BAB II

KAJIAN TEORI

Skripsi adalah suatu tugas yang pasti dilakukan mahasiswa untuk menjadi syarat kelulusan dan mendapat gelar sarjana. Terkadang banyak mahasiswa yang membutuhkan waktu lama untuk mengerjakan skripsi tersebut atau menunda kelulusan mereka dengan berbagai macam alasan. Dengan demikian penulis memilih skripsi dan kartu bimbingan skripsi sebagai konten utama untuk pengembangan sistem informasi dengan berbasis *website* menggunakan *framework Codeigniter*. Pengembangan sistem informasi tersebut dilakukan berdasarkan tahapan-tahapan *System Development Life Cycle (SDLC)* yang secara umum digunakan dalam membuat program.

A. Kartu Bimbingan Skripsi

Skripsi adalah suatu karya tulis ilmiah yang dikerjakan mahasiswa tingkat akhir untuk menyelesaikan pendidikan. Karena ini adalah karya tulis ilmiah maka hasil akhirnya merupakan sesuatu yang dapat dipertanggunjawabkan secara ilmiah. Menurut Kamus Besar Bahasa Indonesia (KBBI), skripsi adalah karang ilmiah yang wajib ditulis oleh mahasiswa sebagai bagian dari persyaratan akhir pendidikan akademis. Setiap mahasiswa yang sedang mengerjakan skripsi memiliki kartu bimbingan skripsi.

Kartu bimbingan skripsi adalah sesuatu yang digunakan mahasiswa dan dosen pembimbing selama melakukan bimbingan atau konsultasi. Kartu bimbingan skripsi ini diberikan kepada mahasiswa yang sudah memenuhi persyaratan seperti telah menempuh minimal 120 (Seratus dua puluh) SKS (Satuan Kredit Semester), tidak ada nilai C- pada mata kuliah yang telah ditempuh serta maksimal 2 mata kuliah yang memiliki nilai C. Pada umumnya kartu bimbingan skripsi berisi informasi-informasi

seperti identitas mahasiswa dalam hal ini adalah nama, nomor registrasi, program studi, serta nama dosen pembimbing. Didalam kartu bimbingan skripsi berisikan beberapa aturan perihal skripsi seperti kewajiban mahasiswa, aturan-aturan untuk maju seminar pra skripsi (SPS) dan sidang skripsi. Selain itu terdapat juga sebuah tabel yang berisikan tanggal dan tanda tangan dosen pembimbing, tabel ini diisi setiap kali mahasiswa melakukan bimbingan ke dosen pembimbing. Sehingga dosen pembimbing dan mahasiswa tahu sejauh mana pengerjaan skripsi dilakukan.

Kartu bimbingan skripsi inilah yang digunakan dosen pembimbing (DP) dan mahasiswa untuk memonitor pengerjaan skripsi. Dengan melihat berapa banyak tanda tangan dosen pembimbing menyatakan sejauh mana skripsi mahasiswa dikerjakan. Kartu ini juga sebagai pengingat dosen pembimbing kapan dan apa yang dibahas dalam bimbingan-bimbingan sebelumnya. Jika mahasiswa sudah melakukan bimbingan sebanyak jumlah minimal yang ditentukan, maka mahasiswa tersebut berhak untuk maju SPS dan sidang skripsi. Di Program studi Matematika, Pendidikan Matematika dan Ilmu Komputer, mahasiswa melakukan minimal 4 kali bimbingan dengan masing-masing dosen pembimbing untuk maju SPS serta 8 kali bimbingan untuk maju sidang skripsi.

Disetujui untuk diajukan sidang skripsi pada semester.....

Dosen Pembimbing I, Tanggal :

(.....) _____

Dosen Pembimbing II, Tanggal :

(.....) _____

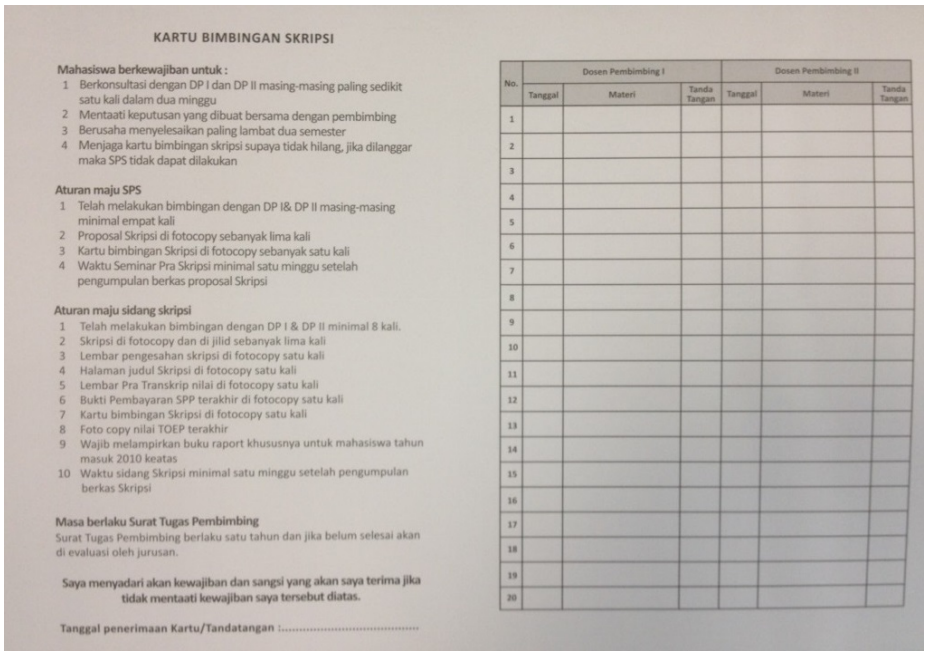
KARTU BIMBINGAN SKRIPSI

Building Future Leaders

Nama : .....
No. Reg : .....
Program Studi : .....
Dosen Pembimbing I : .....
Dosen Pembimbing II : .....

