangun Aplikasi *Online* Desain Keramik Lantai Berbasis Web Menggunakan Laravel *Framework*	2018	Penelitian ini menghasilkan aplikasi pemesanan keramik lantai dengan *value customize* dari segi desain dan warna berbasis web, menggunakan bahasa PHP dan *database* MySQL dan didukung menggunakan *framework* laravel 5.2, yang bertujuan untuk memberikan kebebasan kepada *User* keramik lantai untuk menentukan desainnya sendiri.
Haya Rasikhah	Rancang Bangun Aplikasi Berbasis Web Pemesanan Lapangan Olahraga Menggunakan *Framework* Laravel	2020	Penelitian yang membangun aplikasi pemesanan lapangan futsal, badminton dan voli di wilayah depok, menggunakan bahasa PHP dan *database* MySQL dan di dukung mengunakan *framework* laravel, yang bertujuan untuk mengefisienkan waktu proses pemesanan dan mempermudah mendapatkan informasi jadwal dan ketersediaan lapangan.