BAB II
LANDASAN TEORI

Landasan teori memaparkan teori serta konsep dasar yang berasal dari buku-buku rujukan dan jurnal yang berkaitan dengan tema penelitian sebagai panduan dan pemecahan masalah penelitian. Pada bab ini akan dijelaskan teori yang berhubungan dengan penelitian ini.

2.1. *e-commerce*

e-commerce merupakan yang populer dalam bidang perdagangan saat ini yang merupakan singkatan *Electronic Commerce*. *e-commerce* merupakan suatu cara berniaga yang dilakukan secara *online* dengan memanfaatkan fasilitas internet. Menurut Irmawati (2011), *e-commerce* merupakan transaksi perniagaan yang dilakukan melalui jaringan komputer, seperti pembelian, penjualan, pertukaran produk, pelayanan dan pemberian informasi. Ada lima jenis dari *e-commerce* (Nanehkaran, 2013):

a. B2B, merupakan singkatan dari *Business to Business*. Ini merupakan jenis transaksi perdagangan antara perusahaan dengan perusahaan lainnya untuk mentransfer layanan dan produk.
b. B2C, merupakan singkatan dari *Business to Consumer*. Jenis ini mengacu pada transaksi antara bisnis dan konsumen sehingga menciptakan suatu etalase elektronik yang menawarkan informasi dan barang.
c. C2B, merupakan singkatan dari *Consumer to Business* yang merupakan transfer layanan, barang atau informasi dari orang ke bisnis atau merupakan model pengguna akhir menciptakan produk dan layanan yang digunakan oleh bisnis dan institusi.
d. C2C, merupakan singkatan dari *Consumer to Consumer* yang melibatkan transaksi antar pengguna dan itu merupakan model bisnis dua konsumen saling berbisnis secara langsung.

e. *m-commerce,* merupakan singkatan dari *Mobile Commerce*. Istilah ini diciptakan pada tahun 1997 untuk menargetkan pembelian dan penjualan produk, informasi dan layanan melalui perangkat genggam nirkabel seperti *handphone*, laptop, dan PDA yang berinteraksi dengan menggunakan jaringan komputer yang berkemampuan untuk melakukan pembelian barang secara *online*.

2.2. HTML5 Aplikasi

Dalam buku yang berjudul *Training Guide: Programming in HTML5 with JavaScript and CSS3* (Johnson, 2013), HTML merupakan akronim dari *Hypertext Markup Language*. HTML berasal dari bahasa *markup* yang sebelumnya digunakan dalam penerbitan dokumen yang disebut dengan SGML (*Standard Generalized Markup Language*). *World Wide Web Consortium* yang dikenal dengan sebutan W3C (http://www.w3c.org), bertanggung jawab untuk mengembangkan standar terbuka untuk *web*. W3C memperkenalkan XHTML untuk memecahkan persalahan dalam HTML, hingga versi 4. XHTML merupakan spesifikasi berbasis XML yang memperketat spesifikasi HTML agar mematuhi aturan XML yang menggambarkan dokumen yang terbentuk dengan baik. Walau XHTML memecahkan beberapa permasalahan yang terdapat pada HTML, ada masalah lain yang membutuhkan solusi, yaitu terjadi peningkatan kebutuhan jumlah media pada *web*. *Cascading Style Sheets* (CSS) memberikan dukungan untuk menambahkan gaya seperti warna dan font secara konsistem di seluruh situs *web*. Namun karena meningkatnya jumlah kebutuhan media pada *web* yang menginginkan lebih banyak media, *browser* menambahkan dukungan yang dapat diprogram dengan menyediakan JavaScript, tetapi versi awal JavaScript lambat dan sulit untuk diprogram.

HTML5 tidak berasal dari XHTML, melainkan berasal dari HTML 4.01. Dengan menerapkan aturan XHTML ke HTML5 membuat halaman *web* lebih sesuai dengan berbagai *browser* yang lebih luas. HTML5 mewakili HTML, CSS, dan JavaScript yang dibuat kembali dengan cara yang memecahkan kebutuhan untuk situs *web* yang kaya dan interaktif yang dapat memutar *audio* dan *video* dan

mendukung animasi dari dalam *browser* tanpa perlu menggunakan *plug-in* yang berisi sebagian besar *tag* dari HTML 4.01. Namun beberapa *tag* telah ditafsirkan ulang menjadi *tag* semantik.

2.3. *Agile Software Development*

Metodologi pengembangan aplikasi yang digunakan pada penelitian ini adalah metodologi SDLC (*Software Development Life Cycle*) menggunakan pendekatan *AGILE Life Cycle Model*. AGILE *Software Development* merupakan salah satu metodologi dalam melakukan pengembangan pada suatu perangkat lunak. Kata AGILE digunakan untuk menggambarkan konsep model proses yang berbeda dari konsep model proses yang telah ada sebelumnya. Secara bahasa kata AGILE berarti tangkas, cepat, atau ringan. AGILE *Software Development* dicetuskan oleh Kent Beck beserta 16 orang rekannya dengan menyatakan bahwa konsep ini merupakan cara membangun perangkat lunak dengan melakukannya dan membantu orang lain membangunnya sekaligus.

