BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Landasan Teori

2.2.1 Sistem

Kata sistem berasal dari bahasa Yunani, yaitu *systema*, yang artinya himpunan bagian atau komponen yang saling berhubungan secara teratur dan merupakan suatu keseluruhan. Selain itu, bisa diartikan sekelompok elemen yang independen, namun saling terkait sebagai satu kesatuan. Sistem terdiri atas struktur dan proses. Struktur sistem merupakan unsur-unsur yang membentuk sistem tersebut, sedangkan proses sistem menjelaskan cara kerja setiap unsur sistem dalam mencapai tujuan. Setiap sistem merupakan bagian dari sistem lain yang lebih besar dan terdiri atas berbagai sistem yang lebih kecil, yang disebut subsistem. Setiap sistem diciptakan untuk menangani sesuatu yang berulang-ulang atau yang secara rutin terjadi[6].

Karakteristik sistem menurut *Edhi Sutanta (2003),* yaitu sebagai berikut.

1. Komponen (*components*)

   Komponen sistem adalah segala sesuatu yang menjadi bagian penyusunan sistem. Komponen sistem dapat berupa benda nyata ataupun abstrak. Komponen sistem disebut sebagai subsistem.

2. Batas (*boundary*)

   Batas sistem diperlukan untuk membedakan satu sistem dengan sistem yang lain. Tanpa adanya batas sistem, sangat sulit untuk memberikan batasan scope tinjauan terhadap sistem.

3. Lingkungan (*environments*)

   Lingkungan sistem adalah segala sesuatu yang berada di luar sistem lingkungan sistem yang dapat menguntungkan ataupun merugikan. Umumnya lingkungan yang menguntungkan akan selalu diper- tahankan untuk menjaga keberlangsungan sistem, sedangkan lingkungan sistem yang merugikan akan diupayakan agar mempunyai pengaruh seminimal mungkin, bahkan ditiadakan.

4. Penghubung/antarmuka (*interface*)

   Penghubung/antarmuka merupakan sarana memungkinkan setiap komponen sistem, yaitu segala sesuatu yang bertugas men- jembatani hubungan antar komponen dalam sistem. Penghubung/antarmuka merupakan sarana setiap komponen saling berinteraksi dan berkomunikasi.

5. Masukan (*input*)

   Masukan merupakan komponen sistem, yaitu segala sesuatu yang perlu dimasukan ke dalam sistem sebagai bahan yang akan diolah lebih lanjut untuk menghasilkan keluaran (output) yang berguna.

6. Pengolahan (*processing*)

   Pengolahan merupakan komponen sistem yang mempunyai peran utama mengolah masukan agar menghasilkan output yang berguna bagi para pemakainya.

7. Keluaran (*output*)

   Keluaran merupakan komponen sistem yang berupa berbagai macam bentuk keluaran yang dihasilkan oleh komponen pengolahan.

8. Sasaran (*objectives*) dan tujuan (*goal*)

   Setiap komponen dalam sistem perlu dijaga agar saling bekerja sama agar mampu mencapai sasaran dan tujuan sistem.

9. Kendali (*control*)

   Setiap komponen dalam sistem perlu dijaga agar tetap bekerja sesuai dengan peran dan fungsinya masing-masing.

10. Umpan balik (*feedback*)

    Umpan balik diperlukan oleh bagian kendali (*Control*) sistem untuk mengecek terjadinya penyimpanan proses dalam sistem dan mengembalikannya pada kondisi normal.

2.2.2 Sistem Informasi

Sistem informasi adalah sekumpulan hardware, software, brainware, prosedur, dan/atau aturan yang diorganisasikan secara integral untuk mengolah data menjadi informasi yang bermanfaat guna memecahkan masalah dan pengambilan keputusan. Sistem informasi adalah satu ke-satuan data olahan yang terintegrasi dan saling melengkapi yang meng-hasilkan data olahan, baik dalam bentuk gambar, suara maupun tulisan[6].

Sistem informasi menyediakan tiga macam tipe informasi, yaitu sebagai berikut:

1. Informasi pengumpulan data adalah sebuah Informasi berupa akumulasi atau pengumpulan data untuk menjawab pertanyaan, berguna bagi manajer bawah untuk mengevaluasi kinerja personel-personelnya.

