BAB 2
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Beberapa Penelitian sebelumnya membahas tentang marketplace dan progressive web app, salah satunya penelitian Iskandar, Edi and Buwono, Robby Cokro and Putri, Sintia Ogi Nindya (2020) dengan judu*l IMPLEMENTASI PROGRESSIVE WEB APPS PADA MARKETPLACE*. Pengembangan beberapa platform atau menggunakan pendekatan lintas platform Aplikasi Web Progresif, dapat diterapkan melalui serangkaian konsep dan teknologi pada semua situs web yang memenuhi persyaratan tertentu. Aplikasi web progresif sebagai kemungkinan pemersatu teknologi untuk aplikasi web dan aplikasi lainnya. Setelah pengenalan fitur, pemeriksaan kinerja, dilakukan perbandingan antara kedua aplikasi yaitu aplikasi seluler lintas platform dan Aplikasi Web Progresif, dan disediakan repository open source untuk verifikasi validitas hasil. Tujuan penelitian untuk mengetahui seberapa efektifkah dalam pengembangan terhadap marketplace..

Penelitian ini dibagi menjadi dua bagian yaitu teoritis dan empiris. Bagian teoritis terdiri dari literatur yang dikumpulkan dari buku, artikel dan jurnal sedangkan bagian empiris menyajikan studi kasus tentang pasar hewan tradisional. Pada Penelitian ini menunjukan bahwa aplikasi ini menyediakan fitur seperti aplikasi web, pekerja layanan mendukung PWA untuk menjadi solusi prioritas utama di pasar hewan e-niaga. Sebagai hasil prakteknya, penelitian ini melahirkan webshop dengan service worker terintegrasi. Pada pembuktiannya pekerja layanan dapat membuat aplikasi web lebih mudah dan bekerja secara offline dengan menggunakan sumber daya dan cache cerdas.

Menurut Tjarco Kersens, dengan judul *Applicability of Progressive Web Apps in Mobile Development*. Penelitian ini bertujuan untuk menganalisis penerapan PWA sebagai alternatif dari pengembangan aplikasi asli dan web tradisional, untuk memberikan argumen kepada pengembang untuk pertimbangan pendekatan pengembangan dan untuk mengisi kesenjangan pengetahuan dalam literatur akademis tentang pengembangan seluler. Perbandingan yang digunakan yaitu performa dan konsumsi energi PWA

dengan native dan web implementasi aplikasi di iOS dan Android, peneliti menemukan bahwa versi PWA memiliki kinerja yang serupa di banyak kasus, lebih baik di beberapa dan hanya lebih buruk dalam waktu peluncuran di iOS.

Menurut beberapa penelitian Moh. Abdur Rashid Al Aziz , Pudji Sriyanto, Agung Budianto Achmad yang berjudul Pengawasan dan Tata Laksana Pemeriksaan Kesehatan Ternak Sapi Di Pasar Hewan Babat Dan Tikung Kabupaten Lamongan. Penelitian ini bertujuan untuk mengetahui pengawasan dan tata laksana pemeriksaan kesehatan ternak sapi di pasar hewan. Pasar hewan Babat dan Tikung merupakan unit pelaksana teknis yang dikelola oleh Perusahaan Umum Daerah Kabupaten Lamongan yang memiliki sarana penunjang antara lain timbangan ternak, pusat kesehatan hewan, administrasi, dan sistem transportasi.

Tabel 2.1 Perbandingan Teori dalam Marketplace Pasar Hewan Digital

No	Deskripsi	Tahun	Peneliti	Institusi
1	Pengawasan dan Tata Laksana Pemeriksaan Kesehatan Ternak Sapi Di Pasar Hewan Babat Dan Tikung Kabupaten Lamongan	2020	Moh. Abdur Rashid Al Aziz , Pudji Sriyanto, Agung Budianto Achmad	Universitas Airlangga, Surabaya
2	Applicability of Progressive web apps in Mobile Development	2019	Kerssens Tjarco.	Thesis. Universiteit Van Amsterdam
3	Mobile Commerce Beyond Electronic Commerce: Issue And Challenges	2012	Jahanshahi, Alireza dan Amin	Asian Journal of Business and Management Sciences
4	Implementasi Progressive Web APPs pada Marketplace	2020	Iskandar, Edi and Buwono, Robby Cokro and Putri, Sintia Ogi Nindya	STMIK AKAKOM Yogyakarta, Yogyakarta.
5	Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek	2016	Rosa dan Muhammad Shalahudin	Informatika Bandung

