BAB 2
TINJAUAN PUSTAKA

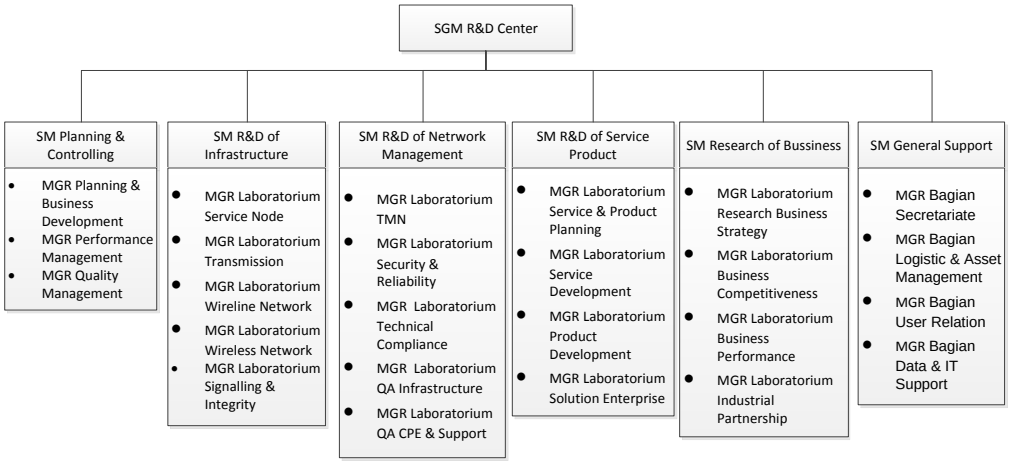
2.1 Tinjauan Instansi

2.1.1 Profil Instansi

Telkom R&D Center adalah merupakan unit bisnis Pendukung PT. Telekomunikasi Indonesia, Tbk yang secara struktural bertanggung jawab langsung kepada Direktur *Network & Solution*. Sejalan dengan perUbahan pengorganisasian bisnis menuju pada model customer centric *organization*, fungsi riset dan pengembangan perusahaan lebih diberdayakan dan focus pada peran membangun kapabilitas perusahaan dalam mempersiapkan pengembangan service dan produk unggulan serta dapat mengantisipasi trend perkembangan bisnis yang berbasis teknologi informasi dan komunikasi.

2.1.2 Struktur Organisasi

Stuktur organisasi merupakan gambaran mengenai hubungan jalur komunikasi diantara bagian-bagian atau orang-orang yang ada di dalam perusahaan sehingga dapat diketahui bagaimana tanggung jawab dan wewenang yang terdapat diantara bagian tersebut.



Gambar 2. 1 Struktur Organisasi

2.1.3 Keterangan Bidang dan Bagian / Laboratorium

1. SGM R&D *CENTER*

   Proses Utama :

   a. Perencanaan Bisnis, pengelolaan performansi dan operasional unit serta pengendalian sistem mutu unit R&D Center.
   b. Pengelolaan dan penyelenggaraan kegiatan riset dan pengembangan Infrastruktur, Management Network/Jaringan, Service & Product dan Bisnis.

2. BIDANG *PLANNING & CONTROLLING*

   Proses Utama :

   a. Perencanaan bisnis (Strategic Plan)
   b. Penyusunan & Evaluasi RKM, RKAP dan SKU
   c. Pengelolaan performansi unit
   d. Pengendalian sistem mutu

3. BIDANG R&D *OF INFRASTRUCTURE*

   Proses Utama :

   a. Riset teknologi untuk mendukung penyusunan rencana strategis pengembangan infrastruktur dan teknologi
   b. Perencanaan infrastruktur (*service node, transmisi, signaling & integrity, wireline & wireless network*) untuk mendukung *service deployment*
   c. Evaluasi performansi jaringan infrastruktur (*service node, transmisi, signaling & integrity, wireline & wireless network*) termasuk performansi interkoneksi
   d. *Technology scanning* dan *technology assessment* untuk mendukung penyelenggaraan bisnis perusahaan
   e. Support project management inovation/support expertise

4. BIDANG R&D *OF NETWORK MANAGEMENT*

   Proses Utama :

   a. Analisis *regulative resources* untuk mendukung evaluasi performansi pengelolaan jaringan telekomunikasi.
   b. Pengelolaan riset jaringan telekomunikasi.
   c. *Technology assessment* dalam bidang manajemen jaringan untuk mendukung penyelenggaraan bisnis perusahaan.
   d. Pengelolaan Layanan *Quality Assurance dan Technical compliance management.*
   e. *Support project management inovation/support expertise*.

5. BIDANG R&D *OF SERVICE & PRODUCT*

   Proses Utama :

   a. Pengembangan *service & product.*
   b. Riset & pengembangan *prototype* servise baru dan penyusunan *standart service.*
   c. Riset dan pengembangan *prototype* produk baru.
   d. Pengembangan *prototype* solusi bisnis untuk pelanggan korporat & OLO.
   e. Service management.

6. BIDANG *RESEARCH BUSINESS*

   Proses Utama :

   a. Pelaksanaan riset dan pengembangan bisnis
   b. Evaluasi dan identifikasi performansi bisnis
   c. Penyediaan data pasar, pelanggan dan kompetitor yang kompetitif
   d. Pengembangan hubungan kemitraan yang strategis dengan institusi yang relevan

7. BIDANG *GENERAL SUPPORT*

Proses Utama :

a. Pengelolaan kesekretariatan
b. Pengelolaan Procurement, Asset & Facilities
c. Pengelolaan Relasi & Klien
d. Dukungan pengelolaan Data & IT Unit

2.1.4 Laboratorium

Telkom R&D Center Bandung memiliki beberapa laboratorium, laboratorium tersebut memiliki tanggung jawab, tugas dan fungsi masing – masing yang dijelaskan sebagai berikut :

1. Laboratorium *Service Node*

   Tanggung Jawab dan Fungsi :

   a) Membuat perencanaan, pengembangan standar dan melaksanakan pemutakhiran *document* yang terkait dengan teknologi *service node*.
   b) Melaksanakan evaluasi, mengidentifikasi dan memberikan rekomendasi roadmap teknologi, rencana pengembangan teknologi dan penyusunan Business Plan atas pengembangan infrastuktur.