AGILE merupakan gabungan metode *incremental* dan *iterative*. Bila dibandingkan dengan metode pengembangan aplikasi *waterfall*, AGILE cenderung tidak memiliki fase-fase. Hal ini merupakan sebab dari AGILE berproses secara *iterative* dengan bentuk perputaran yang pendek. Kebutuhan yang ada kemudian direncanakan, diimplementasikan, diuji dan dievaluasi sehingga implikasi dari AGILE SDLC ini memberikan kemampuan adaptasi yang tinggi terhadap perubahan selama proses pembangunan *software*.

Sebagaimana tertera pada AGILEmanifesto.org, AGILE *Development* merupakan konsep pengembangan aplikasi yang memegang nilai-nilai berikut:

1. Individu dan interaksi lebih dari proses dan sarana perangkat lunak.
2. Perangkat lunak yang bekerja lebih dari dokumentasi yang menyeluruh.
3. Kolaborasi dengan klien lebih dari negosiasi kontrak.
4. Tanggap terhadap perubahan lebih dari mengikuti rencana.

Salah satu ciri dari AGILE adalah tanggap terhadap perubahan. Hal ini adalah perubahan yang merupakan hal utama dalam membangun *software*, seperti perubahan kebutuhan *software*, perubahan anggota tim, dan perubahan teknologi.

Selain itu, komunikasi merupakan sesuatu yang penting dalam AGILE *Software Development* ini, seperti komunikasi antara anggota tim, antara orang-orang teknis dan *businessman*, dan juga antara *developer* dengan manajernya. Hal ini didukung dengan 12 prinsip untuk mencapai proses yang termasuk dalam *agility* sebagaimana yang telah didefinisikan oleh AGILE *Alliance*, diantaranya:

1. Prioritas tertinggi adalah memuaskan pelanggan melalui penyerahan awal dan berkelanjutan perangkat lunak yang bernilai.
2. Menerima perubahan *requirements* meski perubahan tersebut diminta pada akhir pengembangan.
3. Memberikan perangkat lunak yang sedang dikerjakan dengan sering, beberapa minggu atau beberapa bulan, dengan pilihan waktu yang paling singkat.
4. Pihak bisnis dan pengembang harus bekerja sama setiap hari selama pengembangan berjalan.
5. Bangun proyek dengan individu-individu yang bermotivasi tinggi dengan memberikan lingkungan dan dukungan yang diperlukan, dan mempercayai mereka sepenuhnya untuk menyelesaikan pekerjaannya.
6. Metode yang paling efektif dan efisien dalam menyampaikan informasi kepada tim pengembangan adalah dengan komunikasi *face-to-face*.
7. Perangkat lunak yang dikerjakan merupakan pengukur utama kemajuan.
8. Proses AGILE memberikan proses pengembangan yang bisa ditopang. Sponsor, pengembang, dan *user* harus bisa menjaga kekonstanan langkah yang tidak pasti.
9. Perhatian yang terus menerus terhadap rancangan dan teknik yang baik meningkatkan *agility*.
10. Kesederhanaan -seni untuk meminimalkan jumlah pekerjaan- adalah penting.
11. Arsitektur, *requirements*, dan rancangan terbaik muncul dari tim yang mengatur sendiri.
12. Pada interval reguler tertentu, tim merefleksikan bagaimana menjadi lebih efektif, kemudian menyesuaikannya.

Beberapa jenis metode pengembangan aplikasi AGILE:

1. *Adaptive Software Development* (ASD)
2. *Feature Driven Development* (FDD)
3. *Dynamic Software Development Method* (DSDM)
4. *Rapid Application Development* (RAD)
5. Scrum
6. *Extreme Programming* (XP)
7. *Rational Unify Process* (RUP)

Ada banyak macam dari metodologi *AGILE Software Development*. Namun pendekatan metodologi AGILE yang digunakan pada penelitian ini adalah *Extreme Programming*. *Extreme Programming* (XP) merupakan suatu pendekatan pengembangan perangkat lunak yang menyederhanakan berbagai tahapan dalam suatu proses pengembangan perangkat lunak sehingga pengembangan tersebut menjadi lebih adaptif dan juga fleksibel. Metode XP tidak hanya berfokus pada *coding*, namun juga meliputi seluruh area pengembangan perangkat lunak.

