BAB 2

TINJAUAN PUSTAKA

2.1 Tinjauan Perusahaan

PT Rekayasa Teknologi Indonesia (REKATINDO) adalah salah satu perusahaan yang berdiri di indonesia, perusahaan ini bergerak di bidang pengembangan dan pemeliharaan *aplikasi* dan *hardware*. Produk indutri dan komersial *platform* yang dibuat rekatindo lebih di khususkan untuk militer dan sipil.

2.2 Sejarah Instansi

PT. Rekayasa Teknologi Indonesia (REKATINDO), didirikan pada tahun 2003 oleh para insinyur yang memiliki pengalaman dalam perancangan dan pengembangan proyek rekayasa dan teknologi.

Dimulai pada bulan januari 2004, PT Rekatindo memulai proyek untuk mabes TNI AU dengan mendesain dan membuat *Vehicle Command Post* (VCP I), Proyek ini berjalan selama 4 bulan. Lalu di bulan Mei hingga desember 2004 Rekatindo juga melakukan perbaikan pada RPBS JAATS Radar di Bandara Soekarno hatta/PT.Angkasa Pura II.

Berlanjut pada Februari mulai mendesain & mengembangan Komunikasi dan Sistem *Audio Mobile Outdoor* untuk KTT Asia-Afrikaatas permintaan Paspampres RI, diwaktu yang bersamaan juga mereka membuat Desain & Pengembangan *Tower Set* - Lanud Abd, di malang. Pembuatan kedua proyek ini membutuhkan waktu 7 bulan yang berakhir pada bulan Agustus 2005.

Di Awal tahun 2006 PT Rekatindo mendapat proyek dari Mabes TNI AU untuk proyek pembuatan tower set untuk wilayah Madiun dan Bogor, selagi proyek ini berjalan PT Rekatindo juga membuat desain untuk VCP 3. Setelah menyelesaikan pembuatan tower, maka pada bulan april mereka membuat proyek baru yakni membuat sebuah game perang dan juga simulator tempur yang dimulai pada bulan April dan hingga kini pengembangan dari aplikasi tersebut masih di kembangkan.

Pada tahun 2007 PT Rekatindo membuat digital *SignalProcessor* untuk perusahaan *Merpati Training Center*, proyek ini dikerjakan pada bulan maret. Lalu perancangan desain dan tower set di Tasikmalaya pada bulan yang sama.

Di tahun 2008 tepatnya di july PT Rekatindo membuat *WIRE BUNDLE PANZER* V1 atas permintaan PT PINDAD, selama itu PT Rekatindo mempelajari dan membuat *prototype* dari PANZER tersebut, dan tidak lama berselang pada bulan Desember PT PINDAD langsung memesan 40 unit PANZER ini[11].

2.3 Logo Instansi

Dibawah ini adalah logo perusahaan pada saat berdirinya sampai dengan sekarang tidak mengalami perubahan.

Gambar 2.1 Logo Perusahaan

2.4 Struktur Organisasi

Struktur Organisasi dan *Job Description* menjelaskan struktur organisasi yang ada didalam PT. Rekayasa Teknologi Indonesia dan penjelasan pekerjaan yang dilakukan dari setiap bagiannya.



Gambar 2.2 Struktur Organisasi

2.5 Job Description

Pada perusahaan PT.Rekayasa Teknologi (REKATINDO) memiliki beberapa pegawai tetap dan pegawai kontrak.PT ini dipimpin oleh satu presiden *director* utama dan juga ada 3 director lainnya serta 1 orang yang mengurus bagian asuransi.Berikut adalah *Job Description* di PT.Rekayasa Teknologi :

1. ***Presiden Director***

   Presiden director selaku pemegang jabatan tertinggi dalam perusahaan ini mempunyai tugas yaitu :

   1. Memimpin seluruh dewan atau komite eksekutif.
   2. Menawarkan visi dan imajinasi di tingkat tertinggi.
   3. Memimpin rapat umum, dalam hal : untuk memastikan pelaksanaan tata-tertib, keadilan dan kesempatan bagi semua untuk berkontribusi secara tepat, menyesuaikan alokasi waktu per item masalah, menentukan urutan agenda, mengarahkan diskusi kearah consensus serta menjelaskan dan menyimpulkan tindakan dan kebijakan.
   4. Bertindak sebagai perwakilan organisasi dalam hubungannya dengan dunia luar.

5. Memainkan bagian terkemuka dalam menentukan komposisi dari board dan sub-komite, sehingga tercapainya keselarasan dan efektivitas.
6. Mengambil keputusan sebagaimana didelegasikan oleh dewan direksi atau pada situasi tertentu yang dianggap perlu, yang diputuskan, dalam meeting-meeting dewan direksi.
7. Menjalankan tanggung jawab dari direktur perusahaan sesuai dengan standar etika dan hukum.