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
JURUSAN MATEMATIKA
Kampus B, Universitas Negeri Jakarta
Jl. Pemuda No. 10, Rawamangun, Jakarta Timur 13220
Telepon. 021-7098082

Gambar 2.1: Tampak Depan Kartu Bimbingan Skripsi Matematika



KARTU BIMBINGAN SKRIPSI

Mahasiswa berkewajiban untuk :
1. Berkonsultasi dengan DP I dan DP II masing-masing paling sedikit satu kali dalam dua minggu
2. Mentaati keputusan yang dibuat bersama dengan pembimbing
3. Berusaha menyelesaikan paling lambat dua semester
4. Menjaga kartu bimbingan skripsi supaya tidak hilang, jika dilanggar maka SPS tidak dapat dilakukan

Aturan maju SPS
1. Telah melakukan bimbingan dengan DP I& DP II masing-masing minimal empat kali
2. Proposal Skripsi di fotocopy sebanyak lima kali
3. Kartu bimbingan Skripsi di fotocopy sebanyak satu kali
4. Waktu Seminar Pra Skripsi minimal satu minggu setelah pengumpulan berkas proposal Skripsi

Aturan maju sidang skripsi
1. Telah melakukan bimbingan dengan DP I & DP II minimal 8 kali.
2. Skripsi di fotocopy dan di jilid sebanyak lima kali
3. Lembar pengesahan skripsi di fotocopy satu kali
4. Halaman judul Skripsi di fotocopy satu kali
5. Lembar Pra Transkrip nilai di fotocopy satu kali
6. Bukti Pembayaran SPP terakhir di fotocopy satu kali
7. Kartu bimbingan Skripsi di fotocopy satu kali
8. Foto copy nilai TOEP terakhir
9. Wajib melampirkan buku raport khususnya untuk mahasiswa tahun masuk 2010 keatas
10. Waktu sidang Skripsi minimal satu minggu setelah pengumpulan berkas Skripsi

Masa berlaku Surat Tugas Pembimbing
Surat Tugas Pembimbing berlaku satu tahun dan jika belum selesai akan di evaluasi oleh jurusan.

Saya menyadari akan kewajiban dan sangsi yang akan saya terima jika tidak mentaati kewajiban saya tersebut diatas.

Tanggal penerimaan Kartu/Tandatangan :.....

No.	Dosen Pembimbing I			Dosen Pembimbing II		
	Tanggal	Materi	Tanda Tangan	Tanggal	Materi	Tanda Tangan
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						

Gambar 2.2: Tampak Belakang Kartu Bimbingan Skripsi Matematika

Pada gambar (2.1) dan (2.2) diatas adalah kartu bimbingan skripsi yang digunakan di Program studi Matematika, Pendidikan Matematika, dan Ilmu Komputer Universitas Negeri Jakarta. Pada halaman depan berisikan informasi mahasiswa pe-

milik kartu bimbingan dan nama dosen yang membimbing mahasiswa. Dihalaman berikutnya berisikan aturan-aturan yang harus dipatuhi selama mahasiswa mengerjakan seminar pra skripsi dan skripsi serta tabel yang diisi oleh dosen pembimbing setiap kali mahasiswa melakukan bimbingan. Dari tabel ini dapat dilihat sejauh mana dan sesering apa mahasiswa menerima bimbingan dalam pengerjaan skripsinya.

Aturan-aturan yang tercantum dalam kartu bimbingan skripsi antara lain kewajiban mahasiswa dalam pengerjaan skripsi, aturan untuk maju SPS, dan aturan untuk maju sidang skripsi serta masa berlaku surat tugas dosen pembimbing adalah satu tahun akademik. Berikut aturan-aturan yang tercantum:

Mahasiswa berkewajiban untuk:

1. Berkonsultasi dengan DP I dan DP II masing-masing paling sedikit satu kali dalam dua minggu
2. Mentaati keputusan yang dibuat bersama dengan pembimbing
3. Berusaha menyelesaikan paling lambat dua semester
4. Menjaga kartu bimbingan skripsi supaya tidak hilang, jika dilanggar maka SPS tidak dapat dilakukan

Aturan maju SPS:

1. Telah melakukan bimbingan dengan DP I dan DP II masing-masing minimal empat kali
2. Proposal skripsi difotokopi sebanyak lima kali
3. Kartu bimbingan skripsi difotokopi sebanyak satu kali
4. Waktu Seminar Pra Skripsi minimal satu minggu setelah pengumpulan berkas proposal Skripsi

Aturan maju sidang skripsi:

1. Telah melakukan bimbingan DP I dan DP II minimal 8 kali.
2. Skripsi difotokopi dan dijilid sebanyak lima kali
3. Lembar pengesahan skripsi difotokopi satu kali
4. Halaman judul Skripsi difotokopi satu kali
5. Lembar Pra Transkrip nilai difotokopi satu kali
6. Bukti pembayaran SPP terakhir difotokopi satu kali
7. Kartu bimbingan Skripsi difotokopi satu kali
8. Fotokopi nilai TOEP terakhir
9. Wajib melampirkan buku raport khususnya untuk mahasiswa tahun 2010 keatas
10. Waktu sidang Skripsi minimal satu minggu setelah pengumpulan berkas Skripsi

B. Sistem Informasi

Sistem menurut Fitzgerarld, adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu [5]. Sedangkan menurut Ludwig Von Bartalanfy, Sistem merupakan seperangkat unsur yang saling terikat dalam suatu antar relasi diantara unsur-unsur tersebut dengan lingkungan [2]. Menurut L. Ackof, Sistem adalah setiap kesatuan secara konseptual atau fisik yang terdiri dari bagian-bagian dalam keadaan saling tergantung atau sama lainnya [1]. Sehingga dapat disimpulkan Sistem adalah suatu kegiatan atau rangkaian yang memiliki kesatuan dan saling bergantung satu sama lainnya untuk menyelesaikan suatu sasaran tertentu.

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasasi, dan menyediakan pihak luar tertentu dengan laporan - laporan yang diperlukan [6]. Menurut O'Brien, Sistem informasi ialah kombinasi dari setiap unit dikelola orang (orang), *hardware* (perangkat keras), *software* (perangkat lunak), jaringan komputer dan jaringan komunikasi data (komunikasi), dan *database* (basis data) yang mengumpulkan, mengubah, dan menyebarkan informasi tentang yang bentuk organisasi [9]. Sehingga dapat juga diartikan Sistem Informasi adalah suatu sistem yang dikelola untuk mendapatkan suatu keluaran yang dapat memberikan informasi atau kegunaan bagi pengelolanya.

Sistem informasi merupakan suatu perkumpulan data yang terorganisasi beserta tatacara penggunaanya yang mencangkup lebih jauh dari pada sekedar penyajian. Istilah tersebut menyiratkan suatu maksud yang ingin dicapai dengan jalan memilih dan mengatur data serta menyusun tatacara penggunaanya. Keberhasilan suatu sistem informasi yang diukur berdasarkan maksud pembuatannya tergantung pada tiga faktor utama, yaitu : keserasian dan mutu data, pengorganisasian data, dan tatacara penggunaannya.