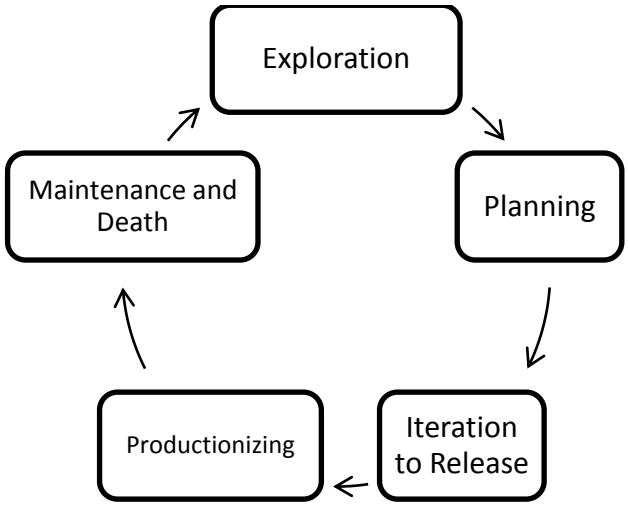
Metode XP diusulkan pertama kali oleh Kent Beck dan Ward Cunningham pada tahun 1996. XP digunakan awalnya disebabkan ada permintaan dari *customer* yang sering berubah dengan cepat sehingga mengakibatkan putaran kehidupan metode pengembangan perangkat lunak tradisional dan menjadi lebih pendek dan tidak selaras dengan metode tradisional karena pada umumnya memerlukan desain yang luas.

Ada 4 tahapan dalam metode *extreme programming*, antara lain:

1. *Planning*, merupakan tahapan awal dalam pembangunan sebuah aplikasi. Dalam tahap ini dilakukan beberapa kegiatan perencanaan, seperti identifikasi permasalahan, menganalisa kebutuhan hingga penetapan jadwal pelaksanaan pembangunan aplikasi.
2. *Design*, merupakan tahapan perancangan pemodelan aplikasi yang dimulai dari pemodelan sistem, pemodelan arsitektur sampai dengan pemodelan basis data. Pemodelan aplikasi dan arsitektur menggunakan diagram *Unified Modelling Language* (UML).

3. *Coding*, merupakan tahapan kegiatan penerapan pemodelan yang sudah dibuat ke dalam bentuk *user interface* dengan menggunakan bahasa pemrograman.
4. *Testing*, merupakan tahapan yang dilakukan setalah tahapan pengkodean selesai dilakukan. Tahapan ini dilakukan untuk mengetahui kesalahan apa saja yang timbul saat aplikasi sedang berjalan serta mengetahui apakah sistem yang dibangun sudah sesuai dengan kebutuhan pengguna. Unit *test* yang telah dibuat harus diimplementasikan menggunakan suatu *framework* dan diatur ke dalam *universal testing suite*, integrasi dan validasi sistem dapat dilakukan setiap hari. *Customer test* dilakukan oleh *customer* dan fokus pada keseluruhan fitur dan fungsional sistem. *Acceptance test* diperoleh dari *customer stories* yang telah diimplementasikan sebagai bagian dari *software release*.

Siklus hidup metode XP terdiri dari lima fase, yaitu: *Exploration, Planning, Iterations to Release, Productionizing, Maintenance and Death* (Pekka Abrahamsson, Outi Salo & Warsta, 2002).



Gambar 2.1 Siklus Metode *Extreme programming* (Sumber: *Agile Software Development Methods*, 2002)

Fase eksplorasi merupakan fase pengumpulan informasi tentang permasalahan dan pemecahan masalah dengan beriteraksi dengan *user*. *User* menceritakan yang mereka ingin masukkan dalam aplikasi yang akan dibangun.

User menjelaskan fitur yang akan ditambahkan ke dalam program. Pada saat yang sama penulis mempersipkan kebutuhan yang akan digunakan dalam penelitian.

Fase perencanaan, merupakan fase setelah fase eksplorasi telah dianggap cukup, kemudian melakukan perencanaan berupa perancangan sistem yang akan dibangun sesuai dengan kebutuhan dari *user*. Pada fase ini termasuk di dalam nya perancangan sistem, desain, *coding* dan *testing*, namun sistem yang dibangun tidak langsung dirilis.

Iterasi untuk fase rilis termasuk beberapa iterasi dari sistem sebelum rilis pertama. Jadwal yang ditetapkan dalam tahap perencanaan dipecah menjadi sejumlah iterasi. Iterasi pertama menciptakan sistem dengan arsitektur keseluruhan sistem. Tes fungsional yang dibuat oleh *user* dijalankan pada akhir setiap iterasi. Pada akhir iterasi terakhir, sistem siap untuk diproduksi.

Tahap produksi membutuhkan pengujian tambahan dan pengecekan kinerja sistem sebelum sistem dapat dilepas ke konsumen. Pada fase ini, perubahan baru masih dapat ditemukan dan keputusan harus dibuat jika mereka termasuk dalam rilis rilis saat ini. Selama fase ini, iterasi mungkin perlu dipercepat, ide dan saran yang tertunda didokumentasikan untuk penerapan selanjutnya selama, misalnya, fase pemeliharaan. Setelah rilis pertama diproduksi untuk *User*, proyek XP harus menjaga sistem dalam produksi berjalan sambil menghasilkan iterasi baru.