Peneliti melihat banyak fragmentasi di platform seluler sehingga beberapa pengembang telah mencari alternatif dalam pengembangan lintas platform. Salah satu metodologi baru yang disebut Aplikasi Web Progresif (PWA) bertujuan untuk menyatukan web dan pendekatan asli dengan menggabungkan prinsip terbaik dari keduanya, sehingga menjembatani kesenjangan tersebut. Peneliti juga memberikan informasi rekam medis ternak dan informasi ternak yang ada di dalam deskripsi hewan ternak dalam platform konten aplikasi tersebut.

2.2. Landasan Teori

2.2.1 Konsep Dasar Program

Menurut Harumy, dkk (2016:4), "program adalah formulasi sebuah algoritma dalam bentuk bahasa pemrograman, sehingga siap untuk dijalankan pada mesin komputer". Membuat program seperti memberitahukan apa yang harus dilakukan kepada orang lain. Sebagai contoh, pada saat memberitahukan algoritma membuat telur dadar kepada orang lain, kita sudah melakukan pemrograman.

Dari definisi program tersebut di atas, secara sederhana program adalah rangkaian instruksi-instruksi dalam bahasa komputer yang disusun secara logika dan sistematis.

Pada saat ini metode, berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti mentransformasi hasil dari satu tahap ke tahap berikutnya, misalnya pada metode pendekatan terstruktur. Aplikasi yang dikembangkan saat ini sangat beragam dan berbagai macam *platform*, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi atau *platform* tersebut. Berikut ini adalah beberapa konsep dasar tentang metodologi pemrograman berbasis objek:

a. Kelas (*class*)

Menurut Rosa dan Shalahudin (2013:104) kelas adalah "kumpulan objek- objek dengan karakteristik yang sama". Sebuah kelas akan mempunyai sifat (atribut), kelakukan (operasi / metode), hubungan (relationship) dan arti dari kelas tersebut.

b. Objek (*object*)

Menurut Rosa dan Shalahudin (2013:106) "objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur dan status, atau hal-hal yang bersifat abstrak". Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan. Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Sehingga sebuah objek hanya ada saat sebuah program dieksekusi, jika masih dalam bentuk kode, disebut sebagai kelas.

c. Metode (*method*)

Operasi atau metode atau method pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek (Rosa dan Shalahuddin, 2013:1)

d. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya.

2.2.2 *Mobile Commerce*

Menurut Argade dan Chavan (2015:112), "*mobile commerce* (*m-commerce*) adalah kemampuan pengiriman *e-commerce* langsung ke tangan konsumen dimana saja melalui teknologi nirkabel". Menurut Jahanshahi dkk (2011:122) "secara umum, *m-commerce* mengacu pada setiap transaksi yang menggunakan nilai uang yang dilakukan melalui jaringan telekomunikasi *mobile*". Proses penjualan dan pembelian barang dan jasa dapat dilakukan secara *mobile*. Proses pembayaran dan transaksi *online* pun juga dapat dilakukan secara *online* melalui layanan bank, sehingga meminimalisir penggunaan uang secara tunai. Bahkan proses pemesanan, penentuan lokasi dapat dilakukan di dalamnya.

1) Struktur *Mobile Commerce*

Konsep *mobile commerce* mirip seperti *electronic commerce* yaitu

mengacu ke tipe bisnis yang berbasis elektronik melalui koneksi internet. Perbedaan krusialnya adalah proses pembelian dan penjualan menggunakan jaringan nirkabel dengan bantuan perangkat *mobile*. *Mobile payment* adalah kunci komponen dari sebuah transaksi m-commerce dengan memanfaat salah satu dari model *mobile payment* seperti *premium* SMS atau WAP berbasis portal.

Jika dibandingkan dengan electronic commerce sistem, *mobile commerce* sistem lebih kompleks dikarenakan komponen *mobile commerce* berhubungan dengan komputasi.

