BAB 2

TINJAUAN PUSTAKA

2.1. Tinjauan Perusahaan

2.1.1. Profil Tempat Kerja

Balai Besar Bahan dan Barang Teknik (B4T) beralamat di Jalan Sangkuriang No 14 Bandung 40135 Telp (022) – 2504088, 2504828, 2510682. Fax (022) – 2502027. Website www.b4t.go.id dan e-mail : info@b4t.go.id.

2.1.2. Sejarah Instansi

B4T didirikan pada tahun 1909 di Batavia (Jakarta sekarang) oleh pemerintah Hindia Belanda dengan nama Laboratorium Voor Metal Onderzoek dan berada dibawah naungan Burgelizke Openbake Warken (Departemen Pekerjaan Umum sekarang). Pada tahun 1912 diperluas menjadi Laboratorium Voor Material Onderzoek. Yang semula balai ini hanya mengurus benda-benda metal diperluas dengan benada material lainnya. Pada tahun 1921 lokasi balai dipindahkan ke Bandung. Tepatnya ke lokasi Technische Hogeschool (Institut Teknologi Bandung sekarang). Tahun 1934 kedudukan balai berpindah ke naungan Van Ekonomische (Departemen Perekonomian dan Perdagangan).

Pada tahun 1942 dibawah kekuasaan pemerintah Jepang, balai berubah nama menjadi Laboratorium Zeiro Sikendya dan kemudia berubah lagi menjadi Laboratorium Kagio Sikendya. Tahun 1945 Indoneisia merdeka dan balai ini kembali berubah menjadi Balai Penyidikan Bahan-bagan yang berkedudukan dibawah Kementrian Kemakmuran. Pada periode tahun 1952 – 1961 balai ini dua kali beralih naungan, yaitu dari Kementrian Perekonomian kek Kementrian Perindustrian. Pada tahun 1961 balai ini menempati Jalan Sangkurian Bandung dengan nama Balai Penelitian Bahan-bahan. Pada tahun 1963 keudukan balai berada dibawah Perindustrian Rakyat.

Tahun 1971 kedudukan balai berada dibawah naungan Lembaga Penelitian dan Pendidikan Industri. Pada tahun 1974 kedudukan balai berada dibawah Puslitbang Industri Logam dan Mesin Departemen Perindustrian.

Tahun 1980 kedudukan balai dbawah Badan Litbang Industri Departemen Perindustrian dan berubah nama menjadi Balai Besar Penelitian dan Pengembangan Industri Bahan dan Barang Teknik. Dan pada akhir tahun 2002 balai ini berubah nama mejadi Balai Besar Bahan dan Barang Teknik yang di singkat B4T dan dipakai hingga sekarang.

2.1.3. Logo Instransi

Berikut gambar logo instansi B4T :



Gambar 2.1 Logo Instansi B4T

Keterangan:

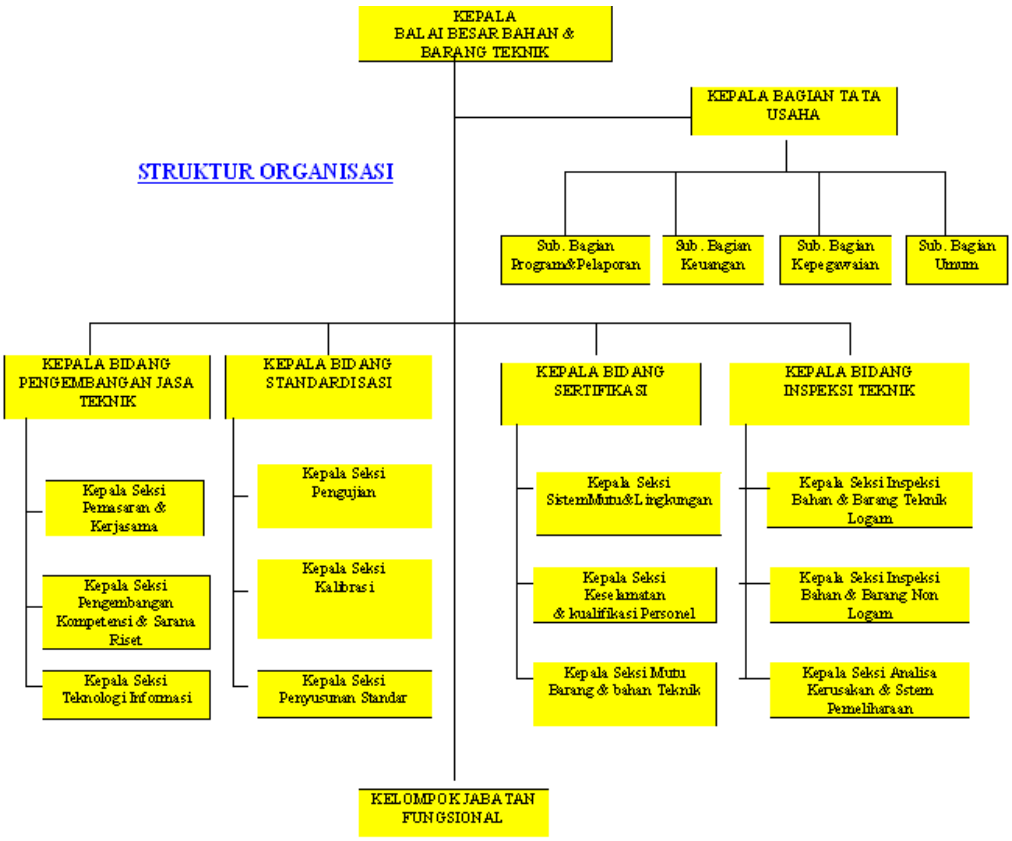
Lambang B4T merupakan singkatan dari Balai Besar Bahan dan Barang Teknik.

2.1.4. Badan Hukum Instansi

Balai Besar Bahan dan Barang Teknik merupakan balai resmi pemerintahan yang berada dibawah naungan Badan Penelitian dan Pengembangan Industri, Departemen Perindustrian. Badan Hukum oleh Mentri Perindustrian Republik Indonesia Nomor : 13/ M – IMD / PER / 6 / 2006

2.1.5. Struktur Organisasi dan Job Description

Berikut gambar Struktur Organisasi B4T:



Gambar 2.2 Struktur Organisasi B4T

2.2. Landasan Teori

Landasan teori menjelaskan beberapa teori yang berkaitan dengan permasalahan yang dibahas sebagai dasar pemahaman dalam sebuah sistem serta metode yang dipakai untuk kegiatan pengembanan terhadap sistem sendiri.

2.2.1. Sistem Berkas atau Pengarsipan

Suatu sistem untuk mengetahui bagaimana cara menyimpan data dari file tertentu dan organisasi file yang di gunakan. Pentingnya mengorganisasi dan mengolah berkas secara efektif dan efisien. [3]

2.2.2. Website

Website atau situs merupakan sekumpulan halam yang saling berhubungan menampilkan informasi data berupa teks, gambar, animasi, video, atau suara. Baik yang berupa statis maupun dinamis membentuk satu rangkaian bangunan yang saling terkait dimana masing – masing dihubungkan dengan jaringan – jaringan halaman (*hyperlink*). Web sendiri terdiri dari dua macam yaitu web statis dan web dinamis.

Web Statis merupakan situs yang memiliki isi tidak dimaksudkan untuk diperbaharui secara berkala sehingga pengaturan ataupun pemutakhiran isi atas situs web tersebut dilakukan secara manual.

Web Dinamis merupakan situs yang secara spesifik didisain agar isi yang terdapat dalam situs tersebut dapat di perbaharui secara berkala dengan mudah. Sesuai dengan namany, isi yang terkandung dalam situs web ini umumnya akan berubah setelah melewati satu periode tertentu. Situs berita adalah salah satu contoh jenis situs yang umumnya pengimplementasikan situs web dinamis. Tidak seperti halnya situs web statis, pengimplementasian situs web dinamis umumnya membutuhkan keberadaan insfratuktur yang lebih kompleks di bandingkan situs web statis. Hal ini disebakan karena pada situs web dinamis halaman web umumnya baru akan dibuat saat ada pengguna yang mengaksesnya. Berbeda dengan situs web statis yang umumnya telah membentuk sejumlah halaman web saat digunakan di server web sehingga saat pengguna mengaksesnya, umumnya pada server dilengkapi dengan mesin penterjemah bahasa skrip (PHP, ASP, ColdFusion, atau lainnya), serta perangkat lunak sistem manajemen basisdata

relational seperti MySQL. Struktur berkas sebuah situs web dinamis umumnya berbeda dengan situs web statis, berkas – berkas pada situs web statis umumnya merupakan sekumpulan berkas yang membentuk sebuah situs web. Berbeda halnya dengan situs web dinamis, berkas – berkas pada situs web dinamis umumnya merupakan sekumpulan berkas yang membentuk perangkat lunak aplikasi web yang akan dijalankan oleh mesin penerjemah server web, berfungsi memanajemen pembuatan halaman web saat halaman tersebut diminta oleh pengguna. [4]