Fase akhir sudah dekat ketika *user* tidak lagi memiliki keinginan untuk diimplementasikan. Ini mensyaratkan bahwa sistem memenuhi kebutuhan *user* juga dalam hal lain (misalnya, mengenai kinerja dan keandalan). Ini adalah waktu dalam proses XP ketika dokumentasi yang diperlukan dari sistem ini akhirnya ditulis karena tidak ada lagi perubahan pada arsitektur, desain atau kode yang dibuat. Fase akhir juga bisa terjadi jika sistem tidak memberikan hasil yang diinginkan, atau jika itu menjadi terlalu mahal untuk pengembangan lebih lanjut.

2.4. UML (*Unified Modelling Language*)

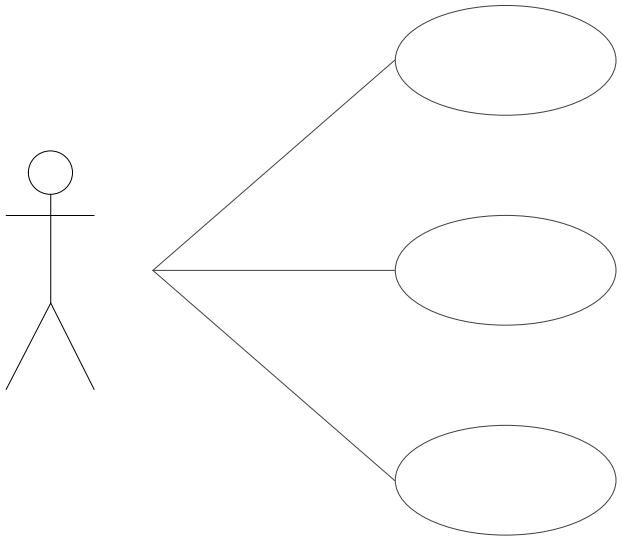
UML adalah singkatan dari *Unified Modelling Language* yang merupakan suatu alat perancangan sistem yang berorientasi pada objek. Kemunculan UML

secara filosofi diilhami oleh konsep sebelumnya yang telah ada, yaitu *Object Oriented* (OO). Ini merupakan konsep yang menggambarkan sebuah sistem layaknya kehidupan yang nyata. Diagram UML bertujuan untuk membantu komunikasi tim pengembangan proyek, menjelajah potensi desain, dan memvalidasi desain arsitektur perangkat lunak.

Ada empat macam diagram untuk memodelkan *software* berorientasi objek yang disediakan oleh UML, yaitu:

4.1. *Use Case Diagram*

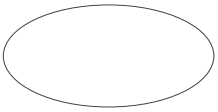
Use Case Diagram merupakan diagram yang menggambarkan hubungan yang terjadi antara aktor dan *use case*. *Use case diagram* digunakan dalam melakukan analisis dan desain pada sebuah sistem.(Ambler, 2005)



Gambar 2.2 Contoh *Use Case Diagram* (Sumber: *The Elements of* UML, 2005)

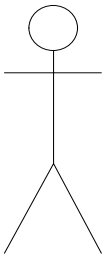
Dari gambar di atas dapat dilihat bahwasanya terdapat beberapa komponen dalam sebuah *use case diagram*, seperti:

a. *Use case*, menggambarkan tentang aksi yang dilakukan oleh aktor. Dengan kata lain *use case* merupakan gambaran fungsional sistem yang akan dibuat agar pengguna lebih mengerti penggunaan sistem. Pada penerapannya, *use case* digambarkan dalam bentuk elips yang horizontal.



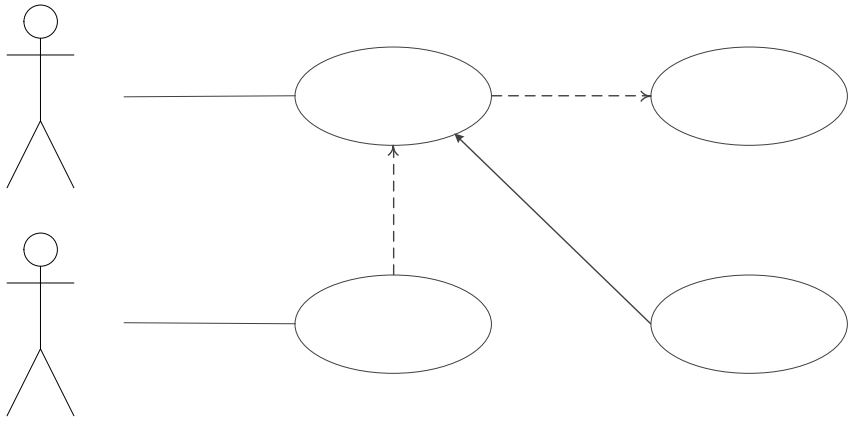
Gambar 2.3 *Use Case* (Sumber: *The Elements of* UML, 2005)

b. *Actor*, menggambarkan seseorang yang berinteraksi dengan sistem. *Actor* hanya bisa menginputkan informasi dan menerima informasi dari sistem dan tidak memegang kendali pada *use case*. Dalam penerapannya, *actor* digambarkan dengan *stickman*.