2) Alur Transaksi *Mobile Commerce*

Sebuah *m-commerce* memiliki alur yang melibatkan pengguna *m-commerce*, vendor telekomunikasi, vendor layanan lainnya yang dalam beberapa hal juga terlibat (misalkan pemerintah, pengembang aplikasi, pihak ketiga, pihak vendor perangkat *mobile*). Alur dimulai dari bagaimana sebuah perangkat *mobile* sudah tersedia layanan untuk koneksi ke internet, beserta dengan aplikasi *client* yang diperlukan atau aplikasi *web browser*. Kemudian pengguna melalui perangkat *mobile*-nya tersebut dapat segera mengakses layanan *m- commerce* untuk memperoleh informasi produk yang diinginkannya. Info ini tentu saja disediakan oleh pelaku *e-commerce* di sisi penjual (yang dapat melakukannya melalui *m-commerce*).

Di Dalam *m-commerce* akan terjadi pertukaran data (*data exchange*), pertukaran informasi (*information exchange*), penyajian informasi, transaksi *online*, verifikasi transaksi dan verifikasi pembayaran, sehingga menjadikan proses transaksi lebih cepat, lebih mudah, lebih efektif dan memiliki nilai lebih jika dibandingkan dengan transaksi *e-commerce* dan transaksi konvensional. Jalur komunikasi dan pertukaran data serta informasi, disajikan melalui jaringan internet yang disediakan oleh vendor telekomunikasi maupun *internet service provider* (ISP). Beberapa buah penyedia layanan *m- commerce* mulai menyediakan aplikasi terintegrasi untuk hal ini.

3) Sistem Kerja *Mobile Commerce*

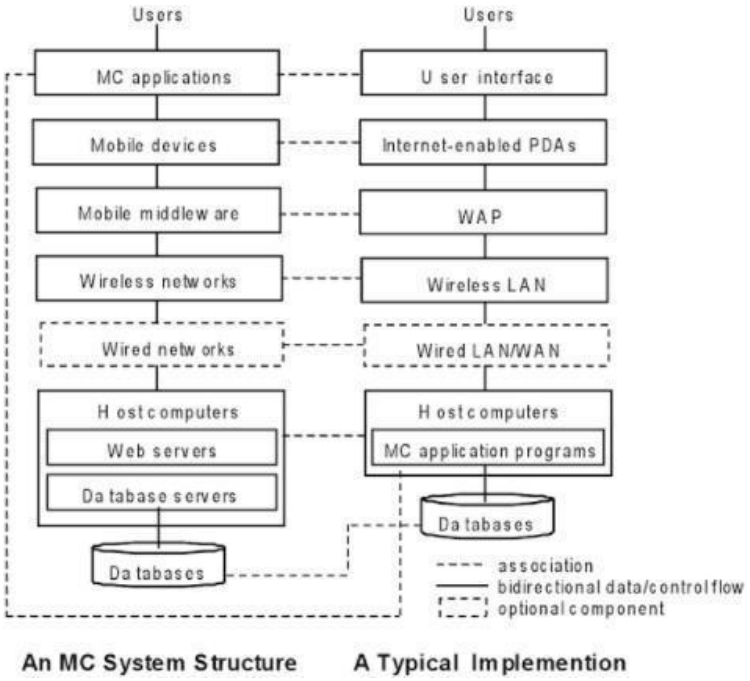
Menurut (Pratama:114) sebuah aplikasi dan layanan *m-commerce* memiliki sistem kerja sebagai berikut:

a) Pengguna mengakses aplikasi dan layanan *m-commerce* dari

perangkat *mobile* mereka dengan koneksi internet. Misalkan menggunakan *smartphone*, *handphone*, *tablet*, yang terhubung dengan jaringan internet.

b) Pengguna mengakses toko *online* yang diinginkan dan mencari produk yang diinginkannya memasukkan ke keranjang belanja virtual sebagaimana halnya *e-commerce* pada umumnya, kemudian melanjutkan ke proses pembayaran secara online. Pembayaran melibatkan sistem manajemen untuk transaksi retail (*Retail Transaction Management System*) yang melibatkan pihak bank tempat konsumen menjadi nasabah. Terdapat PIN, SSL, dan *login* pengguna, untuk autentikasi dan peningkatan keamanan transaksi secara elektronik yang terjadi.

c) Semua proses berbasiskan koneksi internet secara *online* dan *mobile*. Informasi lebih singkat, simpel dan tepat sasaran. Mengingat bahwa tampilan *mobile* umumnya lebih singkat dan terbatas dibandingkan *desktop*.



