BAB 2
TINJAUAN PUSTAKA

2.1 Ikan Arwana

Arwana merupakan ikan hias air tawar yang cukup populer. Arwana dikenal sebagai hewan langka yang berstatus terancam punah dan berharga tinggi. Arwana juga disebut *arowana* atau arwana, karena merupakan ikan dengan sebutan nama latin *Osteoglossumbicirrchossum* dari *genus Osteoglossum* yang berasal dari Brasil. Di Indonesia, ikan yang mencuat ke seluruh dunia sebagai ikan hias, populer dengan sebutan ikan naga perak dan ikan naga perak hitam. Hingga saat ini arwana tergolong ikan hias abadi karena kepopulerannya tidak pernah pudar, bahkan terus meningkat. Namun demikian, tidak semua jenis arwana memiliki kepopuleran yang sama [2].

Gambar 2.1 ikan Arwana super red

2.2 Tingkat Kekeruhan Air

Tingkat kekeruhan air adalah suatu studi dari sifat-sifat optis yang menyebabkan cahaya yang melewati air menjadi terhambur dan terserap dari cahaya yang dipancarkan dalam garis lurus (Fairuz dan Zubir, 2009). Kekeruhan menyebabkan air menjadi seperti berkabut atau berkurangnya transparansi dari air. Arah dari berkas cahaya yang dipancarkan akan berubah ketika cahaya berbenturan dengan partikel di dalam air. Jika level kekeruhan rendah maka sedikit cahaya yang akan dihamburkan dan dibiaskan dari arah asalnya. Tingkat kekeruhan air (turbidity) dapat diketahui dengan menggunakan turbidimeter [3].

2.3 Pakan

Pakan merupakan salah satu aspek penting yang harus diperhatikan dalam kegiatan pemeliharaan ikan, sebab pakan merupakan sumber energi untuk menunjang pertumbuhan. Pakan yang baik adalah pakan yang sesuai dengan kebutuhan fisiologi dan spesies ikan yang dipelihara. Disamping mampu untuk memenuhi kebutuhan nutrisi ikan tersebut, pemberian pakan dengan kualitas dan kuantitas yang baik dapat mengoptimalkan perkembanggan ikan. Pakan harus tersedia dalam jumlah yang cukup, terus menerus (kontinu), dan mempunyai kandungan gizi yang dibutuhkan untuk pertumbuhan ikan.

2.4 Konsep Perancangan B*error*ientasi Objek

OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang b*error*ientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

Konsep OOP, terbagi menjadi 4 kelas, yaitu sebagai berikut:

1. Kelas Abstrak (*Class Abstract*)
   a. Kelas merupakan deskripsi abstrak in*form*asi dan tingkah laku dari sekumpulan data.
   b. Kelas dapat diilustrasikan sebagai suatu cetak biru(blueprint) atau prototipe yang digunakan untuk menciptakan objek.
   c. Kelas merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal.

d. Kelas mendefinisikan suatu struktur yang terdiri atas data kelas (*data field*), prosedur atau fungsi (*method*), dan sifat kelas (*property*).

2. Enkapsulasi (*encapsulation*)

a. Istilah enkapsulasi sebenarnya adalah kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail in*form*asi.

b. Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.

c. Enkapsulasi menekankan pada antarmuka suatu kelas, atau dengan kata lain bagaimana menggunakan objek kelas tertentu.

d. Contoh: kelas mobil menyediakan antarmuka fungsi untuk menjalankan mobil tersebut, tanpa kita perlu tahu komposisi bahan bakar, udara dan kalor yang diperlukan untuk proses tersebut.

3. Pewarisan (*Inheritance*)

a. Kita dapat mendefinisikan suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada.

b. Penurunan sifat ini bisa dilakukan secara bertingkattingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik.

c. Sub kelas memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya.

d. Dengan konsep pewarisan, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kode-kode itu.

4. Polimorfisme (*polymorphism*)

a. Polimorfisme merupakan kemampuan objekobjek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.

b. Polimorfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan *method* mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.

c. *Method overriding*.

d. *Method name overloading*.

2.5 Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk visualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti JAVA, C++, *Visual Basic*, atau bahkan dihubungkan secara langsung ke dalam sebuah *object-oriented database*. Bagian - bagian utama dari UML adalah *View, diagram*, model element, dan *general mechanism* [4].

1. View

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *View* dalam UML antara lain : *use case View, logical View, component View, concurrency View*, dan *deployment View*.

2. *Use case View*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan external actors. *Actor* yang berinteraksi dengan sistem dapat berupa *User* atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *View* ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

3. *Logical View*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,* dan *relationship*) dan kolaborasi dinamis yang terjadi ketika object mengirim pesan ke object lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *class* diagrams untuk struktur statis dan dalam *state, sequence, collaboration*, dan *activity* diagram untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

4. *Component View*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan in*form*asi *administrative* lainnya. *View* ini digambarkan dalam component *View* dan digunakan untuk pengembang (*developer*).

5. *Concurrency View*

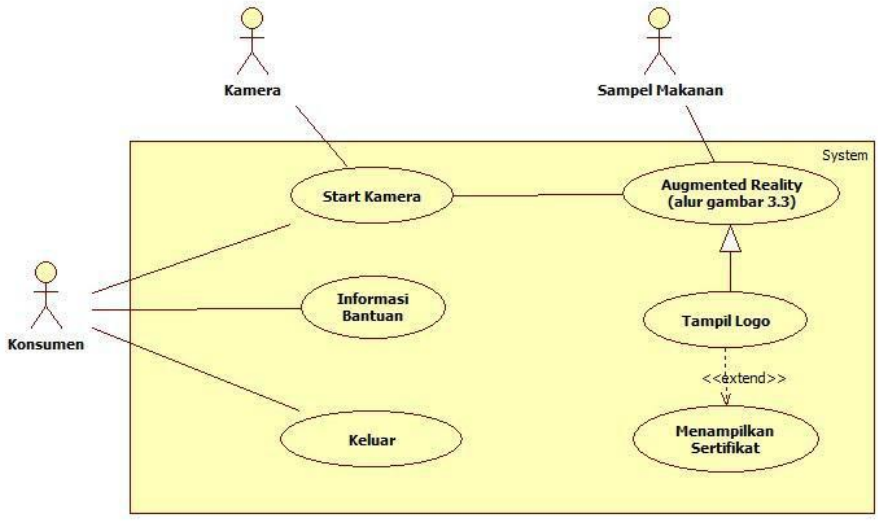
Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration,* dan *activity diagrams*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untuk pengembang , pengintegrasi, dan penguji.

