**BAB II**

**LANDASAN TEORI**

* 1. **Aplikasi**

**Aplikasi berasal dari kata *Aplication* yang menurut kamus Komputer Eksekutif (1993:9), *Aplication* adalah masalah yang memakai teknik pemrosesan data aplikasi biasanya mengacu pada komputasi yang diinginkan, atau pemrosesan data. Pengertian aplikasi menurut Yan Tirtobisono (1992:21) adalah istilah yang digunakan untuk pengguna komputer bagi pemecahan masalah. Sedangkan menurut Kamus Besar Bahasa Indonesia (1998:52), “aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu.**

**Aplikasi adalah suatu program komputer yang dibuat untuk mengejakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan program siap pakai yang dapat digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut.**

* 1. **Manajemen Artefak Pembangun Perangkat Lunak**

Manajemen telah banyak disebut sebagai seni “seni untuk merealisasikan pekerjaan melalui orang lain”. Terjemahan bebasnya kira-kira seperti ini: seorang yang lain mancapai tujuan tertentu, namun untuk mencapai tujuan tersebut dia menggunakan “tangan” orang lain. Dalam kaitannya dengan organisasi, bisa berari bahwa para manajer dalam mencapai tujuan organisasi dengan cara peraturan orang lain ( misalkan staf atau karyawn) untuk melakukan berbagai pekerjaan sesuai dengan tujuan organisasi, tanpa harus melakukan dengan “tangan” sendiri. Manajemen lebih bersifat arsitektur atau konseptual (perancangan-perancangan strategis), bukan tehnikal maupun operasional. Manajemen mencakup fungsi perencanaan ( Penetapan apa yang tidak akan dilakukan ), penorganisasian ( perancangan dan penugasan kelompok kerja ), penyusunan personalia ( Penarikan, seleksi, pengembangan dan penilaian prestasi kerja), pengarahan ( motivasi, kepemimpinan, integritas dan pengolahan konflik) dan pengawasan.

Manajemen Proyek merupakan suatu ilmu pengetahuan, keahlian dan juga keterampilan, cara teknis yang terbaik serta dengan sumber daya yang terbatas untuk mencapai sasaran atau tujuan yang sudah ditentukan agar mendapat hasil yang optimal dalam hal kinerja, waktu dan keselamatan kerja. Tujuan utama dari manajemen proyek adalah agar proyek dapat dilaksanakan dengan efisien, tepat waktu, dan mencapai hasil yang diinginkan. Sering terjadi pada sebuah proyek yang berlarut pada pekerjaannya sehingga pada akhirnya harus mengalami penjadwalan ulang. Oleh karena itu, peran perencanaan dalam suatu proyek sangat penting, segala sesuatu harus dimulai dari rencana dan harus disepakati bersama antara para *stakeholder* yang terlibat pada proyek. *Stakeholder* yang dimaksud didalam proyek adalah pemilik proyek (*project owner*), komite pengarah (*steering committee*), pengguna hasil proyek dan pelaksana proyek (Heryanto, 2015).

Artefak merupakan salah satu dari banyak jenis produk sampingan nyata yang dihasilkan selama pengembangan perangkat lunak. beberapa artefak (misalnya, kasus penggunaan, diagram kelas, dan model Unified Modelling Language ( UML ) Lainnya, persyaratan dan dokumen desain) membantu menjelaskan fungsi, arsitektur, dan desain perangkat lunak. artefak lain berkaitan dengan proses pengembangan itu sendiri seperti rancangan proyek, kasus bisnis, dan penilaian risiko. Namun dalam lingkungan pengembangan perangkat lunak cepat terdapat berbagai macam artefak yang terdapat dalam satu dokumentasi, hal ini mencerminkan fakta bahwa dalam model pengembangan perangkat lunak cepat setiap artefak dapat digunakan sebagai dokumentasi.