Gambar 2.1. Struktur *Mobile Commerce*

Sumber: Hu (2009:88)

2.2.2 *Progressive Web Apps*

Desain web yang responsif kini telah diterapkan berbagai developer demi meningkatkan performa situsnya. Tahun 2013 yang lalu bahkan dijuluki sebagai "Tahun Desain Web Responsif" dan Google mengumumkan Mobilegeddon pada bulan April 2015 ketika mereka mulai meningkatkan peringkat situs web yang mobile-friendly dalam hasil mobile search.

(https://*www.dewaweb.com/blog/progressive-web-apps/* 13 juni 2021 jam 16.20

Antarmuka *Progressive Web Apps* menggunakan dasar konsep manipulasi langsung gerakan multi sentuh. Kontrol antarmukanya meliputi *slider*, *switch*, dan tombol. Karakteristik Progressive Web App antara lain

1) PWA bersifat progresif

PWA bekerja untuk semua user, tidak peduli browser apa yang Anda gunakan atau dimana Anda mengakses website tersebut. Jadi tidak peduli apakah Anda menggunakan Chrome atau Opera, atau Anda mengakses website dari Indonesia atau Amerika Latin, Anda tetap dapat membuka website tersebut dengan waktu loading yang cepat. Progressive Web App akan bekerja dengan baik karena mereka dibangun dengan peningkatan progresif sebagai prinsip intinya.

2) Responsive

Progressive Web Apps juga akan bekerja untuk semua device, baik itu desktop, mobile, atau tablet.

3) Web Apps tidak tergantung dengan konektivitas

Dengan bantuan service workers, PWA tetap bisa bekerja dengan jaringan yang lemah.

4) Terasa seperti aplikasi

PWA dirancang agar terasa seperti aplikasi. Mereka memiliki gaya interaksi dan navigasi seperti aplikasi handphone.

5) Selalu up-to-date

Karena adanya service workers, PWA akan selalu up-to-date.

6) Mudah ditemukan

Sesuai dengan manifest W3C, klasifikasi Progressive Web Apps adalah sebagai aplikasi. PWA juga akan lebih mudah ditemukan

oleh mesin pencari berkat adanya service workers.

7) Meningkatkan user engagement

Dengan fitur-fitur PWA seperti push notifications, ini membantu meningkatkan *user engagement*.

8) Instalasi

Pengguna dan pengunjung website dapat menyimpan progressive web apps yang sering mereka akses langsung ke home screen handphone mereka sehingga Anda tidak perlu mendownload aplikasi dari app stores.

9) Linkable

PWA bisa dengan mudah dibagikan ke orang lain dengan URL tanpa proses instalasi yang complex.

2.2.3 Bahasa Pemrograman

Berikut adalah bahasa pemrograman yang digunakan untuk mengembangkan *aplikasi mobile* dan halaman website admin.

Framework untuk Membangun Progressive Web Apps

1. AngularJS

AngularJS adalah framework berbasis Javascript untuk membangun aplikasi responsif yang kuat dan handal. Framework ini pertama kali diperkenalkan oleh Google pada tahun 2009 dan menjadi salah satu framework Progressive Web Apps yang paling populer. Angular versi 5 memiliki service workers untuk memudahkan proses development. Service workers ini didesain untuk mengoptimalkan pengalaman pengguna ketika mereka menggunakan koneksi yang lambat atau buruk. Angular versi 6 yang sudah dirilis hadir dengan dua perintah CLI tambahan untuk menyederhanakan proses downloading dan installing web aplikasi pada perangkat.

2. React

React merupakan framework Progressive Web Apps untuk library Javascript. Framework ini banyak digunakan untuk mengembangkan PWA karena menawarkan tingkat fleksibilitas yang tinggi. React juga memiliki beberapa kelebihan lain seperti kemampuan rendering yang cepat dengan Virtual-DOM, ekosistem yang luas, memiliki komunitas

terbesar serta didukung oleh Facebook.