6. Deployment View

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan yang lain. *View* ini digambarkan dalam deployment diagrams dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*). Berikut ini menjelaskan tentang diagram-diagram yang ada di UML:

1. Use Case Diagram

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *User* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *User*. Sedangkan *use case diagram* memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.



Gambar 2.2 Use Case Diagram

Diagram Use Case berguna dalam tiga hal :

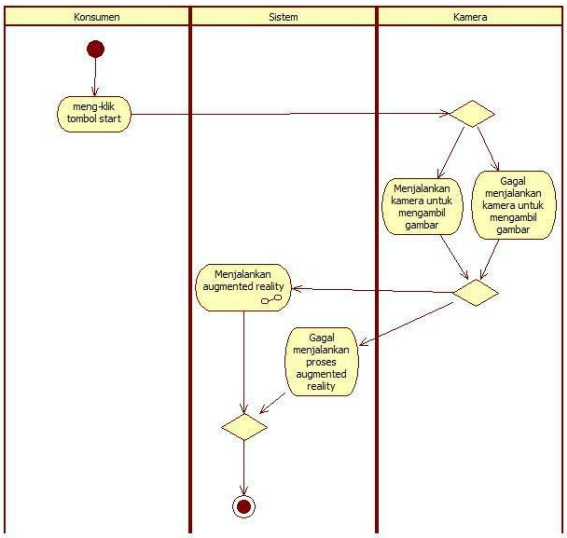
a. Menjelaskan fasilitas yang ada *(requirements*) : *Use Case* baru selalu menghasilkan fasilitas baru ketika sistem di analisa, dan design menjadi lebih jelas.

b. Komunikasi dengan klien : Penggunaan notasi dan simbol dalam diagram *Use Case* membuat pengembang lebih mudah berkomunikasi dengan klien-kliennya.

c. Membuat test dari kasus-kasus secara umum : Kumpulan dari kejadian-kejadian untuk *Use Case* bisa dilakukan test kasus layak untuk kejadian-kejadian tersebut.

2. Activity Diagram

Pada dasarnya diagram *Activity* sering digunakan oleh *flowchart*. Diagram ini berhubungan dengan diagram *Statechart*. Diagram *Statechart* berfokus pada obyek yang dalam suatu proses (atau proses menjadi suatu obyek), diagram *Activity* berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Sebagai contoh, perhatikan proses yang terjadi. "Pengambilan uang dari bank melalui ATM." Ada tiga aktifitas kelas (orang, dan lainnya) yang terkait yaitu : *Customer*, ATM, and Bank. Proses berawal dari lingkaran *start* hitam pada bagian atas dan berakhir di pusat lingkaran *stop* hitam/putih pada bagian bawah. Aktivitas digambarkan dalam bentuk kotak persegi. Lihat gambar di bawah ini, agar lebih jelas :

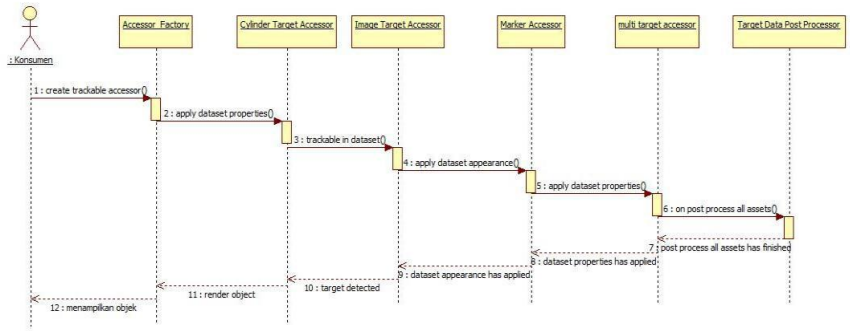


Gambar 2.3 Activity Diagram

Diagram Activity dapat dibagi menjadi beberapa jalur kelompok yang menunjukkan obyek mana yang bertanggung jawab untuk suatu aktifitas. Peralihan tunggal (*single transition*) timbul dari setiap adanya *activity* (aktifitas), yang saling menghubungi pada aktifitas berikutnya. Sebuah *transition* (transisi) dapat membuat cabang ke dua atau lebih percabangan *exclusive transition* (transisi eksklusif). Label *Guard Expression* menerangkan *output* (keluaran) dari percabangan. Percabangan akan menghasilkan bentuk menyerupai bentuk intan. T*ransition* bisa bercabang menjadi beberapa aktifitas paralel yang disebut *Fork***.** *Fork* beserta *join* (gabungan dari hasil *output fork*) dalam diagram berbentuk *solid bar* (batang penuh).

3. Sequence Diagram

Diagram Class dan diagram *Object* merupakan suatu gambaran *model statis*. Namun ada juga yang bersifat *dinamis*, seperti Diagram *Interaction*. Diagram *sequence* merupakan salah satu *diagram Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan; *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Di bawah ini adalah diagram Sequence untuk pembuatan Hotel *Reservation*. Obyek yang mengawali urutan *message* adalah „*aReservation Window'*.



