**BAB II**

**LANDASAN TEORI**

* 1. **Toko Alisha Mart**

Toko Alisha Mart merupakan usaha yang bergerak dibidang penjualan sembako atau barang kebutuhan sehari hari. Usaha ini sudah berdiri sejak September 2020 bermula warung sembako kecil kecilan dengan beralamatkan di Jl. Bunga Kamboja No. 93, RT 011/RW 005, Lahundape, Kec. Kendari Barat, Kota Kendari. Di Toko Alisha Mart yang didirikan oleh Ibu Widya Rani Furi, SE menjual berbagai kebutuhan sembako, kosmetik, dan obat-obatan.

Awal mula sebelum usaha Toko Alisha Mart berdiri Ibu Widya hanya membuka warung sembako kecil-kecilan. Kemudian seiring berjalannya waktu dengan modal yang apa adanya Ibu Widya mulai mendirikan Toko Alisha Mart dan hingga sampai saat ini terus semakin berkembang.

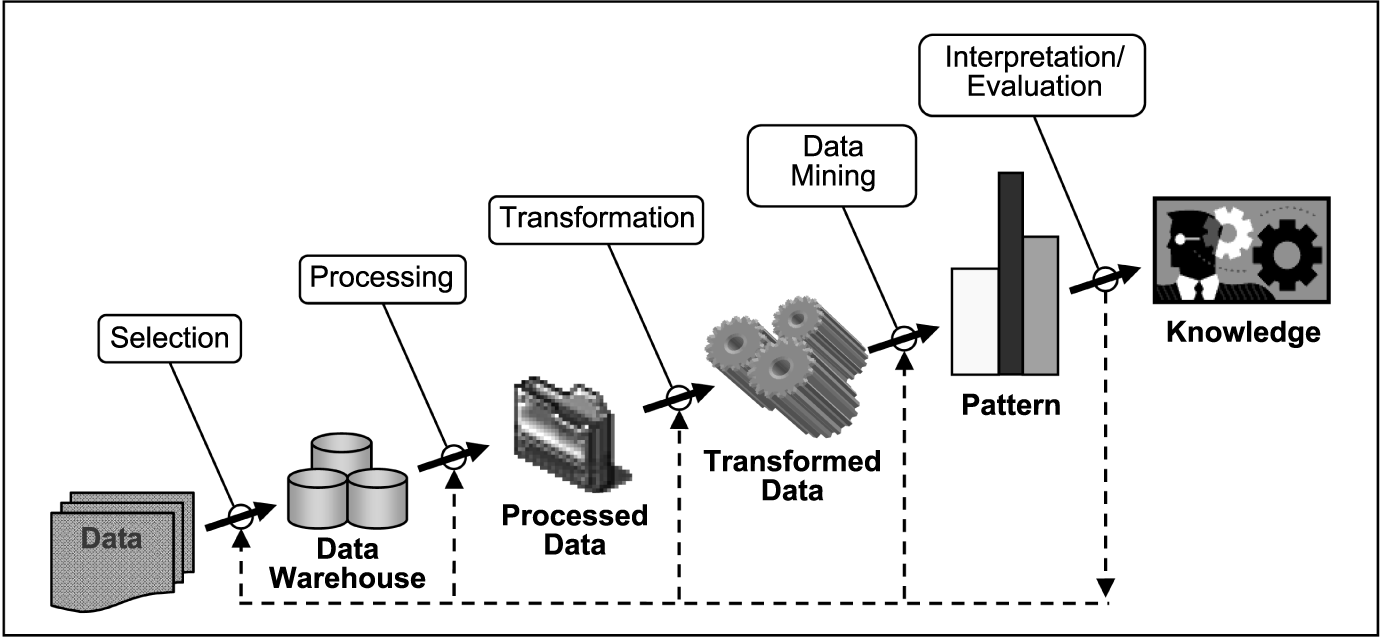
* 1. **Pengertian Prediksi atau Peramalan**

Prediksi adalah proses untuk meramalkan suatu variabel di masa mendatang dengan berdasarkan pertimbangan data pada masa lampau (Wedasari, 2015). Data yang sering digunakan untuk melakukan prediksi adalah data yang berupa data kuantitatif. Prediksi tidak harus memberikan jawaban secara pasti kejadian yang akan terjadi, melainkan berusaha untuk mencari jawaban sedekat mungkin yang akan terjadi.