3. *Hypertext Markup Language* (HTML)

Menurut Anhar (2010:40) pengertian "*Hypertext Markup Language* (HTML) adalah sekumpulan simbol-simbol atau tag-tag yang dituliskan dalam sebuah *file* yang digunakan untuk menampilkan halaman pada *web browser*". HTML pertama kali dikembangkan oleh Tim Berners-Lee bersamaan dengan protokol HTTP (*Hypertext Transfer Protocol*) pada tahun 1989. Tujuan utama pengembangan HTML adalah untuk menghubungkan satu halaman web dengan halaman web lainnya.

Pada dasarnya, setiap halaman web yang ditulis dalam bentuk HTML. HTML merupakan bahasa pemrograman web yang memberitahukan web browser bagaimana menyusun dan menyajikan konten di halaman web. Dengan kata lain, HTML ada pondasi web. HTML disusun dengan bahasa yang sederhana, sehingga sangat mudah diimplementasikan. Saat ini, HTML dapat menampilkan obyek-objek seperti teks, tabel, tautan, gambar, audio dan video.

4. Javascript

Menurut Hernita (2010:18) Javascript merupakan "bahasa yang berbentuk kumpulan skrip yang berfungsi untuk memberikan tampilan yang tampak lebih interaktif pada dokumen web". Dengan kata lain, bahasa ini adalah bahasa pemrograman untuk memberikan kemampuan tambahan ke dalam bahasa pemrograman HTML dengan mengizinkan pengeksekusian perintah-perintah pada sisi *client*, dan bukan sisi *server* dokumen web.

Pada hakikatnya, bahasa pemrograman Javascript berisi skrip yang pemasangannya terselip di sebuah dokumen HTML. Sehingga bahasa Javascript ini tidak memerlukan sebuah kompilator atau penerjemah khusus untuk mengaksesnya. Hal tersebut juga bergantung pada *navigator* yang terdapat di setiap *web browser*.

2.2.4 *Tools* Pendukung

Berikut ini adalah *tools-tools* pendukung yang digunakan untuk mengembangkan *aplikasi mobile* dan halaman *website* admin.

a. IDE (*Integrated Development Environment*)

Menurut Tiano (2015:2) "Xcode adalah IDE (*Integrated Development Environment*) lengkap yang dimiliki Apple untuk membangun perangkat lunak dari segala macam *platform* Apple". Pertama kali digunakan untuk mengembangkan sistem operasi OS X dan diperluas untuk pengembangan iOS, watchOs dan tvOS. Xcode memiliki beberapa fitur utama, seperti IDE menggunakan format file .xcodeproj untuk memanajemen semua kode, asset dan konfigurasi file baik yang dibuat maupun file yang diimpor.

b. Coda 2

Coda 2 merupakan *text editor* untuk sistem operasi Mac OS yang berfungsi membangun aplikasi website. Coda 2 juga memiliki fitur *live preview* yaitu fitur untuk melihat hasil tampilan *website* yang telah diketik tanpa perlu membuka *web browser* untuk melihat hasilnya.

2.2.5 Laravel *Database*

Laravel *database* adalah *database* yang dibangun di cloud *server*. Data disimpan dalam bentuk *javascript object notation* (JSON) dan disinkronisasi secara langsung atau *real time* ke setiap klien yang terhubung. Firebase akan mensinkronisasi data baik aplikasi yang berbasis android, iOS, ataupun *website* dengan berbagi satu *realtime database* dan secara otomatis akan menerima pembaruan terbaru.

A. Fungsi Utama Firebase *Realtime Database*

1) Realtime

Dibandingkan dengan menggunakan *Hypertext Transfer Protocol* (HTTP) *request*, *firebase realtime database* menggunakan sinkronisasi data setiap ada perubahan pada data, setiap perangkat yang terhubung akan menerima pembaruan dalam hitungan milidetik.

2) Offline

Perangkat yang terhubung dengan *firebase realtime database* akan tetap responsif bahkan saat *offline* karena *firebase* SDK menyimpan data ke penyimpanan lokal. Setelah koneksi berhasil tersambung, perangkat klien akan menerima pembaruan yang terlewat dan akan mensinkronisasi dengan data terbaru yang berada di *server*.

3) Dapat diakses dari perangkat klien

Laravel database dapat diakses dari perangkat *mobile* maupun *website*,

tanpa memerlukan aplikasi *server*. Laravel *database* menyediakan *realtime database security rules* yaitu sebuah aturan kapan data tersebut harus dibaca ataupun disimpan.