Gambar 2.4 Sequence Diagram

Pemanggilan diri sendiri disebut dengan iterasi. *Expression* yeng dikurung dengan "[ ]", adalah *condition* (keadaan kondisi). Pada diagram dapat

dibuat *note* (catatan). Pada gambar, terlihat seperti selembar kertas yang berisikan teks. *Note* bisa diletakan dimana saja pada diagram UML.

4. Class Diagram

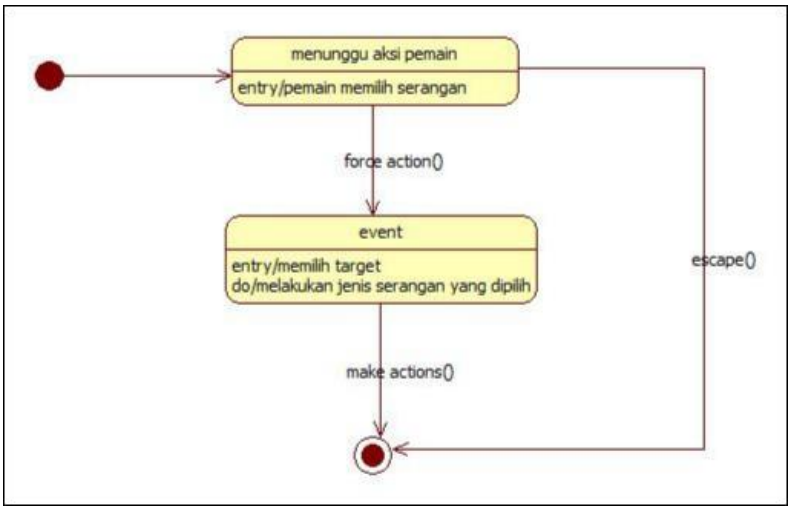
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain b*error*ientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class* diagram menggambarkan struktur dan deskripsi *class*, package dan objek beserta hubungan satu sama lain seperti *containment* , pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu nama (dan *stereotype*), Atribut, dan Metoda.

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar class yang bersangkutan
b. *Protected*, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
c. *Public*, dapat dipanggil oleh siapa saja

5. *State Machine Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (daribsatu *state* ke*state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



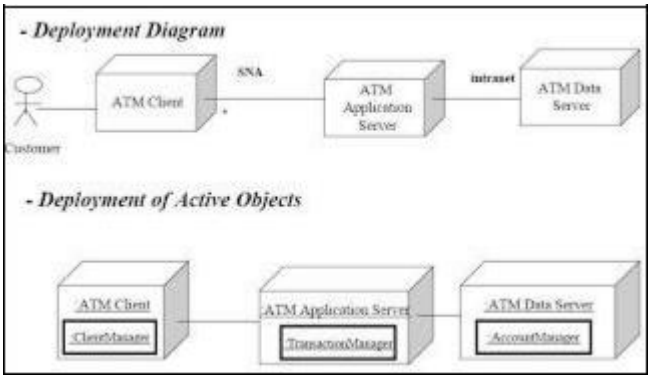
Gambar 2.5 State Machine Diagram

6. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponenbpiranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code* , baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time* , maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

7. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server,*workstation* , atau piranti keras lain yang digunakan untuk men- *deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar 2.6 Deployment Diagram

Tujuan Penggunaan UML adalah sebagai berikut:

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
3. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah (*blue print)* cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bias diketahui in*form*asi secara detail tentang coding program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk diagram (*reserve enginering*).

Dalam penelitian ini UML digunakan untuk memodelkan proses bisnis yang terjadi dan mendokumentasikan system perangkat lunak , sehingga memudahkan dalam mengembangkan sebuah rancangan sistem .

2.6 Android

Android adalah sebuah sistem operasi pada *handphone* yang bersifat terbuka dan berbasis pada sistem operasi Linux. Android bisa digunakan oleh setiap orang yang ingin menggunakannya pada perangkat mereka. Android menyediakan plat*form* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk bermacam peranti bergerak. Awalnya, Google Inc.

membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T- Mobile, dan Nvidia. Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode–kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Berikut merupakan kelebihan dan kekurangan dari *Android* [5].

1. Kelebihan Android
   a. Multitasking

      Android yang mampu membuka beberapa aplikasi sekaligus tanpa harus menutup salah satunya.

   b. Kemudahan dalam Notifikasi

      Setiap ada SMS, Email, atau bahkan artikel terbaru dari RSS Reader, akan selalu ada notifikasi di *Home Screen* Ponsel Android dengan Lampu LED Indikator yang berkedip-kedip.

   c. Akses Mudah Aplikasi *Android* lewat Google Android App Market
      Dengan Google Android App Market dapat mendownload berbagai aplikasi dengan gratis maupun berbayar. Ada banyak ribuan aplikasi dan games yang siap untuk didownload di ponsel *Android*.

   d. Pilihan Ponsel yang beranekaragam

      Android tersedia di ponsel dari berbagai produsen, mulai dari Sony Ericsson, Motorola, HTC sampai Samsung.Setiap pabrikan ponsel pun menghadirkan ponsel *Android* dengan gaya masing-masing.

   e. Bisa menginstal ROM yang dimodifikasi

      Banyak *Costum* ROM yang bisa dipakai di ponsel *Android* guna merubah tampilan.

   f. Widget

      Dengan adanya *Widget* di *homescreen* pengguna bisa dengan mudah mengakses berbagai setting dengan cepat dan mudah.

   g. Google Maniac

Kelebihan *Android* lainnya jika pengguna setia layanan Google mulai dari Gmail sampai Google Reader, ponsel Android telah terintegrasi dengan layanan Google, sehingga dapat dengan cepat mengecek email dari Gmail.

2. Kelemahan *Android*
   a. Koneksi Internet yang terus menerus

      Ponsel berbasis sistem ini memerlukan koneksi internet yang simultan/terus menerus aktif. Koneksi internet GPRS selalu aktif setiap waktu, dimana pengguna harus siap berlangganan paket GPRS yang sesuai dengan kebutuhan.

   b. Iklan

      Aplikasi di Ponsel *Android* memang bisa didapatkan dengan mudah dan gratis, namun konsekuensinya di setiap aplikasi tersebut, akan selalu iklan yang terpampang, entah itu bagian atas atau bawah aplikasi.