Gambar 2.4 *Actor* (Sumber: *The Elements of* UML, 2005)

c. *Relationship,* merupakan hubungan yang terjadi antara *use case* dengan *actor*. Dalam *use case diagram, relationship* meliputi:
   i. Asosiasi antara *actor* dan *use case*, yaitu hubungan yang terjadi karena ada interaksi antara *actor* dan *use case*. Asosiasi tipe ini menggunakan satu garis lurus dari *actor* kepada *use case*.
   ii. Asosiasi antara dua *use case*, yaitu hubungan yang terjadi karena ada interaksi antara *use case* yang satu dengan *use case* lainnya, asosiasi tipe ini menggunakan garis putus-putus atau garis lurus dengan mata panah terbuka di ujungnya.
   iii. Generalisasi antara dua *actor*, yaitu hubungan *inheritance* yang melibatkan *actor* yang satu dengan *actor* lainnya. Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.
   iv. Generalisasi antara dua *use case*, yaitu hubungan *inheritance* yang melibatkan *use case* yang satu dengan *use case* lainnya. Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.



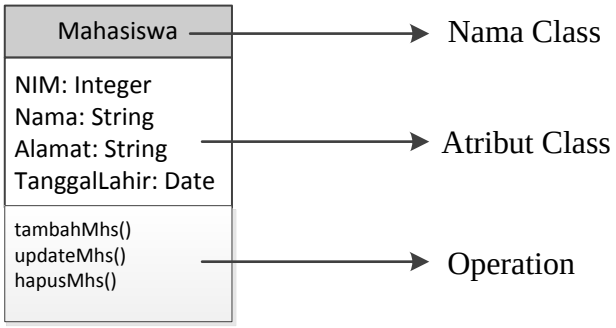
Gambar 2.5 Contoh relasi pada *use case* (Sumber: *The Elements of* UML, 2005)

4.2. *Class Diagram*

Class diagram merupakan diagram yang menggambarkan hubungan antar *class* yang memiliki atribut dan fungsi dari sebuah objek. Dalam penggunaannya, *class diagram* memiliki 3 relasi, yaitu:

a. *Association*, merupakan suatu hubungan yang menunjukkan ada interaksi yang terjadi antar *class*. Hal ini dapat ditunjukkan dengan garis dengan mata panah terbuka di ujungnya, yang mengindikasikan ada aliran pesan dalam satu arah.
b. *Generalization*, adalah suatu hubungan antar *class* yang bersifat dari khusus ke umum.
c. *Constraint*, merupakan suatu hubungan yang digunakan dalam sistem untuk memberi batasan pada sistem sehingga didapat aspek yang tidak fungsional.

Gambar di bawah ini merupakan contoh dari *class diagram*:



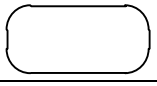
Gambar 2.6 Contoh *Class Diagram* (Sumber: *The Elements of* UML, 2005)

4.3. *Activity Diagram*

Activity diagram merupakan diagram yang menunjukkan konsep aliran data, aksi terstruktur dan dirancang dengan baik dalam sebuah sistem. Di dalam *Journal of Object Technology*: Conrad Bock disebutkan bahwa terdapat beberapa komponen dalam *Activity Diagram*, yaitu:

a. *Activity node*, menggambarkan bentuk notasi dari beberapa proses yang beroperasi dalam kontrol dan nilai data.
b. *Activity edge*, menggambarkan bentuk *edge* yang menghuungkan aliran aksi secara langsung yang menghubungkan *input* dan *output* dari aksi tersebut.
c. *Initial state*, bentuk lingkaran berisi penuh untuk melambangkan awal dari suatu proses.
d. *Decision*, merupakan bentuk wajib dengan suatu *flow* yang masuk beserta dua atau lebih *activity node* yang keluar. *Activity node* yang keluar ditandai untuk mengindikasikan beberapa kondisi.
e. *Fork*, merupakan satu bar hitam dengan satu *activity node* yang masuk beserta dua atau lebih *activity node* yang keluar.
f. *Join*, merupakan satu bar hitam dengan dua atau lebih *activity node* yang masuk beserta satu *activity node* yang keluar, tercatat pada akhir dari proses secara bersamaan. Semua *actions* yang menuju *join* harus lengkap sebelum proses dapat berlanjut.
g. *Final state*, bentuk lingkaran berisi penuh yang berada di dalam lingkaran kosong, menunjukkan akhir dari suatu proses.

Tabel 2.1 Simbol-simbol *activity diagram*

No	Simbol	Deskripsi
1.	Status awal 	Menunjukkan status awal dari aktivitas pada sistem. Dalam suatu diagram aktivitas mesti memiliki suatu status awal.
2.	Aktivitas 	Menunjukkan aktivitas yang dikerjakan oleh sistem. Biasanya aktivitas ini diawali dengan kata kerja.
3.	Percabangan/*decision* 	Menunjukkan asosiasi percabangan. Hal ini digunakan bila terdapat pilihan aktivitas lebih dari satu.