2. Laboratorium *Wireline Access*

   Tanggung Jawab dan Fungsi :

   a) Melakukan asesmen teknologi dan pemilihan teknologi untuk mendukung *infrastructure development.*
   b) Melakukan dan mengelola riset pengembangan teknologi wireline
   c) Memberi rekomendasi *roadmap* teknologi / infrastruktur dan rekomendasi rencana pengembangan infrstruktur untuk mendukung *Buseiness Plan* pengembangan infrastruktur.

3. Laboratorium *Wireless Access*

   Tanggung Jawab dan Fungsi :

   a) Melakukan asesmen teknologi dan pemilihan teknologi untuk mendukung *infrastructur development*
   b) Melakukan dan mengelola riset pengembangan teknologi *wireless*.
   c) Memberi rekomendasi roadmap teknologi / infrastruktur dan rekomendasi rencana pengembangan infrstruktur untuk mendukung *Buseiness Plan* pengembangan infrastruktur.

4. Laboratorium Transmission

   Tanggung Jawab dan Fungsi :

   a) Melakukan asesmen teknologi dan pemilihan teknologi transmissi untuk seluruh aspek.
   b) Melakukan riset dan pengembangan teknologi transmissi.
   c) Memberikan rekomendasi terhadap *roadmap* teknologi dan pengembangan serta penyusunan *Business Plan* infrastruktur.

5. Laboratorium TMN

   Tanggung Jawab dan Fungsi :

   a) Melakukan asessment teknologi dan pemilihan teknologi *telco management nework.*
   b) Melakukan riset & pengembangan konsep TMN yang berbasis TDM, *IP dan Mobile*
   c) Memberikan bantuan *expertise* tentang pengujian.

6. Laboratorium *Reliability & Security*

   Tanggung Jawab dan Fungsi :

   a) Melakukan riset dan pengembangan sistem *reliability* dan *security* sistem jaringan , sistem energi telekomunikasi dan Satelit untuk mendukung bisnis jaringan

b) Melakukan uji coba teknis dan pengembangan standar *system Reliability & Security* termasuk system frekuensi dan *numbering*.

7. Laboratorium *Signalling & Integrity*

Tanggung Jawab dan Fungsi :

a) Melakukan riset, pengembangan dan perencanaan sistem *signalling & Integrity* untuk mendukung penyusunan *Business Plan* Pengembangan Infrastruktur.
b) Mendukung pelaksanaan evaluasi untuk identifikasi performansi dan *interkoneksi eksisting*.
c) Memberikan usulan alternatif konfigurasi interkoneksi yang lebih prospektif.

8. Laboratorium *Technical Compliance*

Tanggung Jawab dan Fungsi :

a) Melakukan *technical compliance* melalui evaluasi teknis dan audit
b) Memberikan rekomendasi dan laporan atas *technical compliance.*
c) Melaksanakan riset dan pengembangan *fraud* dan *revenue assurance* serta *network compliance.*
d) Memberi rekomendasi dan sosialisasi hasil riset *fraud revenue*
e) Memberi bantuan expertise tentang *network technical compliance*, *fraud* dan *revenue assurance*.

9. Laboratorium QA Infrastruktur

Tanggung Jawab dan Fungsi :

a) Melaksanakan pengujian perangkat infrastruktur telekomunikasi (Transmisi, *Wireless , Wireline , Satelite , FO, Cable & Assesories*)
b) Melakukan pemeriksaan, evaluasi teknis, rekomendasi dan membuat laporan hasil uji
c) Melaksanakan kalibrasi alat ukur untuk internal maupun eksternal
d) Memberikan bantuan *expertise* tentang pengujian.

10. Laboratorium QA CPE & *Support*

Tanggung Jawab dan Fungsi :

a) Melaksanakan pengujian perangkat *switching / Node*, CPE dan Energi
b) Melakukan pemeriksaan, evaluasi teknis, rekomendasi dan membuat laporan hasil uji
c) Mengembangkan Sistem manajemen mutu jaringan telekomunikasi CIQS
d) Memberikan bantuan expertise tentang pengujian dan pengelolaan CIQS.

11. Laboratorium *Service & Product Planning*

Tanggung Jawab dan Fungsi :

a) Memberikan rekomendasi tentang kebijakan dan strategi dalam *road map* pengembangan service & Produk
b) Melakukan evaluasi dan kajian terhadap service dan produk untuk peningkatan performansi.

12. Laboratorium *Service Development*

Tanggung Jawab dan Fungsi :

a) Melakukan kajian teknis terhadap rencana pengembangan *service* baru
b) Memberikan rekomendasi pengembangan *service* baru
c) Mengembangkan dan mempersiapkan implementasi *prototype service* baru.

13. Laboratorium Produk *Development*

Tanggung Jawab dan Fungsi :

a) Melakukan kajian teknis terhadap rencana pengembangan produk baru
b) Memberikan rekomendasi pengembangan Produk baru
c) Mengembangkan dan mempersiapkan implementasi *prototype* produk baru.

14. Laboratorium *Enterprise Solution*

Tanggung Jawab dan Fungsi :

a) Melakukan riset tentang peluang solusi dan teknologi
b) Mengembangan *prototype* solusi enterprise dan dukungan atas pengembangan aplikasi internal.

15. Laboratorium *Business Strategy*

Tanggung Jawab dan Fungsi :

a) Melakukan riset strategi dan roadmap bisnis portofolio
b) Melakukan kajian dan rekayasa bisnis untuk pengembangan *service* dan produk baru
c) Melakukan riset strategi marketing.

16. Laboratorium *Business Performance*

Tanggung Jawab dan Fungsi :

a) Melakukan riset dan memberikan rekomendasi tentang peningkatan performansi dan strategi *marketing / pricing*
b) Melakukan analisis dan evaluasi bisnis terhadap produk eksisting.