Sebuah situs web biasa nya ditempatkan setidaknya pada sebuah server web yang dapat diakses melalui jaringan seperti internet, ataupun jaringan wilayah lokal (LAN) melalui alamat internet yang dikenali sebagai URL. Gabungan atas semua situs yang dapat diakses publik di internet tersebut pula sebagai Waring Wera Wanua atau lebih dikenal dengan singkatan WWW. Meskipun setidaknya halaman berada di situs internet umumnya dapat diakses publik secara bebas, pada prakteknya tidak semua situs memberikan kebebasan bagi publik untuk mengaksesnya, beberapa situs web mewajibkan pengunjung untuk melakukan melakukan pendaftaran sebagai anggota, atau bahkan meminta pembayaran untuk dapat menjadi anggota untuk dapat mengakses ini yang terdapat dalam situs web tersebut.

2.2.3. *Object Oriented Programming*

Object Oriented Programming atau Pemrograman Berorientasi Objek adalah pemrograman yang memiliki orientasi kebendaan, jika dibuat program tentang hewan, *programmer* harus berfikir apa nama hewan, bagaimana hewan berkembang, bagaimana hewan bergerak dan seterusnya. Jika dibuat program tentang sebuah benda, maka *programmer* harus berfikir tentang data yang berhubungan dengan benda tersebut dan apa yang dapat dikerjakan oleh maupun terhadap benda tersebut. OOP mempunyai tiga pilar pendukung, yakni : [5]

1. *Encapsulation* (enkapsulasi)
2. *Inheritance* (pewarisan)
3. *Polimorphisme* (polimorfisme)

Prinsip enkapsulasi merupakan dasar pertama bagi OOP. Menurut Siswoutomo (2005b, h. 6-8), enkapsulasi berarti menyembunyikan (membungkus) detail *class* dari *object* yang mengirimkan pesam kepadanya. *Class* adalah sebuah tipe data yang *user-defined,* berisi karakteristik abstrak dari sebuah *object* (benda), seperti atribut, properti, serta aksi yang bisa dilakukan oleh benda tersebut, atau sering dikenal dengan istilah *method*. *Object* adalah *instance* dari *class*, atau dengan kata lain *object* adalah implementasi dari *class* dengan atau parameter yang lebih konkrit.

Inheritance atau pewarisan adalah sebuah usaha untuk menurunkan *class* turunan (*sub class*) dari *class* induk yang memiliki sifat – sifat khusu, namun memiliki pula sifat – sifat yang dimiliki oleh *class* induk. *Sub class* ini biasanya memiliki karakter yang lebih khusus bila dibandingkan dengan *class* induk. Selain *single inheritance*, dikenal pula *multiple inheritance. Multiple inheritance* adalah sebuah usaha *inheritance* dengan *class* induk lebih dari satu, antara *class* induk yang satu dengan yang lain tidak terdapat hubungan *inheritance.*

Polimorfisme memungkinkan *object* memiliki karakter yang berbeda – beda saat program dijalankan, tergantung dari *class* induk yang dirujuk oleh *object*. Dengan *polimorfisme* ini, *object* mampu mengeksekusi *method* yang berbeda – beda dan menghasilkan keluaran yang berbeda – beda pula meskipun menggunakan nama *Object* yang sama. [5]

2.2.4. Basis Data

2.2.4.1. Pengertian Basis Data

Secara umum, basis data dapat diartikan sebagai kumpulan daya yang tersimpan dalam suatu tabel yang saling berelasi.

Adapun kegunaan basis data adalah :

1. Basis data merupakan komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.
2. Menentukan kualitas informasi yang akurat, tepat pada waktunya, dan relevan, dimana informasi dikatakan bernilai jika manfaatnya lebih efektif dibandingkan dengan biaya untuk mendapatkannya.