* 1. ***Data Mining***

*Data Mining* adalah salah satu bidang yang berkembang pesat karena besarnya kebutuhan akan nilai tambah dari *database* dengan skala besar. *Data Mining* adalah serangkaian proses untuk menggali nilai tambah berupa pengetahuan yang selama ini tidak diketahui secara manual dari suatu kumpulan data. *Data Mining* memiliki hubungan dari bidang ilmu seperti *artificial intelligent*, *machine learning*, statistik dan *database*. Beberapa teknik *data mining* antara lain: *clustering, classification, association rule mining, neural network, genetic algorithm* dan lain-lain (Witten, Frank, & Hall, 2011).

Adapun proses *data mining* dapat dibagi menjadi beberapa tahap yang diilustrasikan pada Gambar 2.1 :



**Gambar 2.1 Tahapan Data Mining** (Witten et al., 2011)

* 1. **Metode *Association Rule***

*Association rule* adalah salah satu teknik utama atau prosedur dalam *Market Basket Analysis* untuk mencari hubungan antar *item* dalam suatu data *set* dan menampilkan dalam bentuk *association rule* (Widiati & Evita Dewi, 2014).

*Association rule* (aturan asosiatif) akan menemukan pola tertentu yang mengasosiasikan data yang satu dengan data yang lain. Untuk mencari *association* *rule* dari suatu kumpulan data, tahap pertama yang harus dilakukan adalah mencari *frequent itemset* terlebih dahulu. *Frequent itemset* adalah sekumpulan *item* yang sering muncul secara bersamaan. Setelah semua pola *frequent itemset* ditemukan, barulah mencari aturan asosiatif atau aturan keterkaitan yang memenuhi syarat yang telah ditentukan (Widiati & Evita Dewi, 2014) .

* 1. **Algoritma *Apriori***

Ide dasar dari algoritma ini adalah dengan mengembangkan *frequent* *itemset*. Dengan menggunakan satu *item* dan secara rekursif mengembangkan *frequent itemset* dengan dua *item*, tiga *item* dan seterusnya hingga *frequent itemset* dengan semua ukuran (MAULIDHA OKTAVIANI P, 2015)

Untuk mengembangkan *frequent set* dengan dua *item*, dapat menggunakan *frequent set item*. Alasannya adalah bila *set* satu *item* tidak melebihi *support minimum*, maka sembarang ukuran *itemset* yang lebih besar tidak akan melebihi *support* *minimum* tersebut. Secara umum, mengembangkan *set* dengan fc-*item* menggunakan *frequent* *set* dengan k – 1 *item* yang dikembangkan dalam langkah sebelumnya. Setiap langkah memerlukan sekali pemeriksaan ke seluruh isi *database*. Dalam asosiasi terdapat istilah *antecedent* dan *consequent*, *antecedent* untuk mewakili bagian “jika” dan *consequent* untuk mewakili bagian “maka”. Dalam analisis ini, *antecedent* dan *consequent* adalah sekelompok *item* yang tidak punya hubungan secara bersama. Dari jumlah besar aturan yang mungkin dikembangkan, perlu memiliki aturan - aturan yang cukup kuat tingkat ketergantungan antar *item* dalam *antecedent* dan *consequent.* Untuk mengukur kekuatan aturan asosiasi ini, digunakan ukuran *support* dan *confidence.* *Support* adalah rasio antara jumlah transaksi yang memuat *antecedent* dan *consequent* dengan jumlah transaksi. *Confidence* adalah rasio antara jumlah transaksi yang meliputi semua *item* dalam *antecedent* dan *consequent* dengan jumlah transaksi yang meliputi semua *item* dalam *antecedent.*