Dalam Aplikasi ini *android* digunakan untuk menampilkan mengatur dan menampilkan informasi tentang suhu serta pakan dan minum ayam.

2.6.1 Android Studio

Android Studio adalah sebuah IDE untuk pengembangan aplikasi di *platform Android*. Saat ini usia Android Studio masih tergolong muda, baru versi 0.2.3 (masih *early access preView*). SDK sebelumnya di-*bundle* bersama dengan Eclipse, sementara Android Studio menggunakan IntelliJ IDEA Community Edition. Beberapa pendukung IntelliJ IDEA sering mengatakan bahwa Eclipse terlalu rumit bagi pemula. Android Studio menggunakan *Gradle* untuk manajemen proyeknya. *Gradle* adalah *build automation tool* yang dapat dikonfigurasi melalui DSL berbasis Groovy. Ini yang membedakan *Gradle* dari *Ant* atau *Maven* yang memakai XML. Penggunaan DSL berbasis Groovy menyebabkan Gradle lebih fleksibel dan dapat diprogram dengan mudah.

Android Studio *build system* digunakan untuk membangun, test, menjalankan, dan membuat paket dari aplikasi yang dibangun. Build system tidak tergantung (*independent*) dari Android Studio, jadi *developer* dapat memanggilnya

dalam Android Studio atau dengan menggunakan *command line*. Setelah menuliskan code pada aplikasi yang akan dibangun, *developer* dapat menggunakan *fitur build system* untuk:

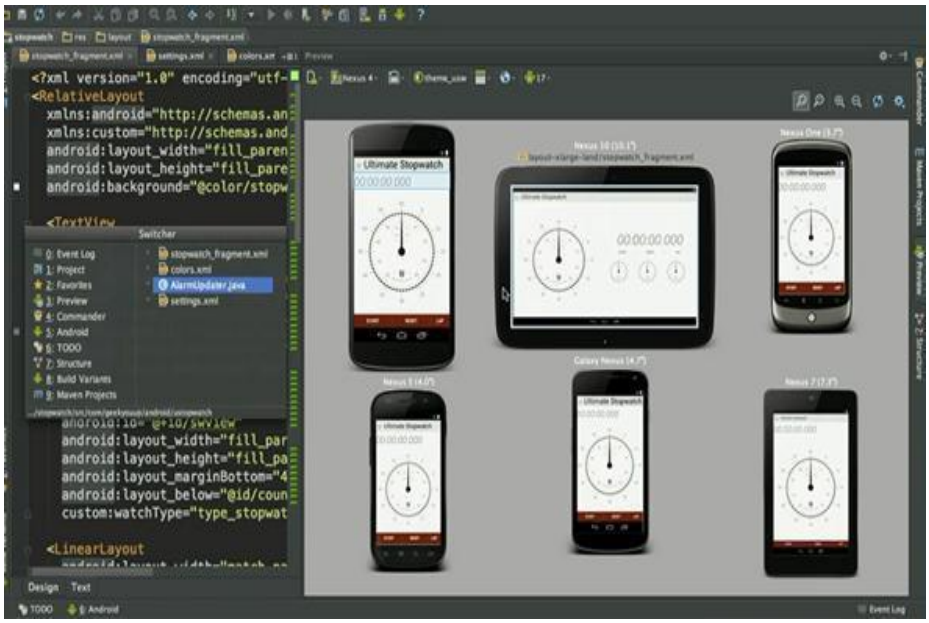
1. Kostumisasi, konfigurasi, dan meng-*extends* proses *build*
2. Membuat beberapa apk untuk aplikasi android dengan fitur yang berbeda menggunakan project yang sama
3. Menggunakan ulang *code* dan *resources*

Fleksibilitas dari system pengembangan Android Studio memungkinkan untuk mencapai keseluruhan hal ini tanpa harus memodifikasi file inti dari *project*. Android Studio ini adalah lingkungan pengembangan baru dan terintegrasi penuh, yang baru saja dirilis oleh Google untuk sistem operasi *Android*. Android Studio dirancang untuk menjadi peralatan baru dalam pengembangan aplikasi dan juga memberi alternative lain selain Eclipse yang saat ini menjadi IDE yang paling dipakai. Saat memulai proyek baru dengan Android Studio, struktur proyek akan muncul bersama dengan hampir semua berkas yang ada di dalam direktori SDK, peralihan ke sistem manajemen berbasis Gradle ini memberikan fleksibilitas yang lebih besar pada proses pembangunannya.

Android Studio developer dapat melihat perubahan visual apapun yang dilakukan pada aplikasi secara langsung.Dapat terlihat perbedaannya jika dipasang pada beberapa perangkat Android berbeda, termasuk konfigurasi dan resolusi secara bersamaan. Fitur lain di Android Studio adalah alat alat baru untuk *packing* dan memberi label kode. Dengan begitu mengijinkan *developer* tetap menjadi yang teratas ketika berurusan dengan banyak kode. Program ini juga menggunakan sistem *drag and drop* untuk memindahkan komponen melalui *interface User.*

Selain itu, lingkungan baru ini juga mendukung *Google Cloud Messaging*. Sebuah fitur yang mengijinkan untuk mengirim data dari server ke perangkat *Android* melalui *cloud*, cara terbaik untuk mengirim *Reminder* pada apps. Program ini juga membantu untuk melokalisasi aplikasi anda, memberi gambaran visual untuk tetap memprogram sambil mengontrol alur dari aplikasi. Kemudahan yang diberikan oleh Android Studio adalah sebagai berikut :

1. Lingkungan pengembangan yang mantap dan bersifat langsung.
2. Cara termudah untuk menguji performa pada perangkat dengan tipe lain.
3. Wizard dan template berisi elemen – elemen umum yang ada di semua pemrograman Android.
4. Editor dengan fitur lengkap dengan banyak peralatan ekstra untuk mempercepat pengembangan aplikasi anda. Layout editor yang memungkinkan untuk *drag and drop* komponen UI, pratinjau *layout* pada beberapa konfigurasi layar, dan banyak lagi.
5. Berbasis Gradle.
6. Android spesifik refactoring dan perbaikan yang cepat.
7. Alat Lint untuk menangkap kinerja, kegunaan, versi kompatibilitas dan masalah lainnya.
8. ProGruard dan app-*signature*.
9. Built-in dukungan untuk *Google Cloud platform*, sehingga mudah untuk mengintegrasikan *Google Cloud Messaging* dan *App Engine* sebagai komponen *server-side*.