3. Mengurangi duplikasi data *(data redudancy)*.
4. Meningkatkan relasi antar data *(data relatability)*.
5. Mengurangi pemborosan *external storage*.

Di dalam basis data terdapat beberapa istilah berikut yang juga dikenal sebagai aturan jenjang data :

1. Characters

Merupakan bagian data yang terkecil yaitu karakter numerik, hutuf, maupun karakter spesial (simbol) yang membentuk item data/*field.*

2. Field

Merepresentasikan suatu atribut dari *record* yang menunjukan suatu item dari data, misalnya nama.

3. Record

Merupakan kumpulan dari *field*, menggambarkan suatu unit data individu tertentu. Kumpulan *record* membentuk suatu *file*. Misalnya, file personalian, tiap – tiap *record* mewakili data tiap – tiap karyawan.

4. File

File terdiri dari *record – record* yang menggambarkan satu kesatuan dan yang sejenis. Misalnya, *file* mata kuliah berisi data tentang semua mata kuliah yang ada.

5. Database

Merupakan kumpulan dari *file*/tabel. [6]

2.2.4.2. Sistem Manajemen Basis Data

Sistem Manajemen Basis Data adalah perangkat lunak yang didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar, serta menjadi alternatif penggunaan secara khusus untuk aplikasi, seperti penyimpanan data dalam *file* dan menulis kode aplikasi yang spesifik untuk pengaturannya. Adapun keuntungan dari DBMS (*Database Management System*) adalah :

1. *Data independence*, DBMS dapat mengurangi kebergantungan antar data.

2. *Efficient data access*, DBMS dapat menimpa data dan *meretrive* data dengan efisien. Apabila suatu saat data tersebut dibutuhkan, maka efektif dalam pemanggilannya.
3. *Data integrity and security*, saat data diakses menggunkan DBMS, DBMS mengintergrasi *constraints* dari tiap data tersebut.
4. *Data administration*, Administrasi data digunakan untuk mengatur hal-hal yang berhubungan dengan distribusi data, hak akses user yang mendistribusikan atau menggunakan data tersebut, mengorganisasikan data untuk mengurangi redudansi data dan *fine-tuning* penyimpanan data sehingga *retrive* data berjalan dengan efisien.
5. *Concurrent access and crash recovery*, DBMS dapat menjadwalkan *concurrent access* (pengaksesan secara bersamaan) terhadap suatu data sehingga menghindari dari *crash*.
6. *Reduced application development time*, DBMS mendukung banyak fungsi penting yang diakses oleh banyak aplikasi yang tersimpan dalam media penyimpanan. [6]

2.2.4.3. Sistem Manajemen Basis Data Relational

RDBMS (*Relational Database Management System*) merupakan sekumpulan data yang saling beruhubungan yang disimpan sedemikian rupa sehingga mudah diambil informasinya bagi pengguna. Ada tiga prinsip dalam RDBMS :

1. *Data Definition*

   Mendefinisikan data yang akan dibuat, seperti nama tabel dan *field* serta keterangan mengenai *field* tersebut.
2. *Data Manipulation*

   Memanipulasi data yang sudah dibuat, seperti mengubah nama *field* dan keterangannya, menghapus *record*.
3. *Data Control*

   Bagian ini berkenaan pada mengendalikan data kepada siapa saja yang bisa melihat isi data .[6]

2.2.5. UML

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebua sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas – kelas dalam bahasa pemrograman yang spesifik, skema *database*, dan komponen – komponen yang diperlukan dalam sistem *software*. [7]

2.2.4.1. Artifact UML

UML menyediakan beberapa notasi dan *artifact* standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain *Artifact* didalam UML didefinisikan sebagai informasi dalam bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat. Contohnya adalah *source code* yang dihasilkan oleh proses pemrograman.

Yang harus diperhatikan untuk menjaga konsistensi antar *artifact* selama proses analisis dan desain adalah bahwa setiap perubahan yang terjadi pada satu *artifact* harus juga dilakukan pada *artifact* sebelumnya.

Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut :

1. *Use Case Diagaram*
2. *Class Diagram*
3. *Statechart Diagram*
4. *Activity Diagram*
5. *Sequence Diagram*
6. *Collaboration Diagram*
7. *Component Diagram*
8. *Deployment Diagram*

   Dibuatnya berbagai jenis diagram diatas karena :

   a. Setiap sistem yang kompleks selalu paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil yang satu

sama lain hampir saling bebas (*independent*). Sudut pandang tunggal senantiasa tidak mencukupi untuk melihat sistem yang besar dan kompleks.

b. Diagram yang berbeda – beda tersebut dapat menyatakan tingkat yang berbeda – beda dalam proses rekayasa.
c. Diagram – diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas.

2.2.4.2. Tujuan UML

Tujuan UML diantaranya untuk :

1. Memberikan model yang siap pakai, bahasa permodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahsas permodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek – praktek terbaik yang terdapat dalam bahasa permodelan.

2.2.4.3. Notasi Dalam UML

1. Actor

Actor menggambarkan segala pengguna *software* aplikasi (*user*). *Actor* memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi. Sebagai contoh sebuah *actor* dapat memberikan input kedalam dan menerima informasi dan *software* aplikasi, perlu dicatat bahwa sebuah *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Sebuah *actor* mungkin seorang manusia, satu *device*, *hardware* atau sistem informasi lainnya.

2. Use Case

Use case menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, namum *use case* hanya menjelaskan apa yang dilakukan oleh *actor* dan sistem bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut.

a. *Use-case* Konkret adalah *use case* yang dibuat langsung karena keperluan *actor*. *Actor* dapat melihat dan berinisiatif terhadapnya.
b. *Use-case* Abstrak adalah *use case* yang tidak pernah berdiri sendiri. *Use case* abstrak senantiasa termasuk didalam (*include*), diperluas dari (*extend*) atau memperumum (*generalize*) *use case* lainnya.

Untuk menggambarkannya dalam *use case* model biasanya digunakan *association relationship* yang memiliki *stereotype include*, *extend* atau *generalization relationship*. Hubungan dari *use case* lainnya. Hubungan *extend* antar *use case* yang lain jika kondisi atau syarat tertentu terpenuhi.

3. Class

Class merupakan pembentuk utama dari sistem berorientasi obyek, karena *class* menunjukkan kumpulan *objek* yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface*.

Class digunakan untuk mengabstraksikan elemen – elemen dari sistem yang sedang dibangun. Class bisa merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata.

Notasi *class* berbentuk persegi panjang berisi 3 bagian persegi panjang paling atas untuk nama *class*, persegi panjang paling bawah untuk operasi, dan persegi panjang ditengah untuk atribut.

Atribut digunakan untuk menyimpan informasi. Nama atribut menggunakan kata benda yang bisa dengan jelas merepresentasikan informasi yang tersimpan didalamnya. Operasi menunjukan sesuatu yang bisa dilakukan oleb obyek dan menggunakan kata kerja.

4. Inteface

Inteface merupakan kumpulan operasi tanpa implementasi dari suatu *class*. Implementasi operasi dalam *interface* dijabarkan oleh operasi didalam *class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam obyek.

5. Interaction

Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun huungan antar obyek. Biasanya *interaction* ini dilengkapin juga dengan teks bernama operation *singature* yang tersusun dari mana operasi, parameter yang dikirim dan tipe parameter yang dikembalikan.

6. Note

Note digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemet sehingga bisa langusng terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.

7. Dependency

Dependency merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah yang tergantung pada elemen yang ada dibagian tanpa tanda panah.

Terdapat 2 *stereotype* dar *dependency,* yaitu *include* dan *extend. Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagain dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disiapkan kedalam elemen yang ada di garis dengan panah.

8. Association

Association menggambarkan navigasi antar *class* (*navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *class*) dan apakan suatu class menjadi bagian dari class lainnya (*aggregation*).

Navigation dilambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigiation* menunjukkan bahwa dengan mengetahui salah satu *class* bisa didapatkan informasi dari *class* lainnya. Sementara *Uni Directional navigation* hanya dengan mengetahui *class* diujung garis *association* tanpa panah kita bisa mendapakan informasi dari *class* di ujung dengan panah, tetapi tidak sebaliknya.

Aggregation mengacu pada hubungan "*has-a*", yaitu bahwa suatu *class* memiliki *class* lain, misalnya Rumah memiliki *class* Kamar.

9. Generalization

Generalization menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization*, *class* yang lebih spesifik (*subclass*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*superclass*) atau "*subclass is superclass*". Dengan menggunakan notasi *generalization* ini, konsep *inheritance* dari prinsip hirarki dapat dimodelkan.