Untuk memenuhi permintaan penggunaan tertentu, maka struktur dan cara kerja sistem informasi berbeda-beda bergantung pada macam keperluan atau macam permintaan yang harus dipenuhi. Suatu persamaan yang menonjol ialah suatu sistem informasi menggabungkan berbagai ragam data yang dikumpulkan dari berbagai sumber. Untuk dapat menggabungkan data yang berasal dari berbagai sumber suatu sistem alih rupa (*transformation*) data sehingga jadi tergabungkan (*compatible*). Berapa pun ukurannya dan apapun ruang lingkupnya suatu sistem informasi perlu memiliki ketergabungan (*compatibility*) data yang disimpannya [4].

C. *System Development Life Cycle* (SDLC)

Pengembangan aplikasi perangkat lunak atau sering dikenal dengan *System Development Life Cycle* (SDLC) dapat dianggap sebagai kerangka kerja formal tertua metodologi untuk membangun sistem informasi. SDLC merupakan sebuah kerangka pekerjaan yang melakukan setiap langkah-langkah dalam mengembangkan suatu sistem. Kerangka ini terdiri dari penggambaran perencanaan yang terinci bagaimana mengembangkan, memelihara, menggantikan, mengubah, atau meningkatkan sistem tertentu.

SDLC memiliki berbagai macam model dan desain yang sesuai dengan proses pengembangan suatu sistem, di mana masing-masing model memiliki keunikan dalam rangkaian langkahnya. Model-model SDLC tersebut antara lain, model *Waterfall*, model V, model *Spiral*. Selain itu ada juga model *Agile*, dan model RAD *(Rapid Application Development)*.

1. *Model Waterfall*

Model ini adalah model yang paling umum dan sering digunakan untuk mengembangkan suatu sistem. Model *waterfall* juga dikenal dengan *linear-sequential life cycle model* [10]. Sesuai dengan namanya yaitu *waterfall* atau air terjun, model ini mengembangkan sistem dengan menyelesaikan tahap awal terlebih dahulu. Jika tahap awal telah selesai, hasil pengembangan ke tahap berikutnya dan tidak ada jalan untuk kembali ke tahap sebelumnya. Tahap dalam pengembangan model *waterfall* sebagai berikut :

1. *Requirement Analysis*

   Seluruh kebutuhan sistem harus bisa didapatkan dalam fase ini, termasuk didalamnya kegunaan sistem yang diharapkan pengguna dan batasan sistem. In-

formasi ini biasanya dapat diperoleh melalui wawancara, survei, atau diskusi. Informasi yang didapatkan tersebut dianalisis untuk mendapatkan dokumentasi kebutuhan pengguna untuk digunakan pada tahap selanjutnya.

2. *System Design*

   Tahap ini dilakukan sebelum melakukan pengkodean atau *coding*. Tahap ini bertujuan untuk memberikan gambaran apa yang seharusnya dikerjakan dan bagaimana tampilannya. Tujuan dari tahap ini juga untuk membantu dalam menspesifikasikan kebutuhan *hardware* dan sistem serta mendefinisikan arsitektur sistem secara keseluruhan.

3. *Implementation*

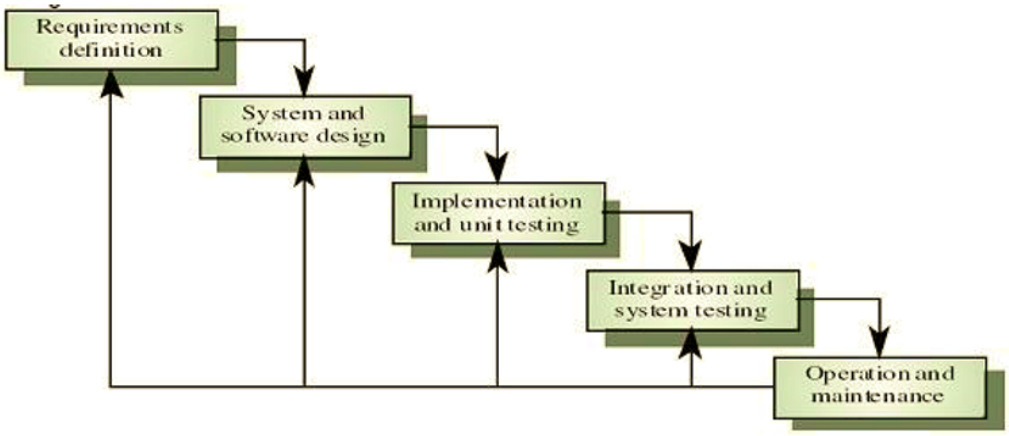
   Dalam tahap ini dilakukan pemrograman. Pembuatan sistem dipecah menjadi beberapa modul kecil yang nantinya akan digabungkan dalam tahap berikutnya. Selain itu dalam tahap ini juga dilakukan pemeriksaan terhadap modul yang dibuat, apakah sudah memenuhi fungsi yang diinginkan atau belum.

4. *Integration and Testing*

   Tahap ini dilakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian ini dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan desain sebelumnya dan masih terdapat kesalahan atau tidak.

5. *Operation and Maintenance*

   Tahap terakhir ini adalah tahap dimana sistem yang sudah jadi akan dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.



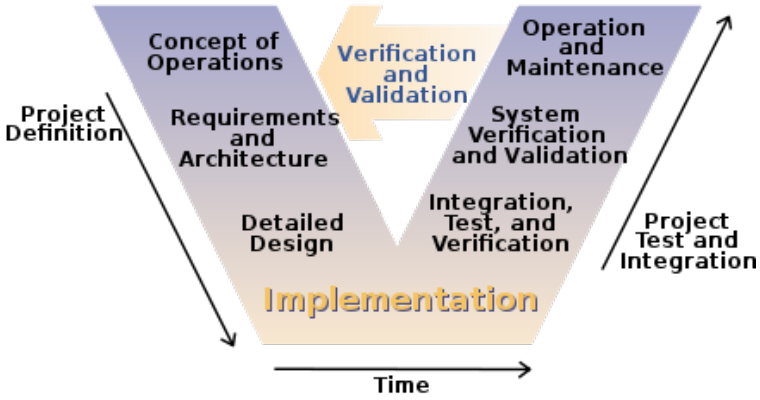
Gambar 2.3: *Model Waterfall*

Keuntungan menggunakan model ini adalah memungkinkan untuk departementalisasi atau menjadikan bagian-bagian dan mengendalikannya. Jadwal pengerjaan dapat diatur dengan batas waktu untuk setiap langkah pengembangannya. Keuntungan lainnnya, model ini mudah dipahami dan digunakan, setiap fase di proses dan diselesaikan dalam sekali waktu, setiap pekerjaan sudah terbagi dengan jelas dan mudah ditelusuri jika terjadi kesalahan.