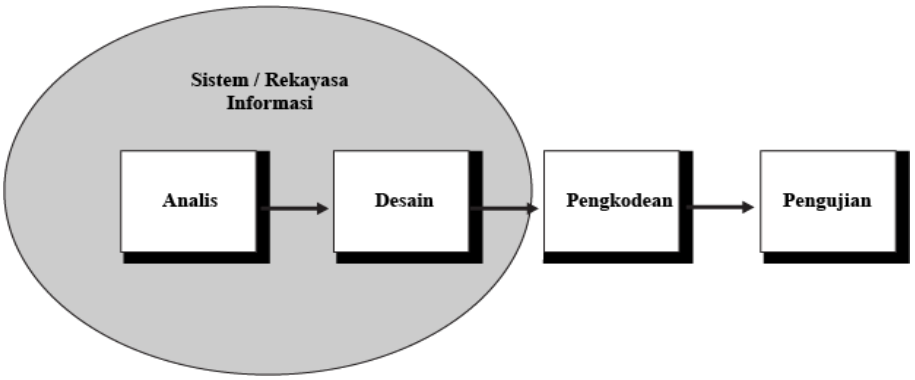
B. Cara Laravel *Database* Bekerja

Laravel *Realtime Database* memungkin untuk membangun aplikasi dengan kaya fitur dengan mengizinkan akses aman ke *database* secara langsung melalui kode dari sisi klien. Data disimpan secara lokal, bahkan saat *offline* dengan fitur *real time* yang terus berjalan. Saat perangkat berhasil tersambung koneksi internet, *Realtime database* akan mensinkronisasi perubahan data lokal dengan pembaruan terbaru yang berada di *server* dengan menggabungkan perbedaan apapun secara otomatis.

Realtime Database menggunakan basis data berbasis NoSQL dan karena itu memiliki optimisasi dan fungsionalitas yang berbeda dengan *database* berbasis relasional. *Realtime Database API* didesain untuk melakukan operasi yang dieksekusi secara cepat. Dengan ini memungkinkan untuk membangun aplikasi yang dapat diakses secara *real time* oleh jutaan pengguna tanpa mengorbankan secara respon.

2.2.6 Model Pengembangan Perangkat Lunak

Model pengembangan perangkat lunak air terjun (*waterfall*) sering juga disebut dengan model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*) (Rosa dan Shalahuddin, 2013: 28). Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau berurutan dimulai dari analis, desain, pengkodean, pengujian, dan tahap pendukung (*support*). Berikut adalah ilustrasi model *waterfall*.



Gambar 2.2. Ilustrasi model *waterfall*

1) Analisis Kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu didokumentasikan.

2) Desain

Menurut Rosa dan Shalahudin (2013:29) "desain perangkat lunak adalah proses multilangkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengkodean". Tahap ini menterjemahkan kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3) Pengujian

Pengujian fokus pada perangkat lunak secara dari segi logika dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

4) Pendukung (*support*) atau pemeliharaan (*maintenance*)

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*, Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengurangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.3 Teori Pendukung

2.3.1 Entity Relationship Diagram (ERD)

Menurut Lubis (2013:8) "*entity relationship diagram* (ERD) adalah suatu pemodelan berbasis pada persepsi dunia nyata yang mana terdiri dari kumpulan objek dasar yang disebut dengan entitas (*entity*) dan hubungan diantara objek tersebut dengan menggunakan perangkat konseptual dalam bentuk diagram".

ERD untuk menggambarkan pemodelan struktur data dan hubungan antar data, digunakan notasi dan simbol.

a. Tipe Relasi

Menurut Yanto (2016:38) ada tiga macam relasi menurutnya yaitu

1. *Unary*, yaitu relasi yang menghubungkan entitas yang sejenis.
2. *Binary*, yaitu relasi yang menghubungkan entitas yang tidak sejenis.
3. *Ternary*, yaitu relasi yang menghubungkan lebih dari dua entitas yang tidak sejenis.

b. Derajat Kardinalitas

Menurut Yanto (2016:38) derajat kardinalitas merupakan penjabaran dari hubungan entitas. Derajat kardinalitas terbagi menjadi atas tiga bagian yaitu:

i. Satu ke satu (*one to one*)