17. Laboratorium *Business Competitivenes*

Tanggung Jawab dan Fungsi :

a) Melakukan riset dan *intelligent marketing* serta analisa hasil riset dan *intelligent marketing*
b) Melakukan riset dan analisa customer, kajian *business competitiveness* serta kajian atas aktivitas yang dilakukan competitor, pasar dan pelanggan.

18. Laboratorium *Industrial Partnership*

Tanggung Jawab dan Fungsi :

a) Membangun dan mengembangkan hubungan kemitraan

b) Mengelola hasil innovasi, riset & hak cipta / paten yang telah dipe*Role*h.
c) Mengelola kerjasama untuk komersialisasi hasil riset, *Monitoring* kegiatan ICT *community development* program.

19. Bagian Data & IT *Support*

Tanggung Jawab dan Fungsi :

a) Menyusun, mengimplementasikan dan mengoperasikan infrastruktur SIM (Sistem Informasi Manajemen).
b) Memelihara sistem basis data corporasi (CDB).
c) Melakukan koordinasi, integrasi dan pengoperasian aplikasi on line baru untuk keperluan perkembangan layanan internal R & D *Center*.
d) Melakukan *clustering* data *center* untuk menjamin *business continuity system* dan internal IT asesmen.

2.2 Landasan Teori

2.2.1 Aplikasi

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.

2.2.2 *Monitoring*

Monitoring adalah proses pengumpulan analisis informasi (berdasarkan indikator yang ditetapkan) secara sistematis dan kontinu tentang kegiatan program / proyek sehingga dapat dilakukan tindakan koreksi untuk penyempurnaan program / proyek itu selajutnya [4].

Adapun pengertian *Monitoring* menurut para ahli :

1. Cassely dan Kumar 1987

   Monitoring merupakan program yang terintegrasi, bagian penting dipraktek manajemen yang baik dan arena itu merupakan bagian integral di manajemen sehari-hari.

2. Calyton dan Petry 1983

   Monitoring sebagai suatu proses mengukur, mencatat, mengumpulkan, memproses dan mengkomunikasikan informasi untuk membantu pengambilan keputusan manajemen program/proyek.

3. Oxfam 1995

   Monitoring adalah mekanisme yang sudah menyatu untuk memeriksa yang sudah untuk memeriksan bahwa semua berjalan untuk direncanakan dan memberi kesempatan agar penyesuaian dapat dilakukan secara metodologis.

4. SCF 1995

   Monitoring adalah penilaian yang skematis dan terus menerus terhadap kemauan suatu pekerjaan.

5. (WHO)

   Monitoring adalah suatu proses pengumpulan dan menganalisis informasi dari penerapan suatu program termasuk mengecek secara reguler untuk melihat apakah kegiatan/program itu berjalan sesuai rencana sehingga masalah yang dilihat /ditemui dapat diatasi.

6. *Monitoring* menurut *Webster's New Collegiate Dictionary* (1981) adalah: "*a device for observing or giving admonition or warning*". Sementara itu menurut *Webstern's New World Dictionary*, maka pengertian "*Monitoring* adalah *something that reminds or warns' or any of various devices for checking or regular the performance*".

Tujuan dari *Monitoring* adalah sebagai berikut :

1. Mengkaji apakah kegiatan-kegiatan yang dilaksanakan telah sesuai dengan rencana.
2. Mengidentifikasi masalah yang timbul agar langsung dapat diatasi.
3. Melakukan penilaian apakah pola kerja dan manajemen yang digunakan sudah tepat untuk mencapai proyek.
4. Mengetahui kaitan antara kegiatan dengan tujuan untuk mempe*Role*h ukuran kemajuan.
5. Menyesuaikan kegiatan dengan lingkungan yang berUbah, tanpa menyimpang dari tujuan.

Sedangkan manfaat dari *Monitoring* adalah :

1. Bagi pihak penanggung jawab program
    a. Salah satu fungsi manajemen yaitu pengendalian atau supervisi.
    b. Sebagai bentuk pertanggungjawaban (akuntabilitas) kinerja.
    c. Untuk meyakinkan pihak-pihak yang berkepentingan.
    d. Membantu penentuan langkah-langkah yang berkaitan dengan kegiatan proyek selanjutnya.
    e. Sebagai dasar untuk melakukan *Monitoring* dan evaluasi selanjutnya.
2. Bagi pihak pengelola proyek
    a. Membantu untuk mempersiapkan laporan dalam waktu singkat.
    b. Mengetahui kekurangan-kekurangan yang perlu diperbaiki dan menjaga kinerja yang sudah baik.
    c. Sebagai dasar (informasi) yang penting untuk melakukan evaluasi proyek.

2.2.3 Basis Data

Basis data terdiri atas dua kata yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek

seperti manusia (pegawai, peserta, pembeli, pelanggan), barang, hewan, peristiwa konsep, keadaan dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.

Basis data dapat didefinisikan dari sejumlah sudut pandang, seperti:

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan file atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

Tujuan utama *DBMS* adalah untuk menyediakan tinjauan abstrak dari data bagi *User*. Jadi sistem menyembunyikan informasi mengenai bagaimana data disimpan dan dirawat, tetapi data tetap dapat diambil dengan efisien. Pertimbangan efisien yang digunakan adalah bagaimana merancang struktur data yang kompleks, tetapi tetap dapat digunakan oleh pengguna yang masih awam, tanpa mengetahui kompleksitas struktur data. Basis data menjadi penting karena munculnya beberapa masalah bila tidak menggunakan data yang terpusat, seperti adanya duplikasi data, hubungan antar data tidak jelas, organisasi data dan *update* menjadi rumit. Jadi tujuan dari pengaturan data dengan menggunakan basis data adalah :

1. Menyediakan penyimpanan data untuk dapat digunakan oleh organisasi saat sekarang dan masa yang akan datang.
2. Kemudahan pemasukan data, sehingga meringankan tugas operator dan menyangkut pula waktu yang diperlukan oleh pemakai untuk mendapatkan data serta hak-hak yang dimiliki terhadap data yang ditangani.
3. Pengendalian data untuk setiap siklus agar data selalu *up-to-date* dan dapat mencerminkan perUbahan spesifik yang terjadi di setiap sistem.
4. Pengamanan data terhadap kemungkinan penambahan, pengUbahan, pengerusakan dan gangguan-gangguan lain.