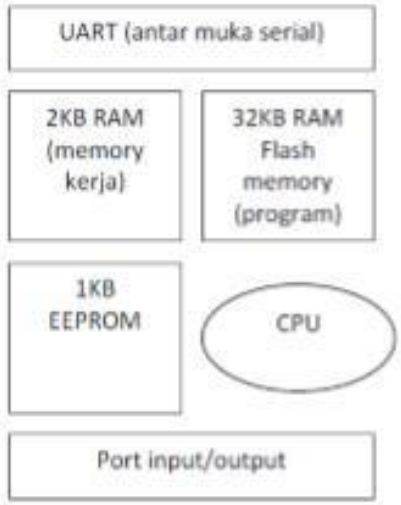


Gambar 2.7 Android Studio

Dalam Penelitian ini android studio digunakan untuk membangun aplikasi android dibuat.

2.7 Arduino

Arduino adalah *platform* pembuatan prototipe elektronik yang bersifat opensource hardware yang berdasarkan pada perangkat keras dan perangkat lunak yang fleksibel dan mudah digunakan.Arduino pada awalnya dikembangkan di Ivrea, Italia. Plat*form* arduino terdiri dari arduino *board, shield*, bahasa pemrograman arduino, dan arduino *development environment*. Arduino *board* biasanya memiliki sebuah chip dasar *mikrokontroler* Atmel AVR ATmega8 berikut turunannya. Bahasa pemrograman arduino adalah bahasa pemrograman yang umum digunakan untuk membuat perangkat lunak yang ditanamkan pada arduino board. Bahasa pemrograman arduino mirip dengan bahasa pemrograman C++. Arduino berfungsi sebagai pengatur kontrol alat. Arduino menjalankan perintah sesuai program yang diberikan dan memproses banyak data dari inputan sensor. Gambarn Blok pada Arduino dapat dilihat dalam gambar dibawah.



Gambar 2.8 Gambaran blok pada Arduino

Gambaran blok yang ada pada Arduino. Terdiri dari blok memori, blok komunikasi, blok I/O dan blok prossesor. Perangkat masukan dan keluaran atau sering dikenal I/O mikrokontroler adalah suatu perangkat yang menghubungkan mikrokontroler dengan dunia luar (rangkaian lain), perangkat ini dibutuhkan sebagai media komunikasi dengan perangkat lain atau perubah tipe sinyal. Beberapa perangkat I/O yang dimiliki ATmega328 antara lain:

1. UART (*Universal Asynchronous Receiver Transmitter*) digunakan sebagai komunikasi serial.
2. SPI (*Serial Peripheral Interface*) merupakan port komunikasi serial sinkron.
3. 12C bus (*Intergrated Circuit bus*) merupakan antamuka serial bus yang dikembangkan oleh philips.
4. *Analog to Digital Conversion* (ADC) adalah rangkaian yang digunakan untuk mengubah data analog ke data digital.

2.8 Sensor Turbidity

Sensor Turbidity adalah ukuran kekeruhan air. Kekeruhan telah menunjukkan tingkat di mana air kehilangan transparansi. Ini dianggap sebagai ukuran kualitas air yang baik. Kekeruhan menghalangi cahaya yang dibutuhkan oleh vegetasi air yang terendam. Ini juga dapat meningkatkan suhu air permukaan di atas normal karena partikel yang tersuspensi di dekat permukaan memfasilitasi penyerapan panas dari sinar matahari [6].

Gambar 2.9 Sensor Turbidity [9]

2.9 Motor Servo

Motor servo adalah sebuah motor dengan sistem closed feedback di mana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri dari sebuah motor, serangkaian gear, potensiometer dan rangkaian kontrol. Potensiometer berfungsi untuk menentukan batas sudut dari putaran servo. Sedangkan sudut dari sumbu motor servo diatur

berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor. Tampak pada gambar dengan pulsa 1.5 mS pada periode selebar 2 mS maka sudut dari sumbu motor akan berada pada posisi tengah. Semakin lebar pulsa OFF maka akan semakin besar gerakan sumbu ke arah jarum jam dan semakin kecil pulsa OFF maka akan semakin besar gerakan sumbu ke arah yang berlawanan dengan jarum jam.

Motor servo biasanya hanya bergerak mencapai sudut tertentu saja dan tidak kontinyu seperti motor DC maupun motor stepper. Walau demikian, untuk beberapa keperluan tertentu, motor servo dapat dimodifikasi agar bergerak kontinyu. Pada robot, motor ini sering digunakan untuk bagian kaki, lengan atau bagianbagian lain yang mempunyai gerakan terbatas dan membutuhkan torsi cukup besar [7].

Gambar 2.10 Motor Servo [7]

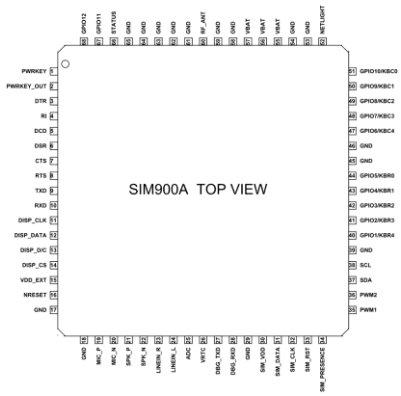
2.10 Modul SIM900A

SIM900A adalah modul SIM yang digunakan pada penelitian ini. Modul SIM900 GSM/GPRS adalah bagian yang berfungsi untuk komunikasi antara mikrokontroler Arduino dengan Web Service.