Perangkat lunak (*Software*) adalah program komputer yang terasosiasi dengan dokumentasi preangkat lunak seperti dokementasi kebutuhan, model sistem, dan cara penggunaannya. Rekayasa perangkat lunak (RPL) merupakan pembangunan sebuah perangkat lunak dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomis yang di percaya dan bekerja secara efisien menggunakan mesin. RPL lebih fokus pada praktek pengembangan perangkat lunak yang bermanfaat bagi pelanggan (*User*) dengan memenuhi kriteria sebagai berikut :

1. Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring  
   berkembangnya teknologi dan lingkungan (*Maintainability*).
2. Dapat diandalkan dengan proses bisnis yang dijalankan dan perubahan yang terjadi (*Dependability dan Robust*).
3. Efisiensi dari segi sumber daya dan penggunaan.
4. Kemampuan untuk dipakai sesuai dengan kebutuhan (*Usability*)  
   Jadi perangkat lunak yang baik adalah perangkat lunak yang fokus kepada pengguna atau pelanggan.
   1. ***Text Mining***

*Text mining* merupakan salah satu bidang khusus dari data *mining*. *Text mining* dapat didefinisikan sebagai suatu proses menggali informasi dimana seseorang user berinteraksi dengan sekumpulan dokumen menggunakan tool analisis yang merupakan komponen-komponen dalam data mining (Wisnu & Hetami, 2015). Text mining digunakan untuk mengolah dokumen sebelum dilakukan proses similarity. Didalam proses *text mining* terdapat proses *preprocessing* (Asshidiq & M.E, 2013). *Preprocessing text* merupakan tindakan menghilangkan karakter-karakter tertentu yang terkandung dalam dokumen, seperti koma, tanda petik dan lain-lain serta mengubah semua huruf kapital menjadi huruf kecil. Selain itu, dalam tahap text *preprocessing* ini dilakukan *tokenization*. *Tokenization* merupakan proses pengolahan token yang terdapat dalam rangkaian teks, sehingga dokumen akan dipecah-pecah menjadi *term* (Aziz, et al., 2015). *Text mining* dalam prakteknya mencari pola-pola tertentu, mengasosiasikan suatu bagian teks dengan yang lain berdasarkan aturan-aturan tertentu, kata-kata yang dapat mewakili sehingga dapat dilakukan analisa keterhubungan antar satu dengan yang lain (Wisnu & Hetami, 2015). Berikut tahapan-tahapan proses didalam *text mining*:

1. *Casefolding*: mengubah semua huruf dalam dokumen menjadi huruf kecil (*lowercase*).
2. *Tokenizing*: memotong tiap kata dalam kalimat atau parsing dengan menggunakan spasi sebagai delimiter yang akan menghasilkan token berupa kata.
3. *Filtering*: menyaring kata yang didapat dari *tokenizing* yang dianggap tidak penting atau tidak memiliki makna dalam proses *text mining* yang disebut *stopword*. *Stopword* berisi kata kata umum yang sering muncul dalam sebuah dokumen dalam jumlah banyak namun tidak memiliki kaitan dengan tema tertentu.
4. *Stemming*: mengembalikan kata-kata yang diperoleh dari hasil *filtering* ke bentuk dasarnya, menghilangkan imbuhan awal (*prefix*) dan imbuhan akhir (*sufix*) sehingga didapat kata dasar.
   1. ***Term Frequency-Inversed Document Frequency* (TF/IDF)**

Basis pembobotan TF-IDF merupakan jenis pembobotan yang melibatkan pengukuran statistik untuk mengukur seberapa penting sebuah kata dalam kumpulan dokumen. Tingkat kepentingan meningkat ketika sebuah kata muncul beberapa kali dalam sebuah dokumen tetapi diimbangi dengan frekuensi kemunculan kata tersebut dalam kumpulan dokumen (Wisnu & Hetami, 2015). TF merupakan pembobotan yang sederhana dimana penting tidaknya sebuah kata diasumsikan sebanding dengan jumlah kemunculan kata tersebut dalam dokumen, sementara IDF merupakan pembobotan yang mengukur seberapa penting sebuah kata dalam dokumen apabila dilihat secara global pada seluruh dokumen (M. Isa & Abidin, 2013). Perhitungan IDF menggunakan persamaan 1.