4.	Penggabungan/*join*	Menunjukkan asosiasi penggabungan, yaitu menggabungkan banyak aktivitas menjadi satu.
5.	Status akhir 	Menunjukkan status akhir dari aktivitas pada sistem. Dalam suatu diagram aktivitas mesti memiliki status akhir.
6.	*Swimlane* 	Berfungsi sebagai pemisah antara organisasi bisnis yang memiliki tanggung jawab terhadap aktivitas yang terjadi.

4.4. *Sequence Diagram*

Sequence diagram merupakan diagram yang menunjukkan kolaborasi dari beberapa objek yang saling berinteraksi antar elemen dari sebuah *class*. Berikut ini merupakan komponen dalam *sequence diagram*:

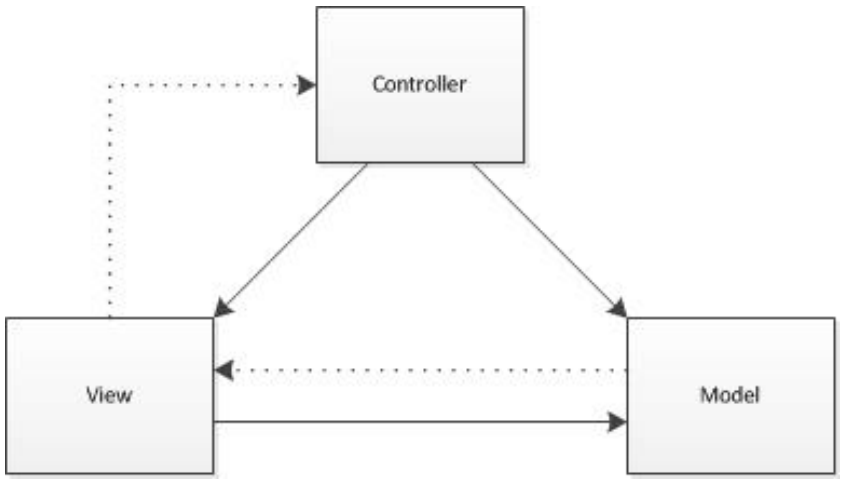
a. *Activations*, menjelaskan tentang eksekusi dari fungsi yang dimiliki oleh suatu objek.
b. *Actor*, menjelaskan tentang peran yang melakukan serangkaian aksi dalam suatu proses.
c. *Collaboration boundary*, menjelaskan tentang tempat untuk lingkungan percobaan dan digunakan untuk memonitor objek.
d. *Parallel vertical lines*, menjelaskan tentang suatu garis proses yang menunjuk pada suatu *state*.
e. *Processes*, menjelaskan tentang tindakan/aksi yang dilakukan oleh aktor dalam suatu waktu.
f. *Window*, menjelaskan tentang halaman yang sedang ditampilkan dalam suatu proses.
g. *Loop*, menjelaskan tentang model logika yang berpotensi untuk diulang beberapa kali.

2.5. MVC (*Model-View-Controller*)

Pemrograman berbasis objek telah berkembang pesat seiring dengan perkembangan PHP di versi 5. Saat ini banyak perusahaan lebih memilih menggunakan aplikasi *website* PHP karena aplikasi *website* saat ini sangatlah kompleks.

Kompleksnya sebuah *website* didukung dengan *script* pemrograman lainnya seperti ajax, jquery, css, html yang digabung menjadi sebuah aplikasi *website* dengan PHP sebagai *script* utama. Mulai dari aplikasi yang menggunakan API berbentuk SOAP, API WDSL, dan aplikasi *website* yang membutuhkan formulir yang kompleks.

MVC merupakan suatu metode pembuatan aplikasi *web* yang memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*). Konsep MVC pertama kali dipublikasikan oleh Trygve Reenskaug saat bekerja menggunakan *Smalltalk* di Xerox PARC. Dalam penerapannya kebanyakan *framework* yang digunakan dalam pembuatan *web* adalab berbasis arsitektur MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna dan bagian yang menjadi *control* dalam sebuah aplikasi *web*.



Gambar 2.7 Konsep MVC (Sumber: Hidayat & Surarso, 2012)

a. ***Model,*** merupakan sesuatu yang merepresentasikan data yang mengatur respon terhadap permintaan, serta memberi hak akses untuk memanipulasi data. Penggunaan model memungkinkan *developer* melakukan *query* antar *database* bila diperintahkan oleh *Controller* (*logic*). Kelebihan penggunaan *Model* adalah proses *maintenance* aplikasi yang lebih menguntungkan karena *detail* dari data dan operasinya dapat ditempatkan pada area yang ditentukan oleh *Model*. Kelebihan lainnya adalah komponen *Model* dapat digunakan

kembali oleh aplikasi lain yang memiliki fungsi yang hampir sama karena telah dipisahkan secara total antara data dengan tampilannya.