Ada banyak perintah AT yang sudah disebutkan di datasheet SIMCOM SIM900A GSM Module. Ada dua inisialisasi perintah pada modul SIM900A yaitu Perintah inisialisasi GSM dan Perintah inisialisasi GPRS Koneksi Internet [8].

Untuk mengirim data Modul komunikasi GSM/GPRS menggunakan core IC SIM900A. Modul ini mendukung komunikasi dual band pada frekuensi 900

/ 1800 MHz (GSM900 dan GSM1800) sehingga fleksibel untuk digunakan bersama kartu SIM dari berbagai operator telepon seluler di Indonesia. Operator GSM yang beroperasi di frekuensi dual band 900 MHz dan 1800 MHz sekaligus: Telkomsel, Indosat, dan XL. Operator yang hanya beroperasi pada band 1800 MHz: Axis dan Three.



Gambar 2.11 Pin-Pin Pada SIM900A

Tampilan dari konfigurasi pin GSM SIM900. Modul ini sudah terpasang pada breakout-board (modul inti dikemas dalam SMD/ Surface Mounted Device packaging) dengan pin header standar 0,1" (2,54 mm) sehingga memudahkan penggunaan, bahkan bagi penggemar elektronika pemula sekalipun.

2.11 Web Service

Web service adalah aplikasi sekumpulan data (*database*), perangkat lunak (*software*) atau bagian dari perangkat lunak yang dapat diakses secara remote oleh berbagai piranti dengan sebuah perantara tertentu. Secara umum,*web service* dapat diidentifikasikan dengan menggunakan URL seperti hanya web pada umumnya. Namun yang membedakan *web service* dengan web pada umumnya adalah interaksi yang diberikan oleh *web service*. Berbeda dengan URL *web* pada umumnya, URL *web service* hanya menggandung kumpulan in*form*asi, perintah, konfigurasi atau sintaks yang berguna membangun sebuah fungsi-fungsi tertentu dari aplikasi [9].

Web service dapat diartikan juga sebuah metode pertukaran data, tanpa memperhatikan dimana sebuah*database* ditanamkan, dibuat dalam bahasa apa sebuah aplikasi yang mengkonsumsi data, dan di plat*form* apa sebuah data itu dikonsumsi. *Web service* mampu menunjang interoperabilitas. Sehingga *web*

service mampu menjadi sebuah jembatan penghubung antara berbagai sistem yang ada.

Menurut W3C *Web services Architecture Working Group* pengertian *Web service* adalah sebuah sistem *software* yang di desain untuk mendukung interoperabilitas interaksi mesin ke mesin melalui sebuah jaringan. *Interface web service* dideskripsikan dengan menggunakan *form*at yang mampu diproses oleh mesin (khususnya WSDL). Sistem lain yang akan berinteraksi dengan *web service* hanya memerlukan SOAP, yang biasanya disampaikan dengan HTTP dan XML sehingga mempunyai korelasi dengan standar Web (Web Services Architecture Working Group, 2004).

Web pada umumnya digunakan untuk melakukan *respon* dan *request* yang dilakukan antara *client* dan *server*. Sebagai contoh, seorang pengguna layanan *web* tertentu mengetikan alamat *url web* untuk membentuk sebuah *request*. *Request* akan sampai pada *server*, diolah dan kemudian disajikan dalam bentuk sebuah respon. Dengan singkat kata terjadilah hubungan *client-server* secara sederhana.

Sedangkan pada *web service* hubungan antara *client* dan *server* tidak terjadi secara langsung. Hubungan antara *client* dan *server* dijembatani oleh file *web service* dalam *form*at tertentu. Sehingga akses terhadap *database* akan ditanggani tidak secara langsung oleh *server*, melainkan melalui perantara yang disebut sebagai *web service*. Peran dari *web service* ini akan mempermudah distribusi sekaligus integrasi database yang tersebar di beberapa *server* sekaligus.

2.12 JavaScipt Object Notation (JSON)

JavaScript Object Notation (JSON) adalah *form*at pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. JSON merupakan *form*at teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena *form*at data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut:

1. Objek

   Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua)dan setiap pasangan nama/nilai dipisahkan oleh , (koma).
2. Larik

   Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).
3. Nilai

   Nilai (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah *objek* atau sebuah larik.

4. String

   String adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

5. Angka

   Angka adalah sangat mirip dengan angka di C atau Java, kecuali *form*at octal dan heksadesimal tidak digunakan.

6. Spasi kosong (*whitespace*)

   Spasi kosong dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

2.13 PHP

PHP (*Hypertext Prepocessor)* merupakan bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML. PHP diciptakan oleh Rasmus Lerdorf pertama kali tahun 1994. Pada awalnya PHP adalah singkatan dari "*Personal Home Page Tools*". Selanjutnya diganti menjadi FI ("*Form*s Interpreter"). Sejak versi 3.0, nama bahasa ini diubah menjadi "PHP: *Hypertext Prepocessor*" dengan singkatannya "PHP". PHP versi terbaru adalah versi ke-5. Berdasarkan *survey* Netcraft pada bulan Desember 1999, lebih dari sejuta site menggunakan PHP, di antaranya adalah NASA, Mitsubishi, dan RedHat.