*IDF(t) = log (D/df(t))* (1)

Dimana:  
df(t) = Jumlah dokumen yang mengandung kata ke-t dari kata kunci

D = Jumlah semua dokumen yang ada di dalam database

IDF = Rasio frekuensi dokumen pada kata ke-t dari kata kunci Perhitungan TF-IDF menggunakan persamaan 2

*TF-IDF(d,t) = TF(d,t) \* IDF(t)* (2)

Dimana:  
d = dokumen ke-d

t = kata ke-t dari kata kunci

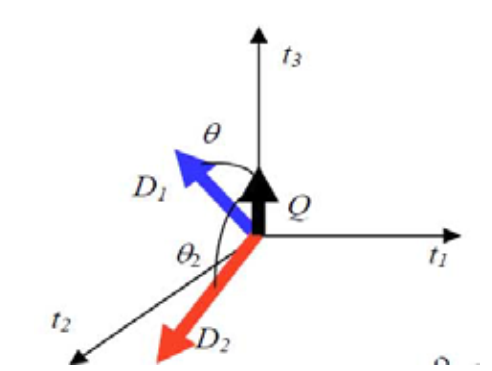
tf = frekuensi banyaknya kata ke-t dari kata kunci pada dpkumen ke-d

TF-IDF = bobot dokumen ke-d terhadap kata kunci ke-t

IDF = rasio frekunsi dokumen pada kata ke-t dari kata kunci

* 1. ***Vector Space Model***

*Vector Space Model* (VSM) sering digunakan untuk mempresentasikan sebuah dokumen dalam ruang vector (P.D & Patrick, 2010). VSM merupakan model Information Retrieval yang mempresentasikan dokumen dan query sebagai vektor pada ruang multidimensi. Kesamaan suatu dokumen dengan query dapat diukur dengan vektor dokumen dan vektor query (Aziz, et al., 2015).



**Gambar 2.1** Representasi Dokumen dan *Query* pada VSM (Mandala & Setiawan, 2002).

Perhitungan kemiripan antara vektor dokumen dan vektor *query* dilihat dari sudut yang paling kecil. Sudut yang dibentuk oleh dua buah vektor dapat dihitung dengan melakukan *inner product* (ER, et al., 2012) .Kemiripan antara vektor dokumen dan vektor *query* akan dihitung dengan pendekatan *cosine similarity* (Aziz, et al., 2015). Pengukuran *Cosine Similarity* menggunakan persamaan 3.

*Sim (di,dj) = Di Dj = Wiq.Wij  
||D1|| ||D2|| √Wiq² \*√Wij²*

Dimana:  
Sim (di, dj) = similaritas antara query dan dokumen  
||D1|| = panjang vektor dokumen 1

||D2|| = panjang vektor dokumen 2  
Wij = bobot term dalam dokumen  
Wiq = bobot query dalam dokumen

*Vector Space Model* (VSM), metode ini merupakan metode yang digunakan untuk mengukur tingkat kemiripan pada setiap Artefak yang dimasukan oleh pengguna dan membandingkan dengan Artefak yang sudah tersimpan didalam database.

* 1. **Scrum**

Scrum sebuah kerangka kerja di mana orang-orang dapat menyelesaikan permasalahan kompleks yang senantiasa berubah, di mana pada saat bersamaan menghasilkan produk dengan nilai setinggi mungkin secara kreatif dan produktif.

Scrum Sifat :

* Ringan
* Mudah dipahami
* Sulit dikuasai

Scrum adalah kerangka kerja proses yang telah digunakan untuk mengelola pengembangan produk kompleks semenjak awal 1990-an. Scrum bukanlah sebuah proses ataupun teknik untuk mengembangkan produk; daripada itu, ini adalah sebuah kerangka kerja di mana di dalamnya anda dapat memasukkan beragam proses dan teknik. scrum akan mengekspos pererakan efektufutas manajemen produk dan praktik pengembangan yang sedan anda jalani, dengan begitu anda dapat melakukan peningkatan.