2. *Director of Commercial*

1. Menghitung biaya yang dibutuhkan di setiap proyek yang di buat mulai dari operasional hingga ongkos produksi.
2. Menangani setiap Transaksi dengan client mulai dari perbincangan harga jadi produk hingga tawar menawar.

3. *Director of Production &Engginering*

1. Mengkoordinasikan dan mengendalikan kegiatan di bidang perencanaan teknik ,produksi , distribusi dan perawatan teknik.
2. Mengkoordinasikan dan mengendalikan pemeliharaan .
3. Melaksanakantugas – tugas lain yang diberikan oleh direktur utama.
4. Dalam menjalankan tugasnya ,direktur teknik bertanggung jawab kepada direktur utama.
5. Mengkoordinasikan kegiatan – kegiatan pengujian peralatan teknik sesuai pesanan customer atau pun instansi.

4. *Director of Finance & ADM*

1. Mengendalikan kegiatan – kegiatan bidang keuangan.
2. Megendalikan program dan pendapatan pengeluaran perusahaan.
3. Merencanakan dan mengendalikan sumber – sumber pendapatan serta pembelanjaan dan kekayaan perusahaan.
4. Melaksanakan tugas yang diberikan oleh atasan.
5. Dalam menjalankan tugasnya bertanggung jawab kepada direktur utama.

5. ***Quality Assurance***

    1. Mengecek kelayakan dari setiap produk.
    2. Memberi jaminan asuransi untuk produk yang di buat[11].

2.6 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 pengembangan terhadap sistem sendiri.

2.6.1 Aplikasi

Perangkat lunak aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media.

Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi (*application suite*). Contohnya adalah Microsoft Office dan Open Office.org, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah[3] .

2.6.2 UML

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah 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 program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem *software* [6].

2.6.2.1 **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 *atifact* sebelumnya.

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

1. *Use Case Diagram*
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 tingkatan yang berbeda-beda dalam proses rekayasa.

c. Diagram-diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas

2.6.2.2 **Tujuan UML**

Tujuan utama UML diantaranya untuk :

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

2.6.2.3 **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 dari *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, namun *Use Case* hanya menjelaskan apa yang dilakukan oleh *actor* dan sistem bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut.

1. *Use-case* Konkret adalah *Use Case* yang dibuat langsung karena keperluan actor. Actor dapat melihat dan berinisiatif terhadapnya
2. *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 *include* menggambarkan bahwa suatu

Use Case seluruhnya meliputi fungsionalitas dari *Use Case* lainnya. Hubungan *extend* antar *Use Case* berarti bahwa satu *use case* merupakan tambahan fungsionalitas dari *Use Case* yang lain jika kondisi atau syarat tertentu terpenuhi.

3. *Class*

Class merupakan pembentuk utama dari sistem berorientasi *obyek*, karena *class* menunjukkan kumpulan *obyek* 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 menunjukkan sesuatu yang bisa dilakukan oleh *obyek* dan menggunakan kata kerja.

4. *Interface*

Interface 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 hubungan antar *obyek*. Biasanya *interaction* ini dilengkapi juga dengan teks bernama *operation signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan.

6. *Note*

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

7. *Depedency*

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

Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan 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 apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*).

Navigation dilambangkan dengan penambahan tanda panah di akhir garis. *Bidirectional navigation* 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 mendapatkan 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 menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya *class* merealisasikan *package*, component merealisasikan *class* atau *interface* [7] .

2.6.3 **Java**

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Dikembangkan oleh Sun Microsystems dan diterbitkan tahun 1995. Java tidak boleh disalahpahami sebagai JavaScript. JavaScript adalah bahasa scripting yang digunakan oleh *web browser*.

2.6.3.1 **Versi Awal Java**

Versi awal Java ditahun 1996 sudah merupakan versi *release* sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya :

1. java.lang: Peruntukan kelas elemen-elemen dasar.
2. java.io: Peruntukan kelas input dan output, termasuk penggunaan berkas.
3. java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.
4. java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
5. java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)

6. java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

2.6.3.2 **Kelebihan Java**

1. Multiplatform. Kelebihan utama dari Java ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas *operating system* Linux tetapi dijalankan dengan baik di atas Microsoft Windows. Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk meninterpretasikan *bytecode* tersebut.
2. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut *Object*. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan Java sebagai salah satu bahasa pemograman termudah, bahkan untuk fungsi yang *advance* seperti komunikasi antara komputer sekalipun.
3. Perpustakaan Kelas Yang Lengkap, Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

4. Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.
5. Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.6.3.3 **Kekurangan Java**

1. Tulis sekali, perbaiki di mana saja - Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE misalnya SWT-AWT *bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.
2. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/*direverse-engineer*.
3. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena *trend* memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun [7].