Kelemahan dari model ini adalah tidak memungkinkan untuk banyak revisi atau perbaikan selama pengerjaan. Terutama saat fase *testing*, sangat sulit untuk kembali dan mengubah sesuatu yang tidak dipikirkan dengan baik dalam tahap konsep. Sehingga model ini tidak disarankan untuk sistem yang kompleks dan besar. Pada penelitian ini, penulis memilih model *waterfall* untuk pengembangan sistem informasi dengan alasan untuk mempercepat proses pengerjaan karena setiap tahap hanya dilakukan satu kali.

2. *Model V*

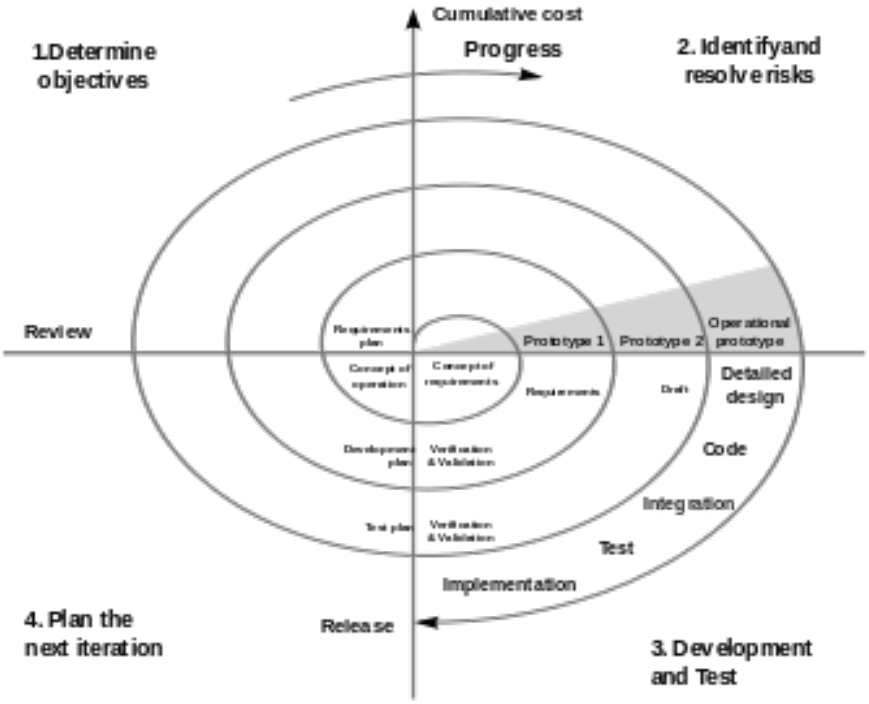
Model V merupakan variasi representasi dari model *waterfall* dimana tahapan-tahapan yang ada pada model ini sama dengan model *waterfall*. Model *waterfall* memiliki tahapan yang berjalan linier, maka model V ini dilakukan dengan cara bercabang. model ini bergerak turun dengan cara yang linear, langkah-langkah yang bengkok ke atas setelah fase *coding* sehingga membentuk V. Model V ini sangat fleksibel namun hanya dapat digunakan sekali dalam suatu proyek. Serta karena keflek-sibelan model ini ada beberapa *activity* yang digambarkan terlalu abstrak sehingga tidak diketahui dengan jelas apa yang termasuk kedalam *activity* tersebut.



Gambar 2.4: *Model V*

3. *Model Spiral*

Model *Spiral* adalah model yang merangkai sifat *iterative* dari *prototype* dengan cara kontrol dan aspek yang sistematis dari model *waterfall*. Model *Spiral* ini memiliki potensi untuk mengembangan versi pertambahan perangkat lunak dengan cepat. Dengan menggunakan model *Spiral*, pengembang dan pemakai dapat lebih mudah memahami dan bereaksi terhadap resiko setiap tingkat evolusi produk dan cocok untuk pengembangan sistem berskala besar. Namun model ini memerlukan penaksiran resiko yang masuk akal dan akan menjadi masalah yang serius jika resiko

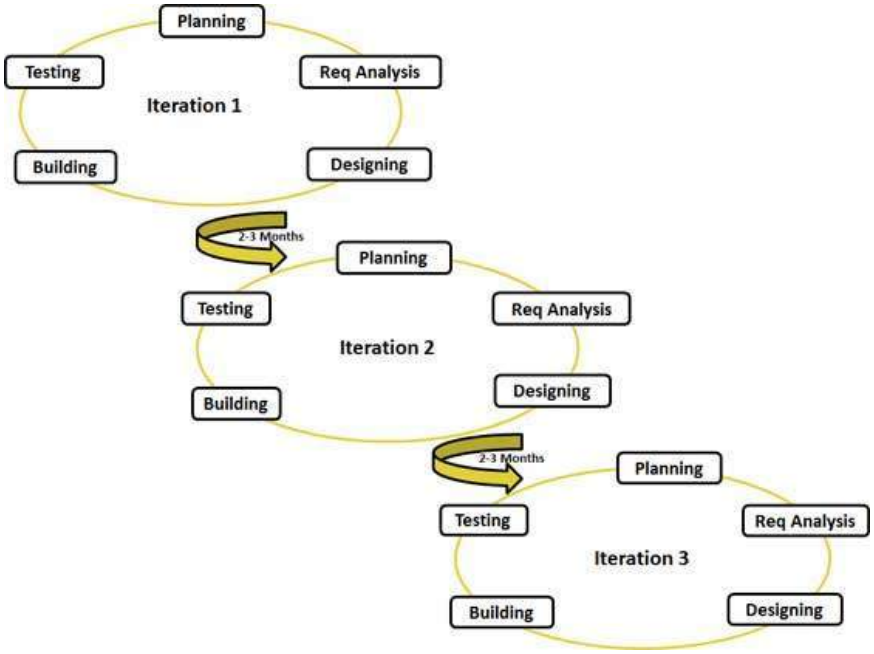


Gambar 2.5: *Model Spiral*

mayor tidak ditemukan dan diatur.