Algoritma *Apriori* adalah salah satu algoritma yang melakukan pencarian *frequent itemset* dengan menggunakan teknik *Association Rule* (MAULIDHA OKTAVIANI P, 2015). Algoritma *Apriori* menggunakan pengetahuan frekuensi atribut yang telah diketahui sebelumnya untuk memproses informasi selanjutnya. Pada algoritma *Apriori* menentukan kandidat yang mungkin muncul dengan cara memperhatikan *minimum* *support* dan *minimum* *confidence*. *Support* adalah nilai pengunjung atau persentase kombinasi sebuah *item* dalam *database*. Rumus *support* adalah sebagai berikut :

(2.1)

Sedangkan nilai *support* dari dua *item* diperoleh dari rumus persamaan berikut :

(2.2)

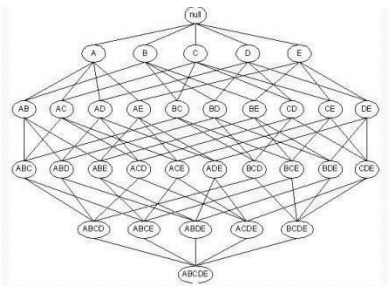
Sedangkan *confidence* adalah nilai kepercayaan yaitu kuatnya hubungan antar *item* dalam sebuah *Apriori. Confidence* dapat dicari setelah pola frekuensi munculnya sebuah *item* ditemukan. Untuk mencari nilai *confidence* dapat menggunakan persamaan berikut :

(2.3)

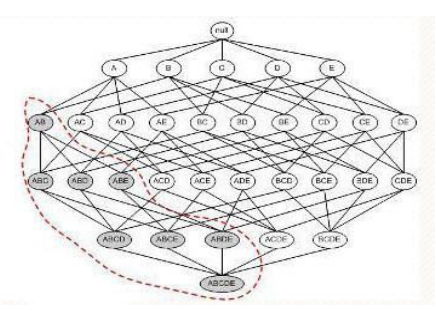
Secara garis besar cara kerja algoritma *apriori* adalah:

* 1. Pembentukan kandidat *itemset,* Kandidat *k-itemset* dibentuk dari kombinasi (k-1)- *itemset* yang didapat dari iterasi sebelumnya. Satu ciri dari algoritma *Apriori* adalah adanya pemangkasan kandidat *k-itemset* yang subset-nya yang berisi k-1 *item* tidak termasuk dalam pola frekuensi tinggi dengan panjang k-1.
  2. Penghitungan *support* dari tiap kandidat *k-itemset. Support* dari tiap kandidat *k-itemset* didapat dengan men-scan *database* untuk menghitung jumlah transaksi yang memuat semua *item* di dalam kandidat *k-itemset* tsb. Ini adalah juga ciri dari algoritma *Apriori* dimana diperlukan penghitungan dengan scan seluruh *database* sebanyak *k-itemset* terpanjang.
  3. Tetapkan pola frekuensi tinggi. Pola frekuensi tinggi yang memuat *k-item* atau *k-itemset* ditetapkan dari kandidat *k-itemset* yang *support*-nya lebih besar dari *minimum* *support.*
  4. Bila tidak didapat pola frekuensi tinggi baru maka seluruh proses dihentikan. Bila tidak, maka k ditambah satu dan kembali ke bagian 1.

Kelebihan dari algoritma *Apriori* ini adalah lebih sederhana dan dapat menangani data yang besar. Sedangkan algoritma lainnya memiliki kelemahan dalam penggunaan memori saat jumlah data besar, tentunya berpengaruh terhadap banyaknya *item* yang diproses serta mudah di pahami struktur kerja dan implementasinya (MAULIDHA OKTAVIANI P, 2015).