PHP adalah bahasa pemrograman script server-side yang didesain untuk pengembangan web, tetapi juga bisa digunakan sebagai bahasa pemrograman umum *(wikipedia). PHP* pertama kali di kembangkan pada tahun 1995 oleh *Rasmus Lerdorf*, namun sekarang dikelola oleh *The PHP Group*. Situs resmi PHP beralamat di *http://www.php.net*. Pada awalnya PHP adalah singkatan dari *Personal Home Page*, namun karena dalam perkembangannya PHP tidak hanya digunakan untuk membuat halaman web pribadi, PHP saat ini merupakan singkatan dari PHP:

Hypertext Preprocessor, sebuah kepanjangan rekursif, yakni permainan kata dimana kepanjangannya berisi juga singkatan itu sendiri.

Sistem kerja dari PHP diawali dengan permintaan yang beasal dari halaman website oleh *browser*. Berdasarkan URL atau alamat website dalam jaringan internet, *browser* akan menemukan sebuah alamat dari webserver, mengidentifikasi halaman yang dikehendaki, dan menyampaikan segala in*form*asi yang dibutuhkan oleh *webserver*. Selanjutnya *webserver* akan mencarikan berkas yang diminta dan menampilkan isinya di *browser*. *Browser* yang mendapatkan isinya segera menerjemahkan kode HTML dan menampilkannya.

2.14 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris : *database management system*) atau DBMS yang *multithread*, *multi-User*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi *GNU General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah *SQL* yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, *MySQL* mendukung operasi basisdata transaksional maupun operasi basis data non-transaksional. Pada modus operasi non-transaksional, *MySQL* dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi *blogging* berbasis web , CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional [10].

2.15 JavaScript

JavaScript merupakan bahasa skrip yang populer di internet dan dapat bekerja di sebagian besar penjelajah web populer seperti *Internet Explorer* (IE), Mozilla Firefox, Netscape dan Opera. Kode *JavaScript* dapat disisipkan ke dalam halaman *web* menggunakan *tag SCRIPT*. *JavaScript* adalah bahasa (pemrograman) yang hebat, sulit dan kurang begitu dipahami pada bidang bahasa pemrograman di komputer. Akan tetapi, kemampuan inti yang dimiliki oleh JavaScript untuk membuat aplikasi-aplikasi sehingga membuat perubahan dari cara pandang pemakaian Internet pada dekade ini. Salah satu contoh yang baik dari aplikasi tersebut adalah *Google Maps* [11].

Keunggulan JavaScript, yang juga dikenal sebagai ECMAScript, hadir pada semua web browser, sehingga memiliki kemampuan untuk menghasilkan keluaran yang sama pada semua *platform* yang didukung oleh *browser*. Seperti yang sudah disebutkan di atas, Google Maps, yang dapat berjalan di atas Linux, Windows dan Mac OS. Dengan pertumbuhan pustaka (*library*) javascript baru-baru ini, hal ini memudahkan untuk menavigasi dokumen, memilih elemen DOM, membuat animasi, menangani *event*, dan mengembangkan aplikasi Ajax. Berbeda teknologi terkenal lain yang didorong oleh kepentingan komersial, JavaScript benar-benar teknologi yang *cross-platform*, bahasa pemrograman *client-side* yang bersifat bebas (untuk dimodifikasi dan gratis tentunya) dan diadopsi secara universal.

JavaScript dieksekusi pada client side (komputer pengguna): Sebuah server website mengirim javascript ke peramban milik pengguna, dan *browser* tersebut menginterpretasikan dan menjalankan kodenya. Semua ini terjadi dalam sebuah sandbox, yang menjaga agar javascript tidak menyentuh *internal* sistem, sehingga mencegah *malicious code* (kode jahat) menginfeksi komputer pengguna.

2.16 HTML

HyperText Markup Language (HTML) adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai in*form*asi di dalam sebuah penjelajah web Internet dan pem*form*atan hiperteks sederhana yang ditulis dalam berkas *form*at ASCII agar dapat menghasilkan tampilan wujud

yang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam *form*at ASCII normal sehingga menjadi halaman web dengan perintah-perintah HTML. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML (*Standard Generalized Markup Language*), HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium* (W3C). HTML dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di CERN pada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa).

HTML dokumen tersebut mirip dengan dokumen tulisan biasa, hanya dalam dokumen ini sebuah tulisan bisa memuat instruksi yang ditandai dengan kode atau lebih dikenal dengan TAG tertentu. Sebagai contoh jika ingin membuat tulisan ditampilkan menjadi tebal seperti: TAMPIL TEBAL, maka penulisannya dilakukan dengan cara: < b> TAMPIL TEBAL . Tanda < b> digunakan untuk mengaktifkan instruksi cetak tebal, diikuti oleh tulisan yang ingin ditebalkan, dan diakhiri dengan tanda untuk menonaktifkan cetak tebal tersebut.

HTML lebih menekankan pada penggambaran komponen-komponen struktur dan *form*at di dalam halaman web daripada menentukan penampilannya. Sedangkan penjelajah web digunakan untuk menginterpretasikan susunan halaman ke gaya built-in penjelajah web dengan menggunakan jenis tulisan, tab, warna, garis, dan perataan *text* yang dikehendaki ke komputer yang menampilkan halaman web. Salah satu hal Penting tentang eksistensi HTML adalah tersedianya *Lingua franca* (bahasa Komunikasi) antar komputer dengan kemampuan berbeda. Pengguna Macintosh tidak dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat dalam pc berbasis Windows. Pengguna Microsoft Windows pun tidak akan dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat pada pengguna yang menggunakan Produk-produk Sun Microsystems. namun demikian pengguna-pengguna tersebut dapat melihat semua halaman web yang telah di*form*at dan berisi Grafika dan Pranala [12].