b. ***View***, merupakan informasi yang ditampilkan kepada pengguna sebagai media antarmuka. Komponen grafis menyediakan representasi proses internal aplikasi dan menuntun alur interaksi pengguna terhadap aplikasi yang ada. Contoh dari *view* adalah *template* dari tampilan pada suatu aplikasi atau *website* sehingga tidak ada *layer* lain yang berinteraksi dengan pengguna kecuali pada layer *view* saja. *View* memberikan kemudahan bagi *designer* sehingga dapat berkonsentrasi penuh pada desain tanpa harus memperhatikan hal-hal *detail* lainnya. Selain itu, *view* juga memungkinkan ketersediaan *multiple interface* dalam aplikasi namun mengeksekusi komponen *Model* sesuai dengan fungsionalitas yang diharapkan.

c. ***Controller***, melakukan segala aktifitas proses bisnis dan aktifitas *control* lainnya seperti mengolah data dari *Model*, menyimpannya dalam variabel-variabel (manipulasi data) dan menampilkannya pada *View*. Benar atau tidaknya hasil olahan data akan sangat tergantung dari logika kerja aplikasi yang tersusun pada bagian *Controller* ini. Dengan kata lain, *Controller* disebut sebagai bagian yang paling signifikan dari aplikasi berbasis MVC. *Controller* menyediakan rincian alur program dan bertanggung jawab menampung *events* dari pengguna melalui *View* dan meng-*update* komponen *Model* menggunakan data yang dimasukkan oleh pengguna.

Dengan menggunakan metode MVC maka dapat memberikan kemudahan dalam memelihara dan mengembangkan aplikasi. Untuk memahami metode pengembangan aplikasi dengan menggunakan MVC, diperlukan pengetahuan tentang pemrograman berorientasi objek (*Object Oriented Programming*).

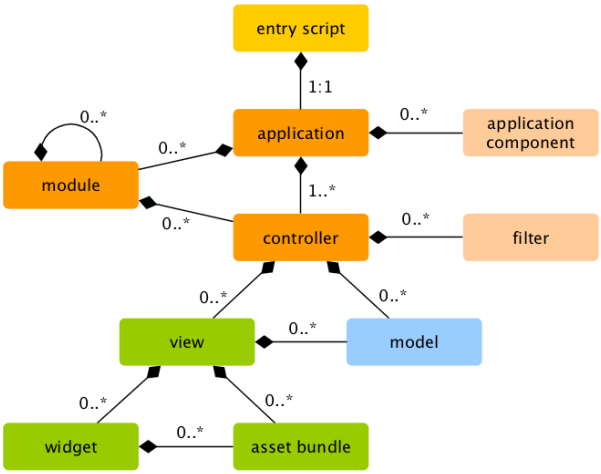
2.6. *Framework* Yii

Dikutip dari *website* resmi Yii, Yii merupakan sebuah *framework* PHP yang berbasis komponen, berkinerja tinggi untuk pengembangan aplikasi *web* dengan skala besar. *Framework* Yii menyediakan *reusability* maksimum dalam pemrograman *web* dan mampu meningkatkan kecepatan pengembangan secara

signifikan. Kata Yii merupakan singkatan dari "*Yes It Is*" yang memiliki makna bahwa *framework* ini mampu dan tepat dipilih untuk mengerjakan projek yang ingin dibuat.

Framework ini pertama kali dikembangkan oleh seorang *master* yang bernama Qiang Xue pada tahun 2008. Seperti kebanyakan *framework* PHP lainnya, Yii termasuk dalam kategori *framework* berbasis MVC (*Modul, View, dan Controller*). *Framework* Yii melampaui *framework* PHP lainnya dalam hal efisien, kekayaan fitur dan kejelasan dokumentasi. Dari awal pengerjaannya *framework* ini didesain dengan hati-hati agar sesuai untuk pengembangan aplikasi *web* secara serius. *Framework* Yii merupakan hasil dari pengalaman kaya para pengembang aplikasi *web* dan investigasi *framework* pemrograman *web* dan aplikasi yang paling populer.

Selain implementasi MVC, *framework* Yii juga memperkenalkan *front-controller* yang disebut aplikasi, yang bertugas mengenkapsulasi kontes eksekusi untuk memproses sebuah *request*. Aplikasi mengumpulkan beberapa informasi mengenai request pengguna dan kemudian mengirimnya ke *controller* yang sesuai untuk penanganan selanjutnya. Berikut ini merupakan diagram yang memperlihatkan struktur statis sebuah aplikasi Yii.