Masalah utama pencarian *Frequent Itemset* adalah banyaknya jumlah kombinasi *itemset* yang harus diperiksa apakah memenuhi minimum *support* atau tidak. Salah satu cara untuk mengatasinya adalah dengan mengurangi jumlah kandidat *itemset* yang harus diperiksa. *Apriori* adalah salah satu pendekatan yang sering digunakan pada *Frequent Itemset Mining*. Prinsip Apriori adalah jika sebuah *itemset infrequent,* maka *itemset* yang *infrequent* tidak perlu lagi diexplore supersetnya sehingga jumlah kandidat yang harus diperiksa menjadi berkurang. Ilustrasinya seperti pada Gambar 2.2 :

**Gambar 2.2 Contoh *Hash Tree* (Herawati, 2013)**

Pada gambar di atas, pencarian *Frequent Itemset* dilakukan tanpa menggunakan prinsip *Apriori*. Dengan menggunakan prinsip *Apriori,* pencarian *Frequent Itemset* akan menjadi seperti pada Gambar 2.3:

**Gambar 2.3 Contoh *Hash Tree* (Herawati, 2013)**

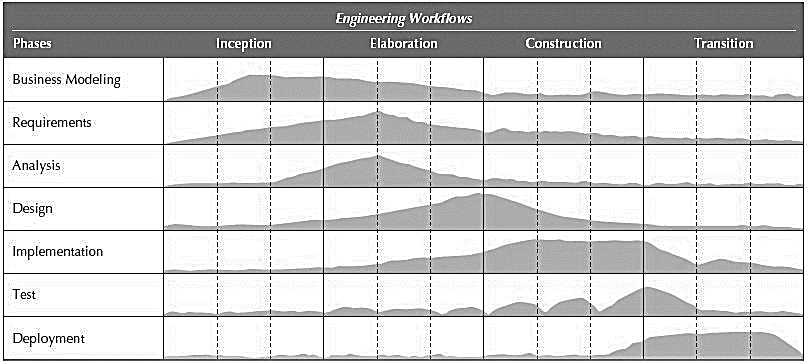
* 1. **Perhitungan Akurasi**

Perhitungan akurasi digunakan untuk mengetahui sejauh mana sistem dapat menguji apakah metode yang digunakan sudah cukup baik. Perhitungan akurasi pada sistem identifikasi daun jeruk siam berpenyakit menggunakan metode M-SVM menggunakan rumus pada Persamaan 2.9 (Achban, 2017).

Keakuratan (%) = x 100% (2.4)

* 1. **Model Pengembangan Sistem**

Model Pengembangan Sistem yang digunakan pada penelitian ini adalah RUP *(Rational Unified Process)* yang merupakan pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang *(iterative)*, fokus pada arsitektur *(architecturecentric)*, lebih diarahkan berdasarkan penggunaan kasus *(use case driven).* RUP merupakan proses rekayasa perangkat lunak dengan pendefinisian yang baik *(well defined)* dan penstrukturan yang baik *(well structured)* (Rossa, A S dan M. Shalahuddin.2013)



**Gambar 2.4 Pemetaan dalam Fase RUP**

RUP memiliki empat buah tahap atau fase yang dapat dilakukan pula secara iteratif.Berikut adalah penjelasan untuk setiap fase pada RUP.

1. *Inception*

Tahap ini yaitu tahap penentuan ruang lingkup pengembangan sistem dari hasil wawancara dan observasi yang dilakukan, meliputi hasil penelitian atau skripsi-skripsi terdahulu.

1. *Elaboration*

Tahap ini yaitu tahap melakukan identifikasi masalah pada sistem yang akan dibuat berdasarkan hasil observasi dan wawancara yang telah dilakukan. Didalam *elaboration* terdapat dua tahapan yaitu:

1. Analisis

Terdapat tiga fase dalam tahapan analisis sistem pada alur pengembangan sistem RUP, yaitu: analisis permasalahan, analisis persyaratan, dan analisis keputusan.

1. Perancangan

Tahap ini terdiri dari: perancangan aplikasi dengan menggunakan diagram UML meliputi *use case diagram*, *activity diagram* dan *sequence diagram,* dan perancangan tampilan.