4. *Model Agile*

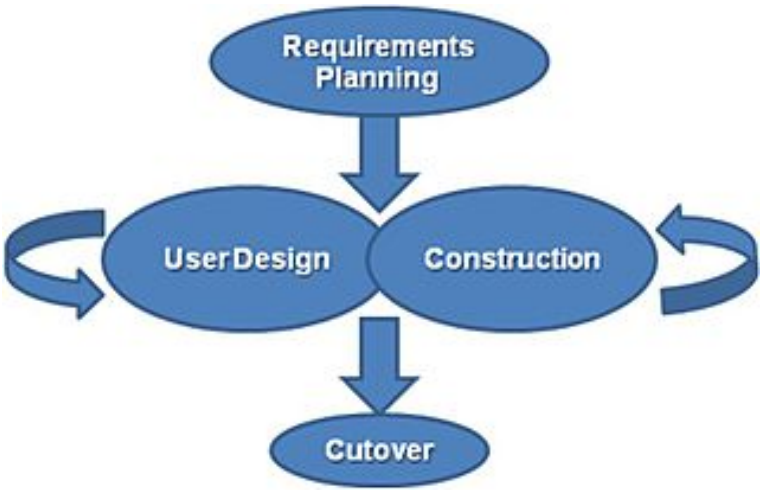
Model Agile ini adalah metodologi untuk sistem pemodelan dan mendokumentasikan perangkat lunak berdasarkan praktik terbaik. Metodologi ini lebih fleksibel dibanding dengan pemodelan tradisional, sehingga cocok dalam lingkungan yang berubah cepat. Dengan menggunakan model *agile* ini, hasil bisa didapat dalam waktu yang sangat namun membutuhkan lebih banyak inputan dari pengguna.



Gambar 2.6: *Model Agile*

5. *Model Rapid Application Development (RAD)*

Rapid Application Development (RAD) atau disebut juga *rapid prototyping* adalah model pengembangan yang tergolong dalam teknik yang bertingkat. RAD menekankan pada siklus pembangunan pendek, singkat dan cepat. Waktu yang singkat merupakan hal yang penting dalam pengembangan model ini. *Rapid application development* menggunakan metode iteratif (berulang) dalam mengembangkan sistem dimana *working model* (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan *(requirement) user* dan selanjutnya disingkirkan. RAD mengikuti tahapan pengembangan sistem sepeti umumnya, tetapi mempunyai kemampuan untuk menggunakan kembali komponen yang ada *(reusable object)*, namun proyek bisa gagal karena waktu yang disepakati tidak terpenuhi.



Gambar 2.7: *Model Rapid Application Development*

D. *Unified Modeling Language* (UML)

Unified Modelling Language (UML) merupakan suatu set standar konstruksi model dan notasi yang dikembangkan secara khusus untuk pengembangan berorientasi objek. Dengan menggunakan UML, analis dan pengguna akhir mampu untuk menggambarkan dan mengerti berbagai diagram spesifik yang digunakan dalam pengerjaan proyek pengembangan suatu sistem. UML dapat juga diartikan sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO *(Object-Oriented)* [12]. Penggunaan UML bertujuan untuk mengidentifikasi bagian-bagian yang termasuk dalam lingkup sistem yang dibahas dan bagaimana hubungan antara sistem dengan subsistem maupun sistem lain diluarnya. Dengan menggunakan UML, pengembang dapat melakukan beberapa hal seperti :

1. Tinjauan umum bagaimana arsitektur sistem secara keseluruhan.
2. Penelaahan bagaimana objek-objek dalam sistem saling mengirimkan pesan dan saling bekerjasama satu sama lain.

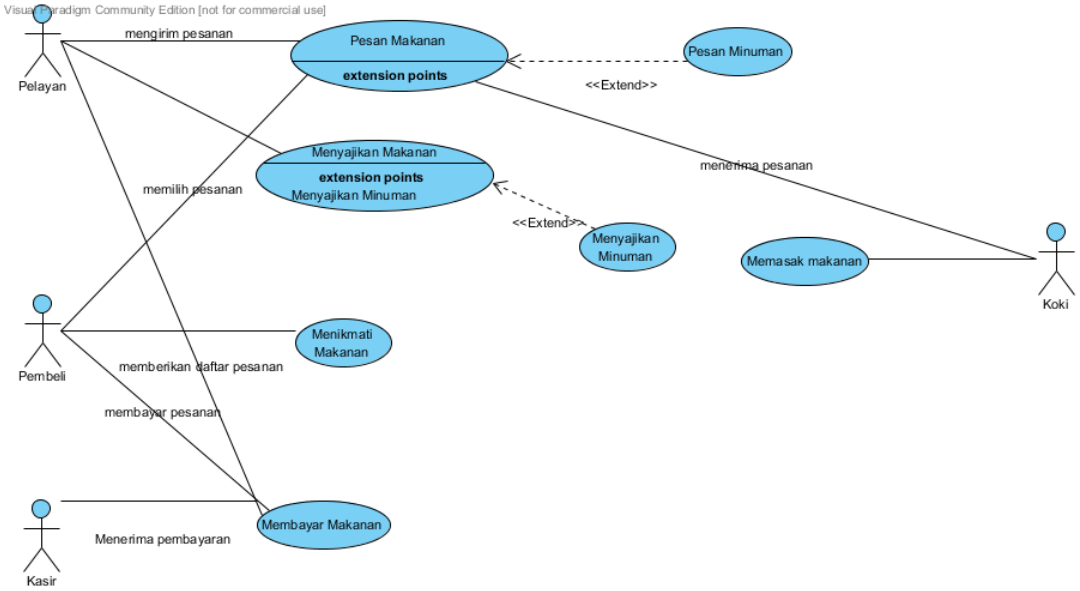
3. Menguji apakah sistem perangkat lunak sudah berfungsi seperti seharusnya.

4. Dokumentasi sistem perangkat lunak untuk keperluan-keperluan tertentu dimasa yang akan datang.

UML memiliki berbagai macam diagram untuk memodelkan sebuah sistem yaitu *use case, class, object, state, sequence, collaboration, activity, component*, dan *deployment diagram*. Namun dalam pengembangan sistem kali ini menggunakan diagram *use-case, class*, dan *activity*.

1. Diagram *Use Case*

Diagram *use case* menggambarkan apa saja aktivitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar seperti apa yang dilakukan bukan tentang bagaimana melakukannya. Diagram ini berkaitan dengan skenario atau contoh interaksi pengguna dengan sistem. Berikut adalah contoh diagram *use case* sistem dalam sebuah restoran.



Gambar 2.8: Diagram *Use Case* Restoran

Kegunaan diagram use case antara lain :

1. Menjelaskan fasilitas yang ada, *use case* baru menghasilkan fasilitas baru ketika sistem di analisa, dan desain menjadi lebih jelas.

2. Komunikasi dengan klien, penggunaan notasi dan simbol dalam diagram *use case* membuat pengembang lebih mudah berkomunikasi dengan klien-kliennya.

3. Membuat tes dari kasus-kasus secara umum, kumpulan kejadian untuk *use case* bisa dilakukan tes kasus layak untuk kejadian-kejadian tersebut.