Oprasi-oprasi dasar pada basis data:

1. Pembuatan basis data baru (*Create Database*)
2. Penghapusan basis data (*Drop Database*)
3. Pembuatan Tabel Baru (*Create Table*)
4. Penghapusan Tabel (*Drop Table*)
5. Pengisian atau penambahan data baru (*Insert Data*)
6. Pengambilan data dari suatu table (*Retrieve Data*)
7. PengUbahan Data dari suatu table (*update Data*)
8. Penghapusan data dari suatu table (*Delete Data*)

2.2.3.1 Basis Data Relasional

Basis data relational adalah basis data yang dilihat oleh pemakai sebagai sekumpulan tabel dua dimensi yang saling berelasi dan ternormalisasi dengan derajat yang beragam untuk mengambarkan suatu berkas data. Model ini menunjukan cara mengorganisasikan data secara fisik dalam memori sekunder, yang akan berdampak pada keseluruhan data yang terkait dalam sistem yang dibangun.

2.2.3.2 Element Basis Data

1. Entitas

Entitas adalah sekumpulan objek yang terdefinisikan yang mempunyai karakteristik sama dan bisa dibedakan satu dengan lainnya. Objek dapat berupa barang, orang, tempat atau suatu kejadian. Contoh entitas : Seseorang yang menjadi siswa di sebuah sekolah, barang yang menjadi inventaris suatu perusahaan**.** Keterangan: siswa merupakan entitas dan barang juga di sebut entitas

2. Atribut

Atribut adalah deskripsi data yang bisa mengidentifikasi entitas yang membedakan entitas tersebut dengan entitas yang lain. Seluruh atribut harus cukup untuk menyatakan identitas obyek, atau dengan kata lain, kumpulan atribut dari setiap entitas dapat mengidentifikasi keunikan suatu individu. Contoh siswa memiliki atribut : no siswa, alamat siswa, dll.

3. Data *Value* (*Nilai Data*)

Data *Value* (*Nilai Data*) adalah data aktual atau informasi yang disimpan pada tiap data, elemen, atau atribut. Atribut nama mahasiswa menunjukan tempat dimana informasi nama mahasiswa disimpan, nilai datanya misalnya adalah Andra, Andri Andro dan lain-lain yang merupakan isi data nama mahasiswa tersebut.

4. File/Table

File/Table adalah kumpulan *record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama, namun berbeda nilai datanya.

5. Record/Tuple

Record/Tuple adalah kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu *entitas* secara lengkap. Satu *record* mewakili satu data atau informasi.

2.2.3.3 Entity-Relationship (ER)

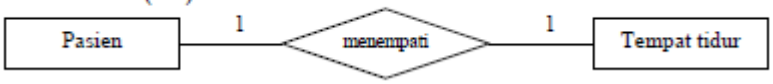
Model data *entity-relationship* (E-R) merupakan model data yang didasarkan pada perpsepsi bahwa dunia nyata merupakan sekumpulan dari jumlah objek (entitas) dasar dan relasi antar objek-objek data tersebut, dimana model ini menggunakan diagram hubungan antara entitas dengan entitas lain atau *relationship*.

Tipe *relationship* tergantung jumlah maksimum kardinalitas setiap entitas. Ada 4 tipe *relationship*, yaitu:

1. Satu-ke-satu (1:1)

Tingkat hubungann satu-ke-satu, dinyatakan dengan satu kejadian pada entitas pertama, hanya mempunyai satu hubungan dengan satu kejadian pada entitas yang kedua dan sebaliknya.

- One to one (1:1)



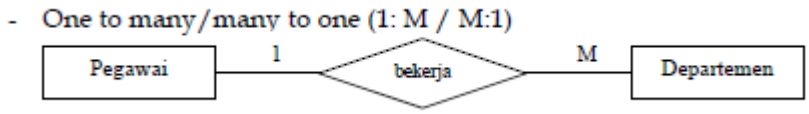
Gambar 2. 2 Relasi satu ke satu

2. Satu-ke-banyak (1:N)

Tingkat hubungan satu-ke-banyak adalah untuk satu kejadian pada entitas yang pertama dapat mempunyai banyak hubungan dengan kejadian pada entitas yang kedua.

3. Banyak-ke-satu (N:1)

Untuk banyak kejadian pada entitas yang pertama hanya dapat mempunyai satu hubungan dengan kejadian pada entitas yang kedua.



Gambar 2. 3 Relasi satu ke banyak

4. Banyak-ke-banyak (N:N) atau (M:N)

Tingkat hubungan banyak ke banyak terjadi jika tiap kejadian pada sebuah entitas akan mempunyai banyak hubungan dengan kejadian pada entitas lainnya, baik dilihat dari sisi entitas yang pertama maupun dilihat dari sisi yang kedua.



Gambar 2. 4 Relasi banyak ke banyak

2.2.4 *Electronic office (E-Office)*

E-Office merupakan layanan aplikasi *Admini*strasi perkantoran, korespondensi surat-menyurat bagi setiap karyawan suatu perusahaan/institusi secara elektronis (*office automation*) [3].

Electronic dalam E-Office dapat berarti bahwa semua pekerjaan yang berhubungan dengan *Admini*strasi perkantoran dikerjakan secara elektronis dan menggunakan bantuan alat komunikasi dan sistem informasi. Dengan

perkembangan teknologi telekomunikasi saat ini, jaringan broadband berkecepatan tinggi, dan internet, maka perkantoran telah menjadi elektronis secara elektronik. Pekerjaan kantor yang selama beberapa dekade dilakukan secara manual telah berUbah menjadi elektronik bahkan beberapa sudah menerapkan kantor maya. Kantor elektronis menggunakan teknologi komunikasi untuk menyediakan layanan *Admini*strasi perkantoran secara elektronis kepada siapa saja, dimana saja, dan tiap saat.