1. *Construction*

Tahap ini menjelaskan bagaimana mengimplementasi dan melakukan uji coba terhadap aplikasi yang telah dibuat. Dalam tahapan implementasi dijelaskan perangkat keras dan perangkat lunak apa saja yang dibutuhkan untuk pengimplementasian sistem. Sedangkan pada tahapan uji coba dilakukan *testing*. Testing diperlukan untuk menjamin kualitas aplikasi yang telah dibuat apakah telah sesuai dengan yang diharapkan.

1. *Transition*

Pada tahap *transition* yaitu pembuatan panduan penggunaan dari sistem yang telah dibuat.

* 1. ***Flowchart***

*Flowchart* adalah simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang saling terhubung sehingga setiap simbol *flowchart* melambangkan pekerjaan dan instruksinya (*Ewolf*, 2011). Simbol–simbol yang digunakan dalam *flowchart* adalah sebagai berikut:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabel 2.1 Simbol-simbol *flowchart*** | | | | |
| **NO.** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | Proses | Mempresentasikan operasi. |
| 2. |  | *Input / Output* | Mempresentasikan *Input* atau *Output* data yang diproses atau informasi. |
| 3. |  | Keputusan | Keputusan dalam program. |
| 4. |  | Dokumen | Dokument I / O dalam format cetak. |
| 5. |  | *Terminal points* | Awal / akhir *flowchart*. |
| 6. |  | *Preparation* | Pemberian harga awal. |
| 7. |  | Manual *input* | *Input* yang dimasukkan secara manual dari *keyboard*. |
| 8. |  | Penghubung | Keluar atau masuk dari bagian lain *flowchart* khususnya |
| 9. |  | Penghubung | Keluar atau masuknya dari bagian lain *flowchart*  khususnya halaman lain. |
| 10. |  | *Display* | *Output* yang ditampilkan pada terminal |
| 11. |  | Anak panah | Mempresentasikan alur kerja. |

Sumber: Ewolf, 2011.

* 1. ***Unified Modeling Language* (UML)**
     1. **Pengertian UML**

*Unified Modeling Language* (UML) merupakan pengganti dari metode analisis berorientasi objek dan desain berorientasi objek (OOA&D) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an. UML merupakan gabungan dari metode Booch, Rumbaugh (OMT) dan Jacobson. Tetapi UML ini akan mencakup lebih luas daripada OOA&D. Pada pertengahan pengembangan UML dilakukan standarisasi proses dengan OMG (*Object Management Group*) dengan harapan UML akan menjadi bahasa standar pemodelan pada masa yang akan datang.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. Jika anda ingin berdiskusi tentang desain dengan seseorang, maka anda hanya membutuhkan bahasa pemodelan bukan proses yang digunakan untuk mendapatkan desain. UML merupakan bahasa standar untuk penulisan *blue print software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak .

* + 1. **Diagram Dalam UML**

Dalam UML terdapat diagram-diagram yang bisa menggambarkan bagian atau aspek tertentu dari sebuah sistem. Sehingga dapat terlihat jelas alur dan gambaran umum dari perangkat lunak yang dibangun. Ada beberapa jenis diagramdalam UML yaitu (Sukamto & Shalahuddin, 2013):

* + - 1. *Usecase* *Diagram*

Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat *user*) dan bukan bagaimana fungsi yang ada di dalam system (Sukamto & Shalahuddin, 2013).

**Tabel 2.2 Simbol *Use Case* Diagram**

|  |  |  |
| --- | --- | --- |
| **No.** | **Simbol** | **Keterangan** |
| 1. |  | Aktor  Menunjukkan *user* yang akan menggunakan system |
| 2. |  | *Usecase*  Menunjukkan proses yang terjadi pada system |
| 3. |  | *Association*  Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 4. |  | *Generalization*  Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*) |
| 5. |  | *Dependency*  Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*Independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
| 6. |  | *Include*  Menspesifikasikan bahwa *usecase* sumber secara eksplisit. |
| 7. |  | *Extend*  Menspesifikasikan bahwa *usecase* target memperluas perilaku dari *usecase* sumber pada suatu titik yang diberikan |
| 8. |  | *System*  Menspesifikasikan paket yang menampilkan sistem secara terbatas |
| 9. |  | *Collaboration*  Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku-perilaku yang lebih besar dari jumlah dan elemen-elemennya (energi) |
| 10. |  | *Note*  Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

* + - 1. *Class* Diagram

Menggambarkan struktur statis *class* di dalam sistem. *Clas*s merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara : *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram* (Sukamto & Shalahuddin, 2013).