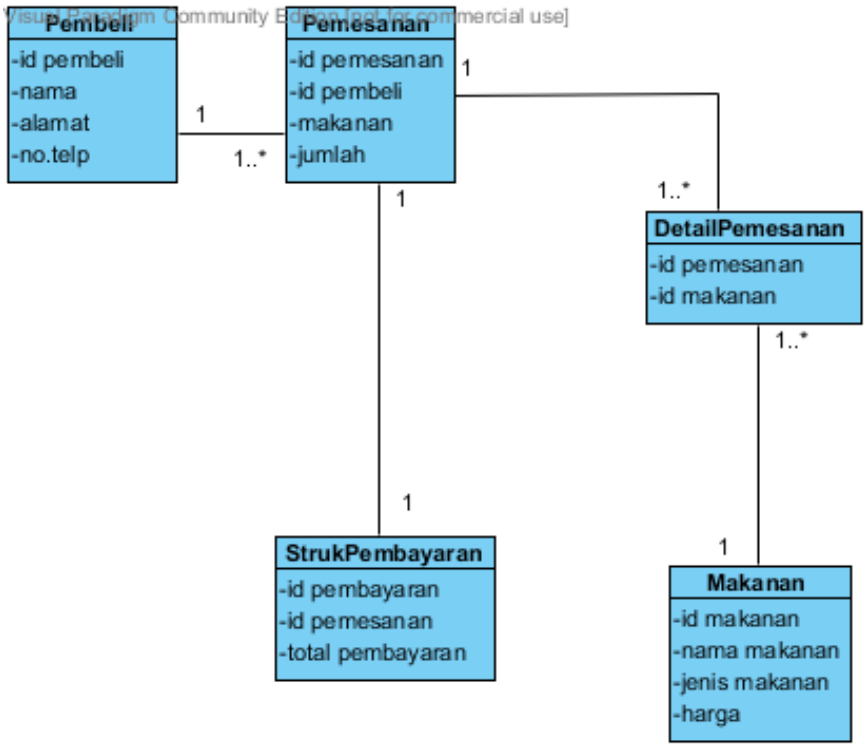
2. Diagram *Class*

Diagram *Class* menggambarkan sekumpulan kelas-kelas berbasiskan objek yang dibutuhkan dalam pemrograman, hubungan antara kelas yang satu dengan kelas yang lain, nama atribut, *properties*. Tujuan utama dari diagram *class* adalah untuk mendokumentasikan dan menggambarkan kelas-kelas yang akan dibuat untuk sistem yang baru [12]. *Class* diagram bersifat statis yaitu menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan. Diagram *class* memiliki 3 macam hubungan (*relationships*) sebagai berikut:

1. *Association*, suatu hubungan antara bagian dari dua kelas.

2. *Aggregation*, salah satu kelas merupakan bagian dari suatu kumpulan.

3. *Generalization*, suatu hubungan turunan dengan mengasumsikan satu kelas merupakan suatu *superclass* dari kelas yang lain.



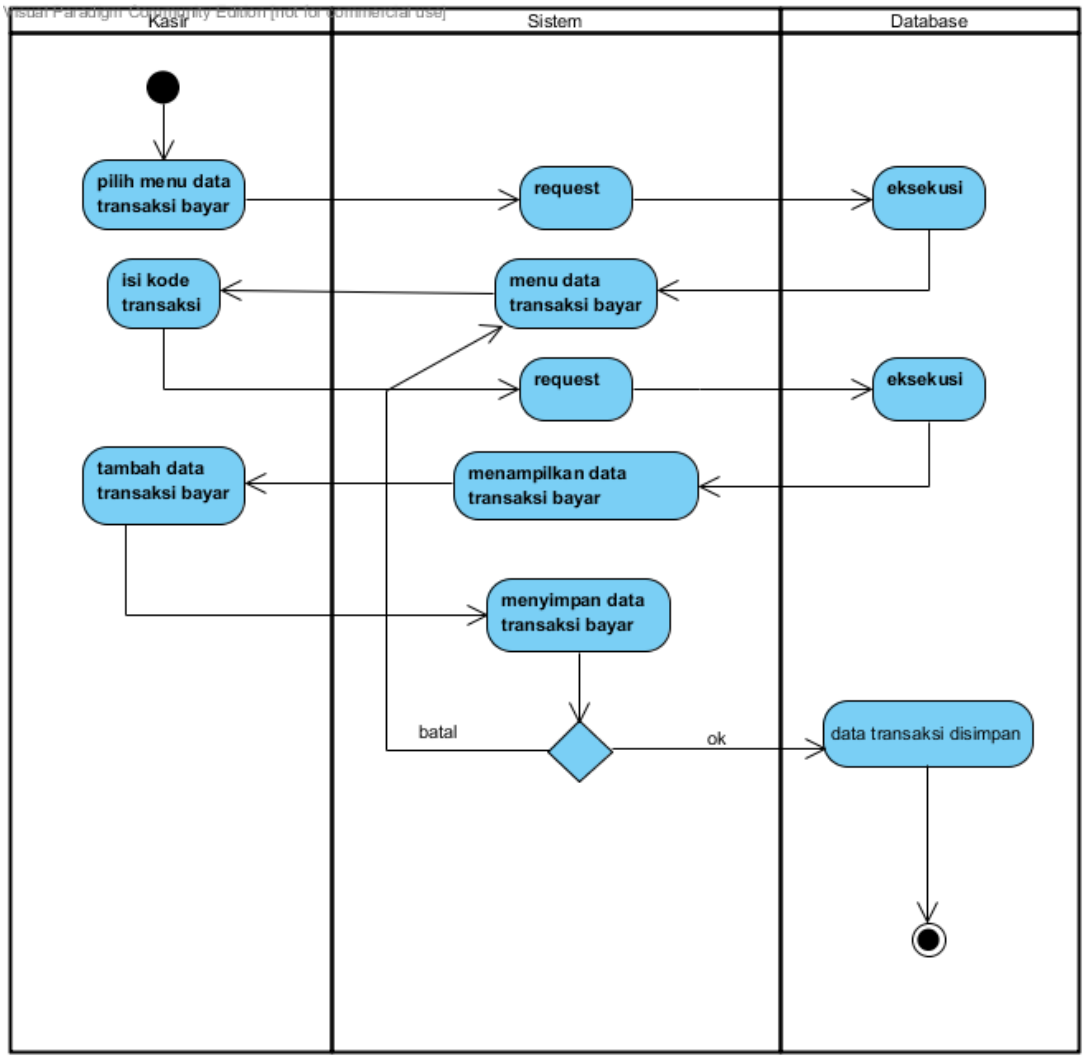
Gambar 2.9: Diagram *Class* Restoran

3. Diagram *Activity*

Diagram *Activity* adalah diagram alur yang menunjukkan aktivitas dari berbagai pengguna atau sistem, orang yang melakukan masing-masing aktivitas, dan juga urutan dari aktivitas-aktivitas yang ada. Diagram ini sering digunakan oleh *flowchart*. Pada diagram *activity* berfokus pada aktivitas-aktivitas yang terjadi yang terkait dalam suatu proses tunggal. Dengan kata lain, diagram ini menunjukkan bagaimana aktivitas-aktivitas tersebut bergantung satu sama lain.