HTML5 dan JavaScript telah menjadi sangat menarik (dan menyenangkan) dengan munculnya kerangka kerja dan lanskap mobile. PhoneGap adalah sebuah framework open source untuk cepat membangun aplikasi cross-plat*form* mobile menggunakan HTML5, JavaScript dan CSS. Prosedur pengembangan aplikasi *cross-platform* adalah sebagai bungkus aplikasi Anda dengan PhoneGap dan menyebarkan ke *platform mobile*. Membangun aplikasi untuk setiap perangkat membutuhkan kerangka kerja dan bahasa yang berbeda. PhoneGap memecahkan ini dengan menggunakan standar berbasis teknologi Web untuk menjembatani aplikasi Web dan perangkat *mobile*. Sebuah aplikasi Web tradisional hanya tidak bekerja tanpa jaringan. Salah satu solusi untuk masalah ini adalah dengan menggunakan dua fitur dari HTML5 Standar:

1. *Offline* aplikasi Web
2. Penyimpanan *database Client-side*.

Pengguna dapat menggunakan fungsi awan pada perangkat *mobile*, bekerja secara *offline* dengan aplikasi dikerahkan secara lokal pada database lokal, dan berbagi data dengan sisa awan ketika akan online lagi.

2.17 Metode Kualitatif

Metode penelitian kualitatif merupakan sebuah cara yang lebih menekankan pada aspek pemahaman secara mendalam terhadap suatu permasalahan. Penelitian kualitatif ialah penelitian riset yang bersifat deskriptif dan cenderung menggunakan analisis serta lebih menonjolkan proses dan makna. Tujuan dari metodologi ini ialah pemahaman secara lebih mendalam terhadap suatu permasalahan yang dikaji. Dan data yang dikumpulkan lebih banyak kata ataupun gambar-gambar daripada angka.

Adapun ciri pokok metode penelitian kualitatif ada lima, yaitu antara lain:

1. Menggunakan lingkungan alamiah sebagai sumber data : Sumber data yang digunakan dalam penelitian kualitatif berupa lingkungan alamiah. Kajian utama dalam penelitian kualitatif yaitu peristiwa-peristiwa yang terjadi dalam kondisi dan situasi sosial. Penelitian dilakukan ketika berinteraksi langsung di tempat kejadian. Peneliti melakukan pengamatan, mencatat, mencari tahu, menggali sumber yang berkaitan

dengan peristiwa yang terjadi pada saat itu. Hasil yang diperoleh segera disusun saat itu juga. Apa yang telah diamati pada dasarnya tidak lepas dari konteks lingkungan dimana tingkahlaku itu berlangsung.

2. Memiliki sifat deskriptif analitik: Data yang diperoleh dari hasil pengamatan, wawancara, dokumentasi, analisis, catatan lapangan, disusun peneliti di lokasi penelitian, bukan dalam bentuk angka-angka. Peneliti melakukan analisis data dengan memperbanyak in*form*asi, mencari hubungannya, membandingkan, dan menemukan hasil atas dasar data sebenarnya (bukan dalam bentuk angka). Hasil analisis data berupa pemaparan yang berkenaan dengan situasi yang diteliti dan disajikan dalam bentuk uraian narasi. Pemaparan data tersebut umumnya adalah menjawab dari pertanyaan dalam rumusan masalah yang ditetapkan.
3. Tekanan pada proses bukan hasil : Data dan in*form*asi yang dibutuhkan dalam penelitian kualitatif berkaitan dengan pertanyaan untuk mengungkapkan proses dan bukan hasil dari suatu kegiatan. Pertanyaan menuntut gambaran keadaan sebenarnya tentang kegiatan, tahap-tahap, prosedur, alasan-alasan dan interaksi yang terjadi dimana dan pada saat dimana proses itu berlangsung.
4. Bersifat induktif : Penelitian kualitatif diawali mulai dari lapangan yaitu fakta empiris. Peneliti terjun langsung ke lapangan, mempelajari suatu proses penemuan yang terjadi secara alami dengan mencatat, menganalisis dan melaporkan serta menarik kesimpulan dari proses berlangsungnya penelitian tersebut. Hasil temuan penelitian dari lapangan dalam bentuk konsep, prinsip, teori dikembangkan bukan dari teori yang telah ada. Penelitian kualitatif menggunakan proses induktif artinya dari data yang terpisah-pisah namun saling berkaitan erat.

5. Mengutamakan makna : Makna yang diungkapkan berkisar pada persepsi orang mengenai suatu peristiwa yang akan diteliti tersebut. Contoh: penelitian yang dilakukan tentang peran kepala sekolah dalam pembinaan guru. Peneliti memfokuskan perhatian pada pendapat kepala sekolah tentang guru yang dibinanya, mencari in*form*asi dan pandangan kepala sekolah tentang keberhasilan dan kegagalannya membina guru, apa saja yang dialami dalam membina guru, mengapa gurunya gagal dibina, dan kenapa hal itu terjadi. Selain mencari in*form*asi kepada kepala sekolah, peneliti mencari in*form*asi dari guru sebagai bahan perbandingan supaya dapat diperoleh pandangan mengenai mutu pembinaan yang dilakukan kepala sekolah. Ketepatan in*form*asi dari partisipan diungkap oleh peneliti agar dapat menginterpretasikan hasil penelitian secara tepat dan sahih.

Berdasarkan ciri-ciri tersebut dapat disimpulkan bahwa penelitian kualitatif dimulai dari lapangan yang berdasarkan pada lingkungan alami, bukan pada teori. Data dan in*form*asi yang diperoleh dari lapangan ditarik makna dan konsepnya, melalui pemaparan secara deskriptif analitik dan tanpa menggunakan angka, karena lebih mengutamakan prosesnya.

Dalam dunia pendidikan, penelitian kualitatif bertujuan untuk menggambarkan suatu proses kegiatan pendidikan yang didasarkan pada apa yang terjadi di lapangan sebagai bahan kajian untuk menemukan kelemahan dan kekurangannya sehingga dapat ditentukan upaya perbaikannya ;menganalisis suatu fakta, gejala dan peristiwa pendidikan yang terjadi di lapangan, menyusun hipotesis yang berkenaan dengan prinsip dan konsep pendidikan didasarkan pada data dan in*form*asi yang terjadi di lapangan. Dalam penelitian ini digunakan metode kualitatif dalam melakukan riset penelitian.