Dalam rekayasa perangkat lunak, diagram kelas didalam *Unified Modeling Language* (UML) adalah jenis diagram struktur statis yang menggambarkan struktur dari suatu sistem dengan menunjukkan kelas sistem, atribut, operasi atau metode, dan hubungan antara kelas.

Sebuah kelas digambarkan sebagai kotak dengan tiga level yakni:

1. Pada bagian atas judul dicetak tebal dengan *text* rata tengah dengan huruf pertama kapital.
2. Pada bagian tengah atribut dicetak rata kiri dengan hurup pertama non kapital dilengkapi dengan tipe data.
3. Pada bagian bawah adalah metode yang digunakan dalam kelas diakhiri dengan kurung kurawal.

lambang (+/-/#) pada kelas menggambarkan aksesibilitas dari atribut atau metode;

1. (+) bermakna *public* sehingga atribut/metode dapat diakses secara langsung dari luar kelas.
2. (-) bermakna *private* dimana kelas/atribut hanya bisa diakses oleh metode didalam kelas itu sendiri dan
3. (#) bermakna *protected* dimana kelas/atribut hanya bisa diakses oleh kelas yang berada dalam *package* yang sama atau *subclass* dari kelas itu sendiri.

**Tabel 2.3 Simbol *Class* Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1 |  | *Generalization* | Hubungan dimana objek anak  *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 2 |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 3 |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor |
| 4 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 5 |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)*. |

* + - 1. *Sequence* Diagram

Menggambarkan kolaborasi dinamis antara sejumlah *object*. Kegunaanya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksiantara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi system (Sukamto & Shalahuddin, 2013)

**Tabel 2.4 Simbol *Sequence* Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SIMBOL** | **NAMA SIMBOL** | **KETERANGAN** |
| 1. |  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |
| **Tabel 2.5 Simbol *Sequence* Diagram (Lanjutan)** | | | |
| **NO** | **SIMBOL** | **NAMA SIMBOL** | **KETERANGAN** |
| 2. |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. |
| 3. |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. |
| 4. |  | *Message* | Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*. |

* + - 1. *Activity* Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi (Sukamto & Shalahuddin, 2013).

**Tabel 2.6 Simbol *Activity* Diagram**

|  |  |  |
| --- | --- | --- |
| No. | Simbol | Keterangan |
| 1. | Start | Kondisi Awal  Menunjukkan awal dari suatu diagram aktivitas |
| 2. | End | Kondisi Akhir  Menunjukkan akhir dari suatu diagram aktivitas |
| 3. |  | Kondisi transisi  Menunjukkan kondisi transisi antar aktivitas |
| **Tabel 2.7 Simbol *Activity* Diagram (Lanjutan)** | | |
| No. | Simbol | Keterangan |
| 4. |  | Swimlane  Menunjukkan aktor dari diagram aktivitas yang dibuat |
| 5. |  | Aktivitas  Menunjukkan aktivitas-aktivitas yang terdapat pada diagram aktivitas |
| 6. |  | Pengecekan kondisi  Menunjukkan pengecekan terhadap suatu kondisi |

* 1. **XAMPP**

XAMPP adalah suatu bundel *web server* yang populer digunakan untuk coba-coba di *Windows* karena kemudahan instalasinya. Program *open source* tersebut berisi antara lain server web Apache, interpreter PHP, dan basis data MySQL. Setelah menginstall XAMPP, kita bisa memulai pemrograman PHP di komputer sendiri maupun mencoba menginstall aplikasi-aplikasi web (Nugroho, 2008).