Perkantoran elektronik, atau e-Office, adalah istilah yang diciptakan untuk menutupi meningkatnya penggunaan komputer berbasis teknologi informasi untuk pekerjaan kantor, terutama di tahun 1980-an . Istilah muncul banyak sebelumnya dalam nama komputer LEO (Lyons Electronic Office), yang pertama berjalan dalam bisnis aplikasi pada tahun 1951 di Inggris.

paperless office adalah slogan, dimaksudkan untuk menggambarkan kantor masa depan. Ini difasilitasi dengan mempopulerkan terminal komputer layar video seperti 1964 IBM 2260. Penggunaan kertas kantor untuk lebih dari dua kali lipat 1980-2000 . Ini telah dikaitkan dengan peningkatan kemudahan produksi dokumen dan meluasnya penggunaan komunikasi elektronik, yang mengakibatkan pengguna menerima sejumlah besar dokumen yang sering dicetak. Namun, sejak sekitar pergantian milenium , penggunaan global kertas kantor telah mendatar dan sekarang menurun , yang telah dikaitkan dengan pergeseran generasi [7].

2.2.5 Pengertian *website*

Website merupakan kumpulan halaman-halaman yang digunakan untuk menampilkan informasi, gambar gerak, suara, dan atau gabungan dari semuanya itu baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing – masing dihubungkan dengan link – link3. Pada awalnya aplikasi web dibangun dengan hanya menggunakan bahasa yang disebut HTML (*HyperText Markup Langauge*). Pada perkembangan berikutnya, sejumlah script dan objek dikembangkan untuk memperluas kemampuan HTML seperti PHP dan ASP pada script dan Apllet pada objek.

Aplikasi Web dapat dibagi menjadi dua jenis yaitu aplikasi web statis dan dinamis [11].

2.2.6 *Unified Modeling Language* (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan men*document*tasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *Class* dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET [13].
Pada UML dikenal beberapa diagram, diantaranya sebagai berikut:

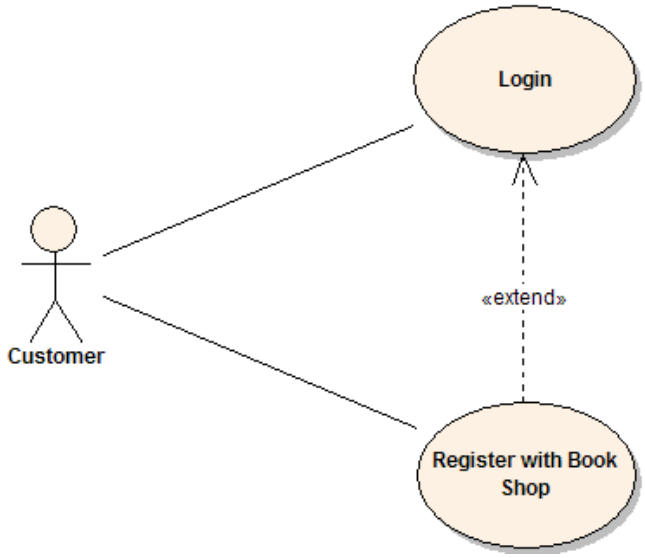
2.2.6.1 *Use Case* Diagram

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *Use Case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use Case* merupakan sebuah pekerjaan tertentu, misalnya *Login* ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use Case Diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem.

Sebuah *Use Case*dapat meng-includefungsionalitas *Use Case*lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *Use Case* yang di-include akan dipanggil setiap kali *Use Case* yang meng-include dieksekusi secara normal.

Sebuah *Use Case* dapat di-include oleh lebih dari satu *Use Case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah *Use Case* juga dapat meng-extend *Use Case* lain dengan behaviour-nya sendiri. Sementara hubungan generalisasi antar *Use Case* menunjukkan bahwa *Use Case* yang satu merupakan spesialisasi dari yang lain [13].



Gambar 2. 5 *Use Case* Diagram

2.2.6.2 *Sequence* Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

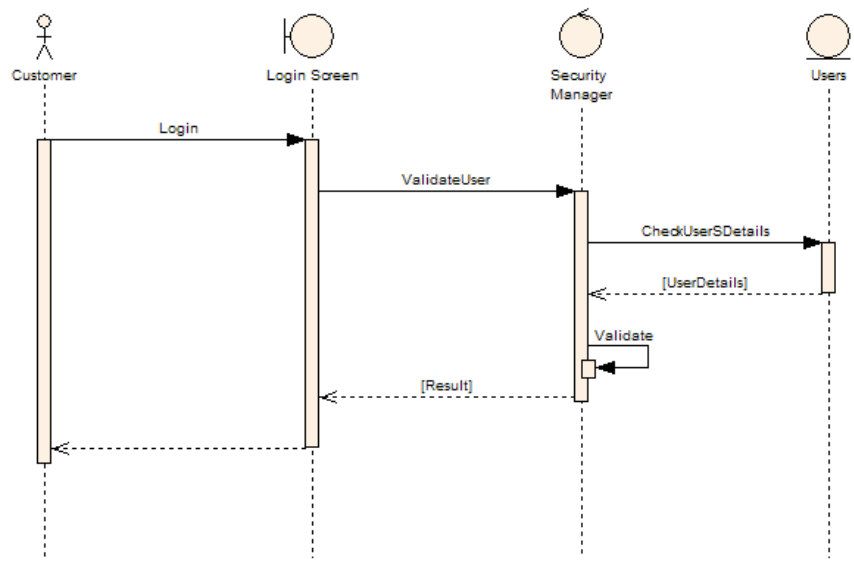
Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-triggeraktivitas tersebut, proses dan perUbahan apa saja yang terjadi secara internal dan outputapa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal.

Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi /

metoda dari *Class*. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity* [13].



Gambar 2. 6 *Sequence Diagram*

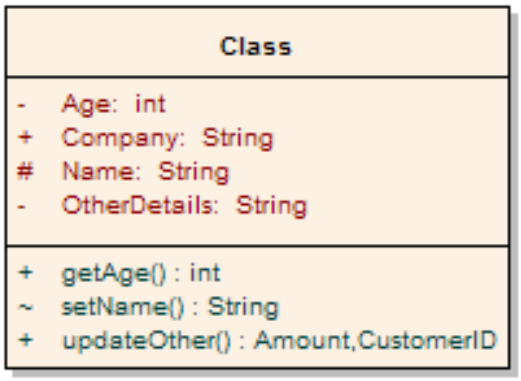
2.2.6.3 *Class* Diagram

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *Class*, packagedan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [13].