Kerangka kerja Scrum terdiri dari tim Scrum, serta peran-peran mereka di dalamamnya, acara-acara, artefak-artefak, dan aturan-aturan. setiap komponen di dalam kerangka kerja memiliki maksud tertentu dan peran penting demi keberhasilan pengguna scrum.

* 1. **Waterfall**

Model Waterfall merupakan salah satu model pengembangan perangkat lunak yang ada di dalam model SDLC (*Sequencial Developmen Life Cycle*). menurut Sukanto dan Shalahuddin (2013:26) mengemukakan bahwa " SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya, berdasarkan *best practice* atau cara-cara yang sudah teruji baik.

Sedangkan Sukanto dan Shalahuddin (2013:28) di jelaskan bahwa model *waterfall*  sering juga di sebut model sekuensi linear atau alur hidup klasik. pengembangan sistem dikerjakan secara terurut mulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung

* 1. **Unified Modelling Language (UML)**

Menurut Windu Gata (Urva, dkk, 2015) hasil pemodelan pada OOAD terdokumentasikan dalam bentuk Unified Modeling Language (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umumdalam industri perangkat lunak dan pengembangan sistem.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

Use case Diagram

Use case diagram merupakan pemodelan untuk kelakukan (behavior) 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 di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbolsimbol yang digunakan dalam use case diagram, yaitu

**Tabel 2.1** Simbol usecase

|  |  |
| --- | --- |
| **Gambar** | **Keterangan** |
|  | Use case menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama use case. |
|  | Aktor adalah abstraction dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan use case, tetapi tidak memiliki control terhadap use case. |
|  | Asosiasi antara aktor dan use case, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data |
|  | Asosiasi antara aktor dan use case yang menggunakan panah terbuka untuk mengidinkasikan bila actor berinteraksi secara pasif dengan sistem. |
|  | Include, merupakan di dalam use case lain (required) atau pemanggilan use case oleh use case lain, contohnya adalah pemanggilan sebuah fungsi program. |

**Tabel 2.1** ( lanjutan )

|  |  |
| --- | --- |
| **Gambar** | **Keterangan** |
|  | Extend, merupakan perluasan dari use case lain jika kondisi atau syarat terpenuhi. |

Diagram Aktivitas *(Activity Diagram*)

Activity Diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam activity diagram yaitu :

**Tabel 2.2** Simbol Activity Diagram

|  |  |
| --- | --- |
| **Gambar** | **Keterangan** |
| ACTI1 | Start point, diletakkan pada pojok kiri atas dan merupakan awal aktifitas. |
| a2 | End point, akhir aktifitas. |
| ACTI1 | Activites, menggambarkan suatu proses/kegiatan bisnis. |
| ACTI1 | Fork (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu. |
| ACTI1 | Join (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi. |
| ACTI1 | Decision Points, menggambarkan pilihan untuk pengambilan keputusan, true, false |

**Tabel 2.2** ( lanjutan )

|  |  |
| --- | --- |
| **Gambar** | **Keterangan** |
| ACTI1 | Swimlane, pembagian activity diagram untuk menunjukkan siapa melakukan apa. |

Diagram Kelas (*Class Diagram*)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem. Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan constraint yang berhubungan dengan objek yang dikoneksikan. Class diagram secara khas meliputi: Kelas (Class), Relasi, Associations, Generalization dan Aggregation, Atribut (Attributes), Operasi (Operations/Method), Visibility, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan multiplicity atau kardinaliti.