Diagram *activity* dapat dibagi menjadi beberapa jalur kelompok yang menunjukkan objek mana yang bertanggungjawab untuk suatu aktivitas, yaitu peralihan tunggal (*single transition*) yang timbul dari setiap adanya *activity*, yang saling menghubungi pada aktivitas berikutnya. Transisi eksklusif (*exclusive transition*) yang dapat membuat cabang ke dua atau lebih percabangan. Label *Guard Expression* (ada di

dalam [ ]) yang menerangkan *output* (keluaran) dari percabangan.



Gambar 2.10: Diagram *Activity* Restoran

E. *Entity Relationship Diagram* (ERD)

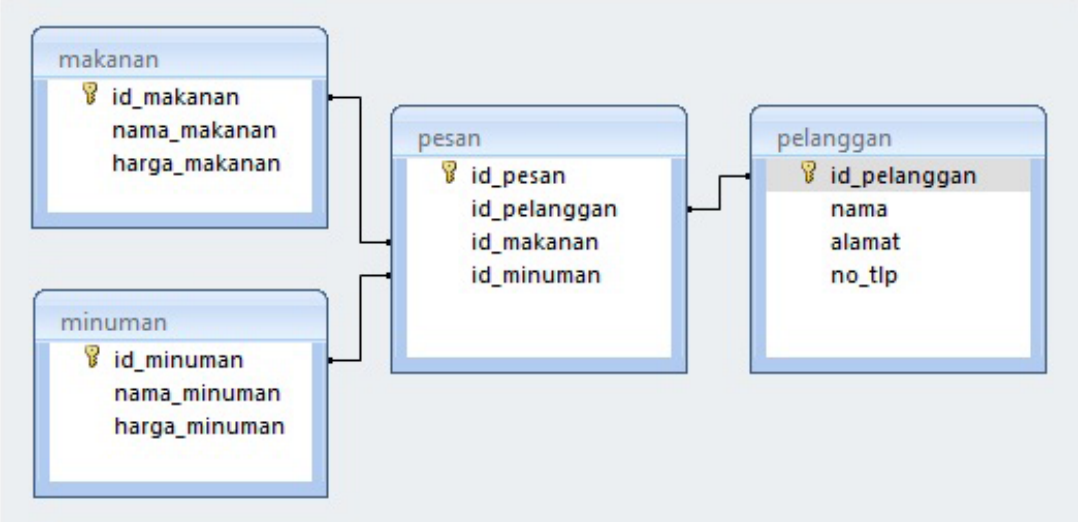
Dalam mengembangkan suatu sistem informasi, penggunaan *Entity Relationship Diagram* (ERD) dapat mempermudah perancangan data-data atau objek-objek dalam sistem. *Entity Relationship Diagram* (ERD) menurut James A. Obrien merupakan perencanaan data dan diagram pengembangan sistem yang memodelkan hubungan antar entitas dalam suatu proses bisnis [9]. ERD dapat dikatakan sebagai alat

untuk mendefinisikan data-data dan objek-objek dalam suatu basis data yang mempunyai hubungan antar relasi. Penggunaan ERD dapat menghindari kemungkinan kekacauan data karena pemodelan data yang tidak efisien sebab ERD menganalisis hubungan dari data yang hendak disimpan dalam basis data. ERD memiliki tiga komponen penyusun yang digunakan, yaitu atribut, entitas, dan hubungan/relasi.

Notasi	Keterangan
Entitas	Entitas adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
Relasi	Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
Atribut	Atribut berfungsi mendeskripsikan karakter entitas (atribut yang berfungsi sebagai *key* diberi garis bawah).
Garis	Garis sebagai penghubung antara relasi dan entitas atau relasi dan entitas dengan atribut.

Gambar 2.11: Komponen Penyusun ERD

Relasi *relationship* memiliki tiga jenis relasi berdasarkan kardinalitas yang menyatakan jumlah himpunan relasi antar entitas. Jenis relasi tersebut yaitu *one-to-one*, *one-to-many*, dan *many-to-many*. Dalam relasi *one-to-one*, setiap entitas pada himpunan entitas A berhubungan dengan satu entitas pada himpunan entitas B, contohnya adalah setiap pegawai hanya bekerja pada satu departemen. Relasi *one-to-many* menggambarkan Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B dapat berhubungan dengan satu entitas pada himpunan entitas A, contohnya satu departemen memiliki banyak pegawai. Sedangkan relasi *many-to-many* memungkinkan Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, contohnya adalah mahasiswa dengan mata kuliah.



Gambar 2.12: Contoh ERD Restoran

Diagram ERD restoran diatas menggambarkan pada suatu restoran untuk memproses pesanan setiap pelanggannya dengan mengetahui makanan dan minuman yang dipesan pelanggan tersebut. Dengan demikian, ERD yang tepat untuk memproses pesanan pelanggan seperti ditunjukkan gambar diatas, dimana pesan mengambil data makanan dan minuman untuk mengetahui menu apa yang dipesan dan data pelanggan untuk mengetahui pelanggan mana yang memesan makanan dan minuman tersebut.

F. *Database MySQL*

Secara konsep, *database* atau basis data adalah kumpulan dari data-data yang membentuk suatu berkas (*file*) yang saling berhubungan (*relation*) dengan tatacara yang tertentu untuk membentuk data baru atau informasi. Basis data (*database*) dapat diartikan sebagai kumpulan dari data yang saling berhubungan (*relation*) antara satu dengan yang lainnya yang diorganisasikan berdasarkan skema atau struktur tertentu.