Hubungan antar *Class diagram* dapat dilihat dibawah ini :

1. Asosiasi, yaitu hubungan statis antar *Class*. Umumnya menggambarkan *Class* yang memiliki atribut berupa *Class* lain, atau *Class* yang harus mengetahui eksistensi *Class* lain. Panah *navigability* menunjukkan arah query antar *Class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *Class*. *Class* dapat diturunkan dari *Class* lain dan mewarisi semua atribut dan metoda *Class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *Class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-passing dari satu *Class* kepada *Class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *Sequence diagram* yang akan dijelaskan kemudian.



Gambar 2. 7 *Class Diagram*

2.2.6.4 *Activity* Diagram

Activity diagram smenggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decisionyang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagramjuga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

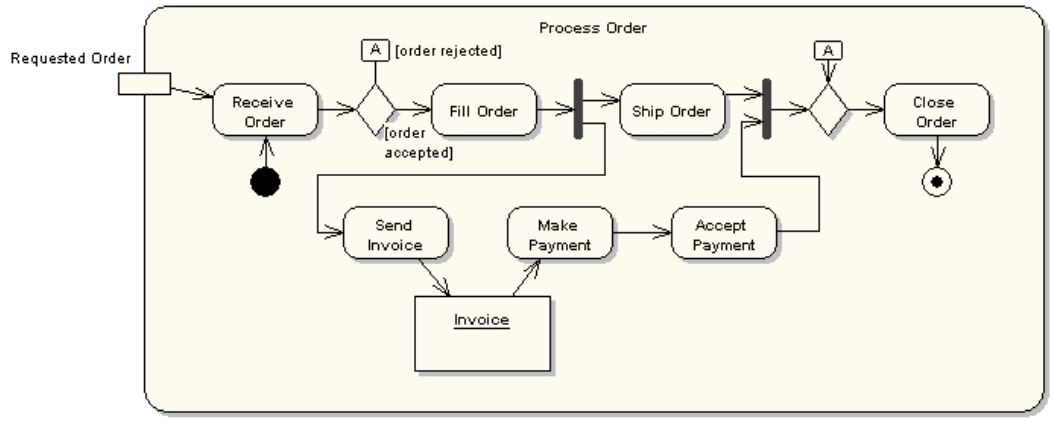
Activity diagram merupakan state diagram khusus, di mana sebagian besar statea dalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *Activity* diagramtidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem)

secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *Use Case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *Use Case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti state, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. Decision digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (fork dan join) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu [13].



Gambar 2. 8 *Activity Diagram*

2.2.7 Framework CodeIgniter dan Konsep Model View Controller (MVC)

Framework adalah sebuah struktur konseptual dasar yang digunakan untuk memecahkan sebuah permasalahan, bahkan isu-isu kompleks yang ada. Sebuah *framework* telah berisi sekumpulan arsitektur atau konsep-konsep yang dapat mempermudah dalam pemecahan sebuah permasalahan. *Framework* hanya

menjadi sebuah konstruksi dasar yang menopang sebuah konsep atau sistem yang bersifat *"essential support"* (penting tapi bukan komponen utama) [2].

CodeIgniter adalah sebuah *web application framework* yang bersifat *open source* digunakan untuk membangun aplikasi PHP dinamis. CodeIgniter sangat ringan, terstruktur, mudah dipelajari, *document*tasi lengkap dan dukungan yang luar biasa dari forum CodeIgniter. Selain itu CodeIgniter juga memiliki fitur-fitur lainnya yang sangat bermanfaat, antara lain:

1. Menggunakan *pattern* MVC.
   Dengan menggunakan *pattern* MVC ini, struktur kode yang dihasilkan menjadi lebih terstruktur dan memiliki standar yang jelas.
2. URL Friendly
   URL yang dihasilkan sangat *url friendly*. Pada CodeIgniter diminimalisasi penggunaan $_GET dan digantikan dengan URI.
3. Kemudahan
   Kemudahan dalam mempelajari, membuat *library* dan *helper*, memodifikasi serta meng-integrasikan *library* dan *helper*.

Jika membandingkan antara CodeIgniter dengan *framework-framework* lainnya maka beberapa poin yang membuat CodeIgniter unggul adalah:

1. Kecepatan
   Berdasarkan hasil *benchmark* CodeIgniter merupakan salah satu *framework* PHP tercepat yang ada saat ini.
2. Mudah dimodifikasi dan beradaptasi
   Sangat mudah memodifikasi *behavior framework* ini. Tidak membutuhkan *server requirement* yang macam-macam serta mudah mengadopsi *library* lainnya.
3. *Document*tasi lengkap dan jelas
   Bahkan tanpa buku ini pun CodeIgniter sebenarnya telah menyediakan sebuah panduan yang lengkap mengenai CodeIgniter. Semua informasi yang anda butuhkan tentang CodeIgniter ada disana.
4. *Learning Curve* Rendah

CodeIgniter sangat mudah dipelajari. Dalam pemilihan *framework* hal ini sangat penting diperhatikan karena kita harus juga memperhatikan skill dari seluruh anggota team. Jika sebuah *framework* sangat sulit dipelajari maka akan beresiko untuk memperlambat *team development*.

MVC adalah konsep dasar yang harus diketahui sebelum mengenal CodeIgniter. MVC adalah singkatan dari *Model View Controller*. MVC sebenarnya adalah sebuah *pattern* atau teknik pemrograman yang memisahkan *bisnis logic* (alur pikir), *data logic* (penyimpanan data) dan *presentation logic* (antarmuka aplikasi) atau secara sederhana adalah memisahkan antara desain, data dan proses [2].