**Tabel 2.3** Multiplicity Class *Diagram*

|  |  |
| --- | --- |
| **Multiplicity** | **Keterangan** |
| 1 | Satu dan hanya satu |
| 0..\* | Boleh tidak ada atau 1 atau lebih |
| 1..\* | 1 atau lebih |
| 0..1 | Boleh tidak ada, maksimal 1 |
| n..n | Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4 |

Diagram Urutan (*Squence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam sequence diagram yaitu :

**Tabel 2.4** Simbol *Sequence* *Diagram*

|  |  |
| --- | --- |
| **Gambar** | **Keterangan** |
|  | EntityClass, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data |
|  | Boundary Class, berisi kumpulan kelas yang menjadi interface atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan form cetak. |
|  | Control class, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek. |
|  | Message, simbol mengirim pesan antar class |
|  | Activation, activation mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi |
|  | Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang lifeline terdapat activation |

* 1. **Entity Relational Diagram (ERD)**

ERD (Fridayanthie, dkk, 2016) adalah model teknik pendekatan yang menyatakan atau menggambarkan hubungan suatu model. Didalam hubungan ini tersebut dinyatakan yang utama dari ERD adalah menunjukan objek data (Entity) dan hubungan (Relationship), yang ada pada Entity berikutnya. Menurut Simarmata (2010:67), “Entity RelationShip Diagram (ERD) adalah alat pemodelan data utama dan akan mambantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antar entitas”.

Proses memungkinkan analis menghasilkan struktur basis data dapat disimpan dan diambil secara efisien. Simbol-simbol dalam ERD (Entity Relationship Diagram) adalah sebagai berikut:

a. Entitas : suatu yang nyata atau abstrak yang mempunyai karakteristik

dimana. kita akan menyimpan data.

b. Atribut : ciri umum semua atau sebagian besar instansi pada entitas

tertentu.

c. Relasi : hubungan alamiah yang terjadi antara satu atau lebih entitas.

d. Link : garis penghubung atribut dengan kumpulan entitas dan kumpulan

entitas dengan relasi.

* + 1. **Kardinalitas Relasi**

Satu ke satu (One to One) Setiap elemen dari Entitas A berhubungan paling banyak dengan elemen pada Entitas B. Demikian juga sebaliknya setiap elemen B berhubungan paling banyak satu elemen pada Entitas A.

Satu ke banyak (One to Many) Setiap elemen dari Entitas A berhubungan dengan maksimal banyak elemen pada Entitas B. Dan sebaliknya setiap elemen dari Entitas B berhubungan dengan paling banyak satu elemen di Entitas A**.**

Banyak ke satu (Many to One) Setiap elemen dari Entitas A berhubungan paling banyak dengan satu elemen pada Entitas B. Dan sebaliknya setiap elemen dari Entitas B behubungan dengan maksimal banyak elemen di entitas A.

Banyak ke banyak (Many to Many) Setiap elemen dari Entitas A berhubungan maksimal banyak elemen pada Entitas B demikian sebaliknya.

**Tabel 2.5**  Daftar simbol ERD pada aplikasi StarUML

|  |  |  |
| --- | --- | --- |
| **No** | **Simbol** | **Keterangan** |
| 1 |  | *Entity* merupakan suatu objek yang dapat diidentifikasi dalam lingkungan pengguna. |
| 2 |  | *One to One Relationship,* relasi satu ke satu berarti setiap himpunan entitas hanya boleh berhubungan dengan satu entitas lainnya. |
| 3 |  | *One to Many Relationship,* relasi satu ke banyak berarti setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya. |
| 4 |  | *Many to Many Relationship,* relasi banyak ke banyak berarti setiap himpunan entitas boleh berhubungan dengan banyak entitas lainnya dan sebaliknya. |

Untuk dokumentasinya bisa diakses langsung melalui situs resmi StarUML di <https://docs.staruml.io>.

* 1. **Framework**

Menurut Betha Sidik framework adalah : “kumpulan intruksi-intruksi yang dikumpulkan dalam class dan *function-function* dengan fungsi masing-masing untuk memudahkan developer dalam memanggilnya tanpa harus menuliskan syntax program yang sama berulang-ulang serta dapat menghemat waktu”.