2.6.4 **NetBeans**

NetBeans mengacu pada dua hal, yakni platform untuk pengembangan aplikasi desktop java, dan sebuah Integrated Development Environment (IDE) yang dibangun menggunakan platform NetBeans.

Platform NetBeans memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut ”modul”. Sebuah modul adalah suatu arsip Java (*Java archive*) yang memuat kelas-kelas Java untuk berinetraksi dengan NetBeans Open API dan *file* manifestasi yang mengidentifikasinya sebagai modul. Aplikasi yang dibangun dengan modul-modul dapat dikembangkan dengan menambahkan modul-modul baru. Karena modul dapat dikembangkan secara independen, aplikasi berbasis platform NetBeans dapat dengan mudah dikembangkan oleh pihak ketiga secara mudah dan *powerful*.

2.6.4.1 **Platform NetBeans**

Platform NetBeans adalah *framework* yang dapat digunakan kembali (*reusable*) untuk menyederhanakan pengembangan aplikasi desktop. Ketika aplikasi berbasis platform NetBeans dijalankan, kelas Main dari platform dieksekusi.Modul-modul yang tersedia ditempatkan di sebuah registry di dalam memori, dan tugas startup modul dijalankan. Secara umum, kode modul dimuatkan ke dalam memori hanya ketika ia diperlukan. Aplikasi dapat menginstal modul secara dinamis. Aplikasi dapat memasukkan modul *Update Center* untuk mengijinkan pengguna aplikasi men- *downloaddigitally-signed* *upgrade* dan fitur-fitur baru secara langsung ke dalam aplikasi yang berjalan. Penginstalan kembali sebuah *upgrade* atau rilis baru tidak memaksa pengguna untuk men-*download* keseluruhan aplikasi lagi.

Platform NetBeans menawarkan layanan-layanan yang umum bagi aplikasi desktop, mengijinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi. Fitur-fitur yang disediakan oleh platform NetBeans:

a. Manajemen antarmuka (misal: menu & *toolbar*)

b. Manajemen pengaturan pengguna

c. Manajemen penyimpanan (menyimpan dan membuka berbagai macam data)

d. Manajemen jendela

e. *Wizard framework* (mendukung dialog langkah demi langkah)

2.6.4.2 **NetBeans IDE**

NetBeans IDE adalah IDE open source yang ditulis sepenuhnya dengan bahasa pemrograman Java menggunakan platform NetBeans. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (J2SE, web, EJB, dan aplikasi mobile). Fitur lainnya adalah sistem proyek berbasis Ant, kontrol versi, dan refactoring. Versi terbaru saat ini adalah NetBeans IDE 5.5.1 yang dirilis Mei 2007 mengembangkan fitur-fitur Java EE yang sudah ada (termasuk Java Persistence support, EJB-3 dan JAX-WS). Sementara paket tambahannya, NetBeans Enterprise Pack mendukung pengembangan aplikasi perusahaan Java EE 5, meliputi alat desain visual SOA, skema XML, web service dan pemodelan UML. NetBeans C/C++ Pack mendukung proyek C/C++.

Modularitas: Semua fungsi IDE disediakan oleh modul-modul. Tiap modul menyediakan fungsi yang didefinisikan dengan baik, seperti dukungan untuk bahasa pemrograman Java, editing, atau dukungan bagi CVS. NetBeans memuat semua modul yang diperlukan dalam pengembangan Java dalam sekali download, memungkinkan pengguna untuk mulai bekerja sesegera mungkin. Modul-modul juga mengijinkan NetBeans untuk bisa dikembangkan. Fitur- fitur baru, seperti dukungan untuk bahasa pemrograman lain, dapat ditambahkan dengan menginstal modul tambahan.Sebagai contoh, Sun Studio, Sun Java Studio Enterprise, dan Sun Java Studio Creator dari Sun Microsystem semuanya berbasis NetBeans IDE [7].

2.6.4.3 **MySQL**

MySQL adalah sebuah implementasi dari sistem manajemen basisdata relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (General Public License). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat 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.

1.6.4.3.1 Keistimewaan MySQL

MySQL memiliki beberapa keistimewaan, antara lain :

1. **Portabilitas**. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os 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 tipe 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 Fungsi**. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).
7. **Keamanan**. MySQL memiliki beberapa lapisan keamanan seperti level 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 (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. **Konektivitas**. MySQL dapat melakukan koneksi dengan clients menggunakan protokol TCP/IP, Unix soket (UNIX), atau Named Pipes (NT).

10. **Lokalisasi**. MySQL dapat mendeteksi pesan kesalahan pada clients dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.
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 basis data, dan pada setiap peralatan yang ada disertakan petunjuk online.
13. **Struktur tabel**. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle [4].