Adapun komponen-komponen MVC antara lain:

1. Model

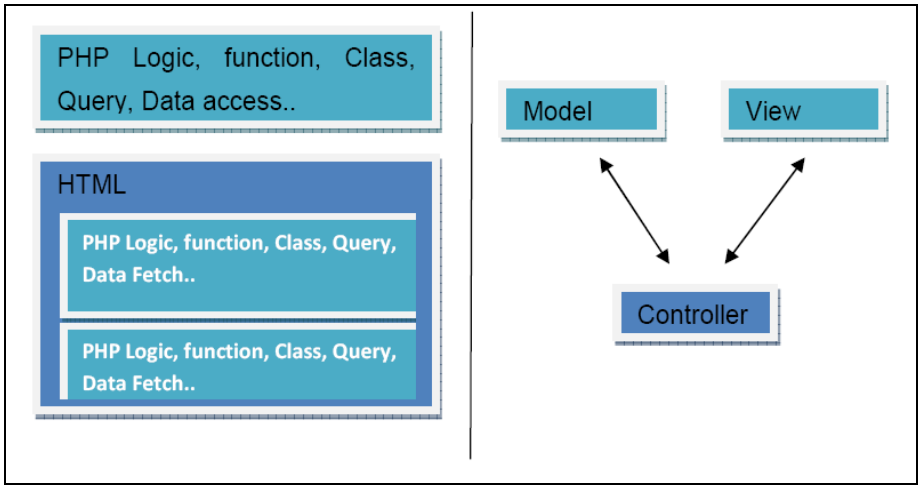
   Model berhubungan dengan data dan interaksi ke *Database* atau *webservice*. Model juga merepresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain. Biasanya di dalam model akan berisi *Class* dan fungsi untuk mengambil, melakukan *update* dan menghapus data *web*site. Sebuah aplikasi *web* biasanya menggunakan basis data dalam menyimpan data, maka pada bagian model biasanya akan berhubungan dengan perintah-perintah query SQL.

2. *View*

   View berhubungan dengan segala sesuatu yang akan ditampilkan ke *end-User*. Bisa berupa halaman *web*, rss, javascript dan lain-lain. Kita harus menghindari adanya logika atau pemrosesan data di *view*. Di dalam *view* hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. *View* dapat dikatakan sebagai halaman *website* yang akan dibuat dengan menggunakan HTML dan bantuan CSS atau JavaScript. Di dalam *view* jangan pernah ada kode untuk melakukan koneksi ke basisdata. *View* hanya dikhususkan untuk menampilkan data-data hasil dari model dan *controller*.

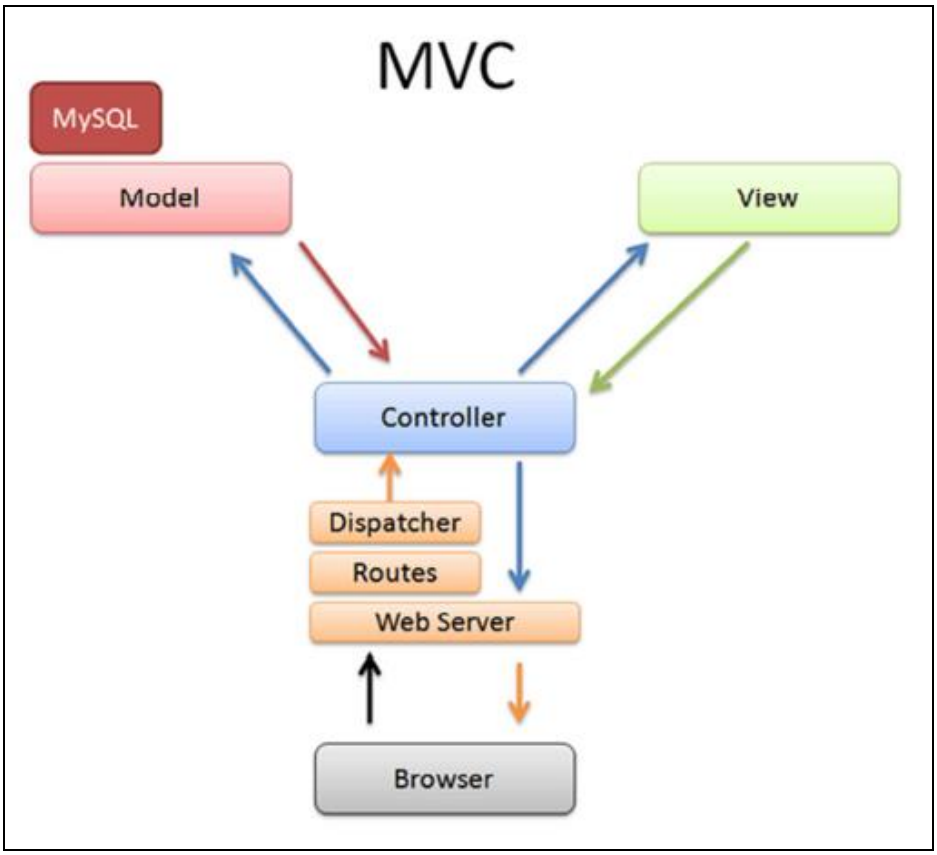
3. *Controller*

   Controller bertindak sebagai penghubung data dan *view*. Di dalam *controller* inilah terdapat *Class-Class* dan fungsi-fungsi yang memproses permintaan dari *view* ke dalam struktur data di dalam model. *Controller* juga tidak boleh berisi kode untuk mengakses basis data karena tugas mengakses data telah diserahkan kepada model. Tugas *controller* adalah menyediakan berbagai *variabel* yang akan ditampilkan di *view*, memanggil model untuk melakukan akses ke basis data, menyediakan penanganan kesalah atau *error*, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap input.