*XAMPP* adalah perangkat yang menggabungkan tiga aplikasi ke dalam satu paket, yaitu *apache, MySQL, dan PHPMyAdmin*. Dengan *XAMPP* pekerjaan anda akan sangat dimudahkan, karena dapat menginstalasi dan menkofigurasi ketiga aplikasi tadi secara sekaligus atau otomatis. *XAMPP* telah mengalami perkembangan dari waktu ke waktu. Versi yang terbaru adalah revisi dari edisi sebelumnya, sehingga lebih baik dan lebih lengkap. Aplikasi utama dalam paket *XAMPP* setidaknya terdiri atas web server.

* 1. ***Database* MySQL**
     1. ***Database***

*Database* atau sering disebut basis data adalah sekumpulan informasi yang disimpan dalam komputer secara sistematik dan merupakan sumber informasi yang dapat diperikasa menggunakan suatu program komputer. Database berfungsi untuk menyimpan informasi atau data. Untuk mengelola database diperlukan software yang sering disebut dengan DBMS (*Database Management System*).

Dengan DBMS pengguna atau user dapat membuat, mengelola, mengontrol, dan mengakses database dengan mudah, praktis dan efesien. Database terdiri dari tabel yang didalamnya terdapat field-field, dan sebuah database terdiri dari beberapa tabel. Dalam pembuatan database, Anda perhatikan hal-hal berikut (Nugroho, 2004) :

1. Setiap tabel dalam database, harus memiliki field (kolom) yang unik yang disebut dengan *Primary Key.*
2. Tabel dalam database tidak boleh ada *redudancy* data yang mengandung *record* ganda. Jika terdapat data yang sama, maka perlu dilihat kembali rancangan tabelnya.
3. Pilih type data yang tepat, sehingga ukuran database seminimal mungkin.
   * 1. **MySQL**

*MySQL* adalah aplikasi *SQL database server* yang multi *user*. Oleh karena itu, *MySQL* digunakan dalam pembuatan aplikasi ini yang digunakan sebagai *database server* untuk menyimpan data lokasi yang dikirim oleh masing-masing telepon seluler. *MySQL* adalah aplikasi *database* yang mudah digunakan karena telahbanyak digunakan di berbagai aplikasi dan *websites.* Pada *Web Hosting* juga sudah banyak yang menyediakan *database* ini.

Dalam *MYSQL* ada beberapa operasi dasar yang lebih dikenal dengan operasi CRUD yaitu (Nugroho, 2008):

1. *Create*

*Create* adalah operasi penambahan data baru ke dalam tabel. Terdapat 2 *Query* untuk menambah data, yang pertama adalah penambahan yang tidak menspesifikasikan nama kolom yang akan ditambahkan dan hanya memberikan isi dari tabelnya. *Query* nya adalah sebagai berikut:

INSERT INTO*table\_name*

VALUES (*value1*,*value2*,*value3*,...);

Untuk yang kedua adalah menginputkan data dengan mencantumkan nama kolom yang akan diisikan dengan isi di dalamnya. *Query*nya adalah sebagai berikut:

INSERT INTO *table\_name* (*column1*,*column2*,*column3*,...)

VALUES (*value1*,*value2*,*value3*,...);

1. *Read*

*Read* adalah operasi untuk menampilkan semua atau sebagian data yang berada di dalam *database*. *Query*nya adalah sebagai berikut:

SELECT *column\_name*,*column\_name*

FROM*table\_name*;

1. *Update*

*Update* adalah operasi untuk mengubah data yang ada di dalam *database*. *Query*nya adalah sebagai berikut:

UPDATE *table\_name*

SET*column1*=*value1*,*column2*=*value2*,...