Relasi satu ke satu yang berarti bahwa tiap entitas dapat dihubungkan dengan paling banyak satu entitas yang lain, demikian sebaliknya.

ii. Satu ke banyak (*one to many*)

Relasi satu ke banyak yang berarti bahwa tiap entitas dapat dihubungkan dengan lebih satu entitas atau lebih, tetapi tidak sebaliknya lebih dari satu (banyak) entitas hanya berhubungan dengan satu entitas yang lain.

iii. Banyak ke banyak (*many to many*)

Relasi banyak ke banyak yang berarti bahwa banyak entitas dapat dihubungkan dengan banyak entitas yang lain.

2.3.2 Unified Modeling Language (UML)

Menurut Gata dan Gata (2013:4) "*Unified Modeling Language* (UML) merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem". UML hadir karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasis UML

adalah sebagai berikut:

a. *Use Case* Diagram

Menurut Gata dan Gata (2013:4) "*Use case* diagram merupakan pemodelan untuk kelakuan (*behaviour*) sistem informasi yang akan dibuat". *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

b. *Class* Diagram

Menurut Gata dan Gata (2013:6) "*Class diagram* merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem".

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan pembatas yang berhubungan dengan objek yang dihubungkan. Di dalam *class* diagram atribut dan operasi memiliki sebuah *visibility*, yang berguna untuk enkapsulasi komponen, berikut adalah *visibility* atribut dan properti:

I. *Private*:

Diawali dengan tanda "-", atribut dan operasi tersebut hanya bisa diakses oleh *class* itu sendiri.

II. *Public*:

Diawali dengan tanda "+", atribut dan operasi tersebut hanya dapat diakses dari semua objek.

III. *Protected*:

Diawali dengan tanda "#", atribut dan operasi tersebut hanya dapat diakses oleh *class* itu sendiri atau turunan *class* tersebut.

2.3.3 Activity Diagram

Menurut Rosa dan Shalahudin (2013:161) *activity* diagram "menggambarkan *workflow* (aliran kerja) atau aktivitas digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi". *Activity diagram* menggambarkan aktivitas sistem bukan apa yang dilakukan oleh aktor, jadi aktivitas dapat dilakukan oleh sistem. *Activity* diagram juga banyak digunakan untuk mendefinisikan hal-hal

berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

2.3.4. Sequence Diagram

Menurut Rosa dan Shalahudin (2013:165) "*Sequence* diagram menggambarkan kelakukan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek". Oleh karena itu untuk menggambar *sequence* diagram maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode- metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *sequence* diagram juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

Banyak sequence diagram yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada *sequence* diagram sehingga semakin banyak *use case* yang didefinisikan maka sequence diagram yang harus dibuat juga semakin banyak.

2.3.5 Pengujian Perangkat Lunak

Sebuah perangkat lunak perlu dijaga kualitasnya bahwa kualitas bergantung kepada kepada kepuasan pelanggan (*customer*). Perangkat lunak sering mengandung kesalahan (*error*) pada proses tertentu pada saat perangkat lunak sudah berada di tangan *user*. Kesalahan-kesalahan (*error*) pada perangkat lunak seperti ini sering disebut dengan *bug*. (Rosa dan Shalahuddin, 2013:271).

Untuk menghindari banyaknya *bug* maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan. Adanya *bug* adalah suatu yang biasa, bahkan sebuah perangkat lunak yang sudah besar dan terkenal pun biasanya masih ada *bug*. Yang bisa dilakukan pengembang perangkat lunak adalah meminimalisir

bug dengan melakukan pengujian.

Menurut Rosa dan Shalahudin (2013:272) pengujian adalah "salah satu set aktifitas yang direncanakan dan sistematis untuk menguji dan mengevaluasi kebenaran yang diinginkan". Aktivitas pengujian terdiri dari satu set atau sekumpulan langkah dimana dapat menempatkan desain kasus uji yang spesifik dan metode pengujian. Salah satu metode untuk pengujian menggunakan *black box testing*.

Menurut Rosa dan Shalahudin (2013:272) "*Black box testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program". Pengujian yang dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian *black box* 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* harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses *login* maka kasus uji yang dibuat adalah:

1) Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
2) Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.