Gambar 2. 9 Perbandingan PHP Biasa Dengan CodeIgniter

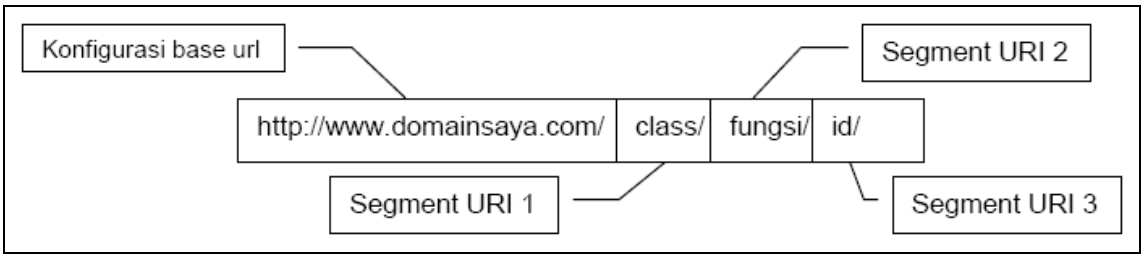
Pada gambar 2.9 , pada eksekusi PHP biasanya kita akan me-*load* semua *library* dan fungsi yang dibutuhkan kemudian digabungkan ke dalam HTML untuk di eksekusi oleh PHP. Untuk kasus sederhana cara tersebut masih baik-baik saja, tetapi ketika aplikasi tersebut menjadi kompleks atau rumit maka kita akan sulit memeliharanya jika tidak didukung oleh arsitektur *software* yang bagus. Hal tersebut bisa terjadi disebabkan oleh kode yang sama namun dibuat berulang-ulang, kode tidak konsisten dan lain-lain.



Gambar 2. 10 Alur Kerja CodeIgniter

Jika dipetakan, alur kerja CodeIgniter akan tampak seperti gambar II.10. *Browser* berinteraksi melalui *controller*. *Controller*-lah yang akan menerima dan membalas semua *request* dari *browser*. Untuk data maka *controller* akan meminta ke model dan untuk *User interface* atau *template* akan meminta ke *view*. Jadi "otak" dari aplikasi ada di *controller*, "muka" aplikasi ada di *view* dan "data" ada di model. Ketika *browser* meminta sebuah halaman *web* maka router akan mencarikan *controller* mana yang harus menangani request tersebut. Setelah itu barulah si *controller* menggunakan model untuk mengkases data dan *view* untuk menampilkan data tersebut.

Segmen-segmen pada URL pada CodeIgniter mencerminkan *controller* yang dipanggil. Contoh: http://www.domainsaya.com/*Class*/fungsi/id maka domain tersebut dapat dipecah menjadi bagian-bagian seperti pada gambar 2.11



Gambar 2. 11 Bagian – bagian URL pada CodeIgniter

Adapun bagian-bagian URL diatas adalah:

1. Konfigurasi *Base* URL

   Bagian base url merupakan url yang kita masukkan pada konfigurasi base_url yang merupakan url paling dasar untuk mengakses *web* atau aplikasi kita.

2. Segmen URL 1

   Segmen url 1 yaitu *Class*. *Class* tersebut merupakan nama kelas *controller* yang akan kita panggil. Apabila segmen ini kosong maka akan digantikan dengan *default controller* yang telah disetting di konfigurasi router.php.

3. Segmen URL 2

   Segmen url 2 yaitu fungsi dari *Class controller* yang telah kita panggil tadi. Apabila segmen kedua ini kosong maka fungsi yang dipanggil adalah fungsi index dari *Class controller* tersebut.

4. Segmen URL 3

   Segmen url 3 biasanya berisi parameter dari fungsi. Jika fungsi dari *controller* yang dipanggil mempunyai parameter maka parameternya harus dimasukkan sebagai segmen url sesuai urutan.

2.2.8 Metode TF/IDF (*Term Frequency* – Inversed *Document* Frequency)

Metode TF-IDF merupakan suatu cara untuk memberikan bobot hubungan suatu kata (*term)* terhadap dokumen (Robertson, 2005). Metode ini menggabungkan dua konsep untuk perhitungan bobot yaitu, frekuens i kemunculan sebuah kata didalam dokumen tertentu dan *inverse* frekuensi dokumen yang mengandung kata tersebut.

Pada algoritma *TF/IDF* digunakan rumus untuk menghitung bobot (W) masing-masing dokumen terhadap kata kunci dengan rumus yaitu:

$$W_{dt} = tf_{dt} * IDF_t \quad (2.1)$$

Dimana:

d = dokumen ke-d

t = kata ke-t dari kata kunci

W = bobot dokumen ke-d terhadap kata ke-t

tf = banyaknya kata yang dicari pada sebuah dokumen

IDF = *Inversed Document Frequency* (Jumlah dokumen yang terambil oleh Sistem)

IDF = $Log \frac{D}{df}$

D = total dokumen

df = banyak dokumen yang mengandung kata yang dicari

Setelah bobot (W) masing-masing dokumen diketahui, maka dilakukan proses *sorting*/pengurutan dimana semakin besar nilai W, semakin besar tingkat similaritas dokumen tersebut terhadap kata kunci, demikian sebaliknya.

2.2.9 Model Ruang Vektor (*Vector Space Model*)

Metode Ruang Vektor adalah suatu metode untuk merepresentasikan sistem temu kembali informasi ke dalam vektor dan memperhitungkan fungsi similarity dalam proses pencocokan beberapa vektor. Suatu sistem temu kembali informasi terdiri atas dua bagian, yaitu penyimpanan dokumen dan pemrosesan query. Baik query maupun dokumen-dokumen yang disimpan, dinyatakan dalam bentuk vector. Elemen vektor tersebut adalah hasil dari pembotan kata (*term*) pada dokumen dan query [6].

Metode Ruang Vektor adalah suatu metode untuk merepresentasikan sistem temu kembali informasi. Penentuan relevansi dokumen dengan query dipandang sebagai pengukuran kesamaan (*similarity measure*) antara vektor dokumen

dengan vektor query. Semakin sama suatu vektor dokumen dengan vektor query maka dokumen dapat dipandang semakin relevan dengan query.

Pada Algoritma *Vector Space Model* digunakan rumus untuk mencari nilai cosinus sudut antara dua *Vector* dari setiap bobot dokumen (WD) dan bobot dari kata kunci (WK). Rumus yang digunakan adalah sebagai berikut

$$Cosine \rightarrow sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t}(w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}} \quad (2.2)$$

Rumus diatas digunakan untuk mendapatkan nilai cosinus minimal = 0 dan maksimalnya = 1.