WHERE*some\_column*=*some\_value*;

1. *Delete*

*Delete* adalah operasi untuk menghapus data yang ada di dalam table. *Query*nya adalah sebagai berikut:

DELETE FROM *table\_name*

WHERE*some\_column*=*some\_value*;

**Tabel 2.8 Tipe *Database* MySQL**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Type** | **Keterangan** | |
| 1. | *Varchar* | Tipe data karakter yang panjangnya tidak tetap | |
| 2. | TINYINT | Adalah tipe data bilangan bulat yang rentangnya -128 sampai 127 atau menggunakan atribut unsigned dari 0 hingga 255 | |
| 3*.* | *Text* | Tipe data yang dapat menampung semua tipe data | |
| 4*.* | *Date* | Tipe data yang digunakan untuk mendiskripsikan tanggal | |
| **Tabel 2.9 Tipe *Database* MySQL (Lanjutan)** | | | |
| **No.** | **Type** | **Keterangan** | |
| 5*.* | *Smallint* | Adalah tipe data bilangan bulat yang rentangnya –32768 sampai 32767. *The unsigned range is* 0 to 65535 | |
| 6*.* | *Mediumint* | Adalah tipe data bilangan bulat yang rentangnya –8388608 to 8388607.*unsigned range*-nya 0 sampai 16777215 | |
| 7*.* | *Int* | Tipe data yang bernilai integer/bilangan bulat | |
| 8*.* | *Time* | Tipe data waktu. Jangkauannya adalah '-838:59:59' hingga '838:59:59'. MySQL menampilkan TIME dalam format 'HH:MM:SS' | |
| 9*.* | *Char* | | Tipe data untuk menampung data yang bertipe karakter |
| 10. | *Primary Key* | | Kunci primer adalah suatu atribut atau satu set minimal atribut yang tidak hanya mendefinisikan secara unik suatu kejadian spesifik tetapi juga dapat mewakili setiap kejadian dari suatu kejadian |

* 1. **PHP**
     1. **Pengertian PHP**

PHP Kepanjangan dari PHP adalah "*Hypertext Preprocessor*" (ini merupakan singkatan rekursif). PHP adalah bahasa *scripting* web *HTML-embedded*. Ini berarti kode PHP dapat disisipkan ke dalam HTML halaman Web. Ketika sebuah halaman PHP diakses, kode PHP dibaca atau"diurai" oleh *server*. *Output* dari fungsi PHP pada halaman biasanya dikembalikan sebagai kode HTML, yang dapat dibaca oleh *browser*. Karena kode PHP diubah menjadi HTML sebelum halaman dibuka, pengguna tidak dapat melihat kode PHP pada halaman. Ini membuat halaman PHP cukup aman untuk mengakses *database* dan informasi aman lainnya (Nugroho, 2004).

Banyak *sintaks* PHP yang hasil adaptasi dari bahasa lain seperti bahasa C, Java dan Perl. Namun, PHP memiliki sejumlah fitur unik dan fungsi tertentu juga. Tujuan dari bahasa pemrograman PHP adalah untuk memungkinkan pengemban web untuk menulis halaman yang dihasilkan secara dinamis dengan cepat dan mudah. PHP juga bagus untuk menciptakan situs Web *database-driven*.

* + 1. **Kelebihan PHP dari Bahasa Pemrograman Lain**

Beberapa kelebihan PHP dari bahasa pemrograman web, antara lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukansebuah kompilasi dalam penggunaanya.
2. *Web Server* yang mendukung PHP dapat ditemukan dimana - mana dari mulai *apache*, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa *script* yang paling mudah karena memiliki referensi yang banyak.
5. PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.
   1. **HTML**

HTML adalah *Hyper Text Markup Language* yang merupakan sebuah bahasa *scripting* berguna untuk menuliskan halaman *web*. Pada *web*, HTML dijadikan sebagai Bahasa *Script* dasar yang berjalan bersama berbagai bahasa *scripting* pemrograman lainnya. Semua *tag*s HTML bersifat dinamis artinya kode HTML tidak dapat dijadikan sebagai *file executable* program. Hal tersebut disebabkan, HTML hanyalah sebuah bahasa *scripting* yang dapat berjalan apabila dijalankan di dalam *browser* (pengakses web). *Browser* yang mendukung HTML antara lain *Internet Explorer, Netscape Navigator,* Operasi, *Mozilla*, dan lain-lain (Harison & Syarif, 2016).