10. Realization

Realization menunjukan hubungan bahwa elemen yang ada dibagian tanpa pernah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya *class* merealisasikan *package*, *component* merealisasikan *class* atau *interface*. [7]

2.2.6. PHP

PHP adalah bahasa skrip yang dapat dotanamkan atau disisiplan ke dalam HTML. PHP banyak dipakai untuk pemrograman situs web dinamis. PHP dapat digunakan untuk membangun sebuah CMS. [4]

2.2.5.1. Kelebihan PHP

Kelebihan PHP adalah sebagai berikut :

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

2.2.5.2. Kekurangan PHP

1. Tidak ideal untuk pengembangan skala besar
2. Tidak memiliki sistem pemrograman berorientasi objek yang sesungguhnya.
3. Tidak bisa memisahkan antara tampilan dengan logik dengan baik (walau penggunaan template dapat memberbaikinya)
4. PHP memiliki kelemahan security tertentu apabila programmer tidak jeli dalam melakukan pemrograman dan kurang memperhatikan isu dan konfigurasi PHP.
5. Kode PHP dapat dibaca semua orang, dan kompilasi hanya dapat dilakukan dengan tool yang mahal dari Zend dengan harga $2000. [4]

2.2.7. Pengertian WampServer

Wamp adalah sebuah aplikasi yang dapat menjadikan komputer menjadi sebuah web server. Kegunaan wam pserver ini untuk membuat jaringan local sendiri dalam artian kita dapat membuat website secara offline untuk masa coba – coba di komputer.

2.2.8. NetBeans

NetBeans mengacu pada dua hal, yakni platform untuk pengembangan aplikasi desktop Java, dan sebuah Integrated Development Environment (IDE) yang dibangun menggunakan platform NetBeans. Selain itu juga Netbeans dapat digunakan dalam membuat projek sebuah web aplikasi PHP yang menggunakan bahasa pemrograman berorientasi objek.

2.2.9. MySQL

2.2.7.1. Sistem Basis Data Relational

MySQL adalah sebuah implementasi dari sistem manajemen basisdata relational (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public Lisense*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk terutama yang sifatnya komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya, SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian

basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasinya dalam melakukan proses perintah – perintah SQL yang dibuat oleh pengguna maupun program – program aplikasi yang memanfaatkannya. Sebagai peladen basisdata, MySQL mendukung operasi basisdata transaksional maupun operasi basisdata nontransaksional. Pada modus operasi nontransaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namum demikian pada modus nontransaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus nontransaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis web (*wordpress*), CMS dan sejenisnya. Untuk kebutuhan sistem yang ditunjukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensunya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus nontransaksional. [4]

2.2.7.2. Keistimewaan MySQL

MySQL memiliki beberapa keistimewaan, antara lain :

1. **Portabilitas**, MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, MacOS X Server, Solaris, Amiga dan masih banyak lagi.
2. **Perangkat lunak sumber terbuka**, MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.
3. **Multiuser**, MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. **Performance tuning**, MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

5. **Ragam tiper data**, MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed / unsigned integer, float, double, char, text, date, timestamp* dan lain – lain.
6. **Perintah dan Funsi**, MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).
7. **Keamanan**, MySQl memiliki beberapa lapisan kemanan seperti lever subnetmask, nama *host*, dan izin akses user dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. **Skalabilitas dan Pembatasan**, MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*record*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapat 32 indeks pada tiap tabelnya.
9. **Konektivitas**, MySQL dapat melakukan koneksi dengan client menggunakan protokol TCP/IP, Unix soket (UNIX), atau Named Pipes (NT).
10. **Lokalisasi**, MySQL dapat mendeteksi pesan kesalah pada clients dengan menggunakan lebih dari 2 puluh bahasa. Meski pun demikian, bahasa Indoneisa belum termasuk kedalamnya.
11. **Antar Muka**, MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
12. **Clients dan Peralatan**, MySQL dilengkapi dengan berbagai peralatan (*tool*) yang dapat digunakan untuk administrasi basisdata, dan pada setiap peralatan yang ada disertakan petunjuk online.
13. **Struktur tabel**, MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basisdata lainnya semacam PostageSQL ataupun Oracle.