* + 1. **Slim Framework**

Slim framework merupakan framework yang ditulis dengan menggunakan bahasa pemrograman php. Untuk package manager yang digunakan adalah composer. Slim mengakomodasi fungsi – fungsi yang terkait dengan pembuatan API (Apllication programming interfaces). Untuk dapat menjalankan slim diperlukan php dengan minimal versi 5.5. Pada aplikasi ini, slim framework akan digunakan sebagai *backend*. *Backend* dapat diibaratkan sebagai halaman belakang sebuah website. Kinerja dari *backend* tidak terlihat oleh mata, sebab pekerjaan ini sifatnya dibalik layar.

* + 1. **Angular Framework**

AngularJS adalah kerangka struktural untuk aplikasi web dinamis. AngularJS memungkinkan penggunanya menggunakan HTML sebagai bahasa pemrograman yang dipakai dan memungkinkan pengguna memperluas sintaks HTML untuk mengekspresikan komponen aplikasi yang dibuat oleh pengguna dengan jelas dan ringkas. Data binding AngularJS dan dependency injection AngularJS dapat memperingkas proses coding. Dan semua proses tersebut terjadi dalam browser, sehingga AngularJS mampu menjadi pasangan yang ideal dengan teknologi server. AngularJS adalah salah satu bentuk dari HTML bila ingin digunakan untuk mendesain aplikasi. HTML adalah bahasa yang cukup baik untuk dokumen statis. AngularJS adalah kerangka untuk membuat single page application. Dalam pengaplikasiannya AngularJS dapat menghasilkan halaman dinamis tanpa harus membuat banyak halaman web (Boedijono, dkk, 2015).

**2.9 *Aplication Programming Interface* (API)**

*Aplication programming interface* disingkat API adalah sekumpulan perintah, fungsi, serta protokol yang dapat digunakan oleh programmer saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi. API memiliki empat *method* yaitu:

1. GET : Mendapatkan *resource* yang ada di API.
2. POST : Menambahkan *resource* baru untuk disimpan diserver.
3. PUT : Mengubah atau memodifikasi *resource* yang sudah tersimpan.
4. DELETE: Menghapus *resource* yang tersimpan.

**2.10 Mysql**

Menurut Asrianda, Database adalah sekumpulan tabel-tabel yang saling berelasi, relasi tersebut bisa ditunjukkan dengan kunci dari tiap tabel yang ada. Satu database menunjukkan satu kumpulan data yang dipakai dalam satu lingkup perusahaan atau instansi. Database mempunyai kegunaan dalam mengatasi penyusunan dan penyimpanan data, maka seringkali masalah yang dihadapi adalah:

1. Redundansi dan Inkonsistensi data

2. Kesulitan dalam pengaksesan data

3. Isolasi data untuk standarisasi

4. Multi user

5. Keamanan data

6. Integritas data

7. Kebebasan data

MySQL adalah suatu sistem manajemen basis data relasional (RDBMS-Relational Database Management System) yang mampu bekerja dengan cepat, kokoh, dan mudah digunakan. Contoh RDBMS lain adalah Oracle, Sybase. Basis data memungkinkan anda untuk menyimpan, menelusuri, menurutkan dan mengambil data secara efesien. Server MySQL yang akan membantu melakukan fungsionaliitas tersebut.

Bahasa yang digunakan oleh MySQL tentu saja adalah SQL-standar bahasa basis data relasional di seluruh dunia saat ini.MySQL dikembangkan, dipasarkan dan disokong oleh sebuah perusahaan Swedia bernama MySQL AB. RDBMS ini berada di bawah bendera GNU GPL sehingga termasuk produk Open Source dan sekaligus memiliki lisensi komersial. Apabila menggunakan MySQL sebagai basis data dalam suatu situs Web. Anda tidak perlu membayar, akan tetapi jika ingin membuat produk RDBMS baru dengan basis MySQL dan kemudian mengualnua, anda wajib bertemu mudah dengan lisensi komersial (Urva, dkk, 2015).