Menurut Gordon C. Everest, *database* adalah koleksi atau kumpulan data yang mekanis terbagi, terdefinisi secara formal dan dikontrol terpusat pada organisasi. Sedangkan menurut Toni Fabbri, *Database* adalah sebuah sistem *file-file* yang terinte-

grasi yang mempunyai minimal *primary key* untuk pengulangan data. Berbeda dengan Connolly yang mengatakan bahwa *database* adalah suatu kumpulan data yang saling berhubungan secara logis dan penjelasan tentang data yang terhubung tersebut dirancang sedemikian rupa sehingga dapat memberikan informasi yang diperlukan oleh organisasi [3]. Dengan demikian, dapat disimpulkan bahwa *database* adalah kumpulan data yang saling berhubungan dengan sebuah *primary key* dimana dapat memberikan sebuah informasi yang diperlukan.

Penggunaan *database* dalam sistem informasi sangat bermanfaat, diantaranya:

1. mengurangi kerangkapan data
2. mencapai independensi data
3. mengintegrasi data dari beberapa *file*
4. meningkatkan efisiensi dan efektivitas penggunaan data
5. meningkatkan keamanan data
6. memudahkan pengguna dalam menggali informasi dari kumpulan data
7. meningkatkan pemakaian data secara bersama
8. mengurangi konflik antar pengguna data

G. *Framework Codeigniter*

Menurut situs resmi *Codeigniter*, *Codeigniter* merupakan salah satu *framework PHP* dengan menggunakan model MVC (*Model, View, Controller*) yang digunakan untuk membuat *website* dinamis. *Codeigniter* dirilis pertama kali pada 28

Februari 2006 dengan perancang EllisLab dan pengembang British Columbia Institute of Technology. Versi stabil terakhir dari *codeigniter* versi 3.0.4 yang dirilis pada 13 Januari 2016 [14].

Codeigniter adalah suatu *framework PHP*. *Framework* secara sederhana dapat diartikan kumpulan dari fungsi-fungsi/prosedur-prosedur dan *class-class* untuk tujuan yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang *programmer*, tanpa harus membuat fungsi atau *class* dari awal. Berikut merupakan beberapa alasan mengapa menggunakan *Framework*:

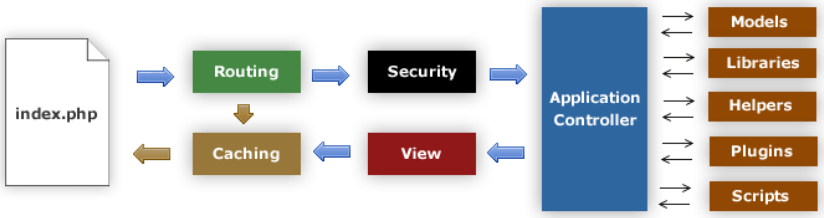
1. Mempercepat dan mempermudah pembangunan sebuah sistem web.
2. Relatif memudahkan dalam proses *maintenance* karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat *programmer* mengikuti pola standar yang ada).
3. Framework menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, *pagination* , *multiple database*, *scaffolding*, pengaturan session, *error handling*, dll.
4. Lebih bebas dalam pengembangan jika dibandingkan CMS.

MVC yang digunakan *codeigniter* merupakan suatu konsep yang cukup popular dalam pembangunan sistem web dimana pengembangan sistem berdasarkan komponen utama yang membangun sebuah sistem seperti manipulasi data, *user interface*, dan bagian yang menjadi *control* sistem. komponen yang membangun suatu MVC *pattern* dalam suatu sistem yaitu:

1. *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu sistem web bagian ini biasanya berupa *file template HTML*, yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian *model*.

2. *Model*, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (input, ubah, hapus, cari), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.

3. *Controller*, merupakan bagian yang mengatur hubungan antara bagian *model* dan bagian *view*, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh sistem.



Gambar 2.13: *Flowchart Codeigniter*

Ada beberapa kelebihan *Codeigniter* (CI) dibandingkan dengan *Framework PHP* lain,

1. Performa sangat cepat : salah satu alasan tidak menggunakan *framework* adalah karena eksekusinya yang lebih lambat daripada *PHP from the scratch*, tapi *Codeigniter* sangat cepat bahkan mungkin bisa dibilang *codeigniter* merupakan *framework* yang paling cepat dibanding *framework* yang lain.

2. Konfigurasi yang sangat minim (*nearly zero configuration*) : tentu saja untuk menyesuaikan dengan *database* dan keleluasaan *routing* tetap diizinkan melakukan konfigurasi dengan mengubah beberapa *file* konfigurasi seperti *database.php* atau *autoload.php*, namun untuk menggunakan *codeigniter* dengan *setting standard*, anda hanya perlu mengubah sedikit saja *file* pada folder *config*.

3. Banyak komunitas: dengan banyaknya komunitas CI ini, memudahkan kita untuk berinteraksi dengan yang lain, baik itu bertanya atau teknologi terbaru.

4. Dokumentasi yang sangat lengkap : Setiap paket instalasi *codeigniter* sudah disertai *user guide* yang sangat bagus dan lengkap untuk dijadikan permulaan, bahasanya pun mudah dipahami.

Keuntungan menggunakan *framework* adalah *programmer* tidak perlu membuat program dari awal, karena *framework* sudah memberikan *library* fungsi - fungsi yang sudah diorganisasi untuk membuat suatu program dengan cepat. Sehingga *programmer* hanya perlu memasukkan data yang akan diproses dan bagaimana menampilkannya.

H. Web Server *XAMPP*

Web server merupakan perangkat lunak yang mengelola permintaan user dari *browser* dimana hasilnya akan dikembalikan ke *browser*. Contoh web server adalah IIS (*Internet Information Services*) produk *Microsoft corp*. Web server adalah bentuk dari server yang khusus digunakan untuk menyimpan halaman website atau home page. Sebuah komputer dapat dikatakan sebagai web server apabila komputer tersebut memiliki suatu program server yang disebut PWS atau *Personal Web Service*. Untuk kemudian PWS ini akan difungsikan agar halaman web yang ada di dalam sebuah komputer server dapat dipanggil oleh komputer klien.

XAMPP adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis PHP dan menggunakan pengolah data MySQL dikomputer lokal. *XAMPP* juga disebut sebuah *CPanel Server Virtual* yang dapat membantu pekerjaan dengan melakukan *preview* sehingga dapat memodifikasi *website* tanpa harus terakses dengan internet. Menurut situs resmi *Xampp*, *Xampp* merupakan sebuah web server yang ber-

tindak sebagai server yang berdiri sendiri atau *local server (localhost)*, yang terdiri dari beberapa program bawaan antara lain: *Apache HTTP Server, MySQL Database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. *Xampp* merupakan singkatan dari X (empat sistem operasi apapun), *Apache, MySQL, PHP*, dan *Perl*. Program ini tersedia dalam *GNU General Public License* dan bebas, merupakan web server yang mudah untuk digunakan yang dapat menampilkan halaman web yang dinamis [15].