Gambar 2.8 Struktur statis aplikasi Yii (Sumber: https://www.yiiframework.com/doc/guide/2.0/id/structure-overview)

Dikutip dari *website* resmi Yii *Framework* (www. yiiframework.com), berikut ini penjelasan dari alur kerja umum sebuah aplikasi Yii:

a. Pengguna membuat permintaan dengan URL http:// www.contoh.com/index.php?r=post/show&id=1 dan server *web* menangani permintaan dengan menjalankan skrip bootstrapindex.php.
b. Skrip bootstrap membuat suatu *instance* aplikasi dan menjalankannya.
c. Aplikasi mendapatkan rincian informasi permintaan pengguna dari komponen aplikasi bernama *request*.
d. Aplikasi menentukan *controller* dan aksi yang diminta dengan bantuan komponen aplikasi bernama urlManager. Dalam contoh ini, *controller* adalah *post* yang merujuk pada kelas PostController, dan aksi adalah *show* yang dalam arti sebenarnya ditentukan oleh *controller*.
e. Aplikasi membuat *instance controller* yang diminta untuk selanjutnya menangani permintaan pengguna. *Controller* menentukan aksi *show* merujuk pada sebuah metode bernama *actionShow* dalam kelas *controller*. Kemudian membuat dari menjalankan *filter* (contoh kontrol akses pengukuran) terkait dengan aksi ini. Aksi dijalankan jika diizinkan oleh *filter*.
f. Aksi membaca *Post model* dimana ID adalah 1 dari *database*.
g. Aksi menyiapkan *view* (tampilan) bernama *show* dengan *model Post*.
h. *View* membaca dan menampilkan atribut *model Post*.
i. *View* menjalankan beberapa *widget*.
j. *View* menyiapkan hasil yang dipasangkan dalam *layout* (tata letak).
k. Aksi mengakhiri pembuatan *View* dan menampilkan hasil akhir pada pengguna.

Ada beberapa keunggulan yang terdapat pada *framework* Yii bila digunakan, antara lain:

a. Memiliki Gii *Code Generator*, merupakan sebuah *code generator* berbasis *web* yang bertujuan untuk membuat *code generator* CRUD (*Create, Read, Update, Delete*) dan modul.

b. *Roles* atau hak akses. Dengan menggunakan *framework* ini dapat menambahkan fungsi *accessRules*() untuk setiap *Class* yang berfungsi sebagai hak akses yang dapat kita tentukan siapa saja yang dapat mengakses fungsi atau *class* tersebut.
c. Terdapat sistem *filter* yang dapat memblokir *user* yang tidak terautentifikasi.
d. Memiliki *Class* CUrl*Manager* yang dapat mengganti URL biasa menjadi URL yang *user friendly*.
e. Terdapat sistem keamanan *Active Record*, sehingga dapat mengatasi terjadinya SQL *Injection*, *Cross-site Scripting* (XXS) dan *Cross-site Request Forgery* (CSRF).
f. Menggunakan metode MVC, sehingga memungkinkan pemisahan antara *layer application-logic* dan *presentation*.
g. Terdapat validasi *form* yang sudah tersedia *class built-in* dan *support* validasi AJAX.
h. Mendukung PHP *Data Object* (PDO). *Framework* Yii menyediakan API generik untuk mengakses data yang tersimpan dalam *Database Management System* (DBMS) yang berbeda, sehingga DBMS tersebut dapat diubah ke yang berbeda tanpa memerlukan perubahan kode yang menggunakan DAO untuk mengakses data.

2.7. Penelitian Terkait

Dalam mengerjakan penelitian ini, penulis tidak terlepas dari penelitian-penelitian sebelumnya yang pernah dilakukan untuk dijadikan sebagai bahan perbandingan dan referensi. Penelitian yang dijadikan referensi ini tidak terlepas dari topik penelitian yang penulis kerjakan saat ini. Berikut ini beberapa daftar penelitian yang penulis jadikan referensi dalam pembuatan penelitian ini:

Tabel 2.2. Daftar Penelitian Terkait

NO	Nama Peneliti dan Tahun	Judul
1.	Yanmin Cheng, Baotian Dong (2016)	*Analysis and Design of Campus e-commerce System*
2.	Syed Emdad Ullah, Tania Alauddin, Hasan U. Zaman (2016)	*Developing an e-commerce Website*

NO	Nama Peneliti dan Tahun	Judul
3.	Seyyedah Sanaz Mousavi Monfared dan Alli Kamandi (2016)	*Agile Techniques and Frameworks Based on The Requirements for e-commerce Applications*
4.	Shabur Miftah Maulana, Heru Susilo, Riyadi. (2015)	Implementasi *e-commerce* Sebagai Media Penjualan *Online* (Studi Kasus Pada Toko Pastbrik Kota Malang).
5.	Junay Diaz Arcanggih, Kertahadi, Riyadi. (2014)	Implementasi *e-commerce* Sebagai Media Promosi dan Penjualan Secara Elektronik (Studi Kasus pada Toko Jumbo Cell Bangil).
6.	Agung Wahana, Irvan Purliansyah. (2012)	Pembangunan e-commerce (Penjualan *Online*) pada Turpez Shop.
7.	Arip Aryanto, Tri Irianto Tjendrowasono. (2012)	Pembangunan Sistem Penjualan *Online* pada Toko Indah Jaya Furniture Surakarta.
8.	Yaser Ahangari Nanehkaran. (2013)	*An Introduction to Electronic Commerce.*
9.	Dewi Irmawati (2011)	Pemanfaatan *e-commerce* dalam Dunia Bisnis.
10.	Glenn Johnson (2013)	*Training Guide: Programming in* HTML5 *with* JavaScript *and* CSS3.