2. Informasi pengarahan perhatian merupakan Informasi untuk membantu manajemen memusatkan perhatian pada masalah-masalah yang menyimpang, ketidakberesan. Informasi ini membantu manajemen menengah untuk melihat penyimpangan yang terjadi.
3. Informasi pemecahan masalah yaitu Informasi yang membantu para manajer atas mengambil keputusan memecahkan permasalahan yang dihadapinya. Problem solving dihubungkan dengan keputusan yang tidak berulang-ulang serta situasi yang membutuhkan analisis yang dilakukan oleh manajemen tingkat atas.

2.2.3 Rekayasa Web

Rekayasa Web adalah sebuah aplikasi yang menggunakan pendekatan sistematis, disiplin, dan terukur untuk pengembangan, operasi dan pemeliharaan aplikasi berbasis Web (Web-based applications)[7]. Rekayasa Web adalah subdisiplin dari rekayasa perangkat lunak yang membantu menyediakan metodologi untuk merancang, mengembangkan, memelihara, dan melibatkan aplikasi web. Rekayasa Web membantu para pengembang sistem di bawah kontrol, memperkecil risiko-risiko yang akan terjadi dan meningkatkan kualitas, dapat dipelihara, dan memiliki skalabilitas aplikasi Web[8].

2.2.4 Metode Pengembangan Sistem

Dalam pengembangan perangkat lunak pastinya melewati beberapa tahapan pengembangan, mulai dari perencanaan sampai dengan implementasi. Bila perangkat lunak atau sistem tersebut sudah diimplementasikan pada perkembangannya timbul permasalahan terkait sistem tersebut, maka perlu dikembangkan untuk dapat mengatasi permasalahan baru tersebut. Siklus ini disebut dengan siklus hidup pengembangan sistem (system development life cycle/SDLC)[8]. Siklus hidup pengembangan sistem dimulai dari tahap perencanaan, tahap pengembangan (mulai dari investigasi, analisis, desain sampai dengan implementasi) dan tahap evaluasi. Pada tahap evaluasi ini akan dilakukan secara terus menerus untuk memastikan bahwa sistem yang diimplementasikan berjalan dengan lancar dan tidak terjadi permasalahan. Jika terjadi suatu permasalahan maka sistem informasi akan dikembangkan kembali atau diganti dengan sistem yang baru. Pengembangan sistem yang baru ataupun pengembangannya ini akan dimulai dari tahap perencanaan kembali.

Berikut ini merupakan penjelasan dari siklus hidup pengembangan sistem:

a. Tahap Perencanaan

Perencanaan pengembangan sistem pada dasarnya bertujuan untuk mengetahui clan mengedepankan sistem yang akan dikembangkan. Target apa saja yang di dicapai, lama

pengerjaan, serta mempertimbangkan kebutuhan dana yang digunakan dalam pengembangan sistem. Pada tahap perencanaan ini juga mendefinisikan tentang masalah bisnis yang dihadapi oleh organisasi, perihal apa saja yang kiranya menghambat laju kembangnya bisnis.

b. Tahap *Analysis*

Pada tahapan analisis ini mengumpulkan kebutuhan-kebutuhan bisnis apa saja dari berbagai stakeholders yang berkepentingan. Mendefinisikan kebutuhan-kebutuhan bisnis yang akan ditindaklanjuti guna diakomodir oleh sistem yang akan dibangun. Membuat prototype sederhana untuk memenuhi kebutuhan• kebutuhan tersebut. Melakukan prioritas terhadap kebutuhan-kebutuhan bisnis yang telah didefinisikan. Membuat clan mengevaluasi alternatif - alternatif solusi untuk masing-masing kebutuhan tersebut. Mengulas rekomendasi - rekomendasi solusi dengan pihak manajemen.

c. Design

Melakukan perancangan dengan mengintegrasikannya dengan jaringan. Merancang arsitektur aplikasi yang akan digunakan. Merancang tampilan layar untuk pengguna. Merancang system interfaces. Merancang clan mengintegrasikan sistem dengan database. Membuat *prototypes* untuk detil-detil perancangan. Membuat clan mengintegrasikan sistem dengan system control.

d. Implementation

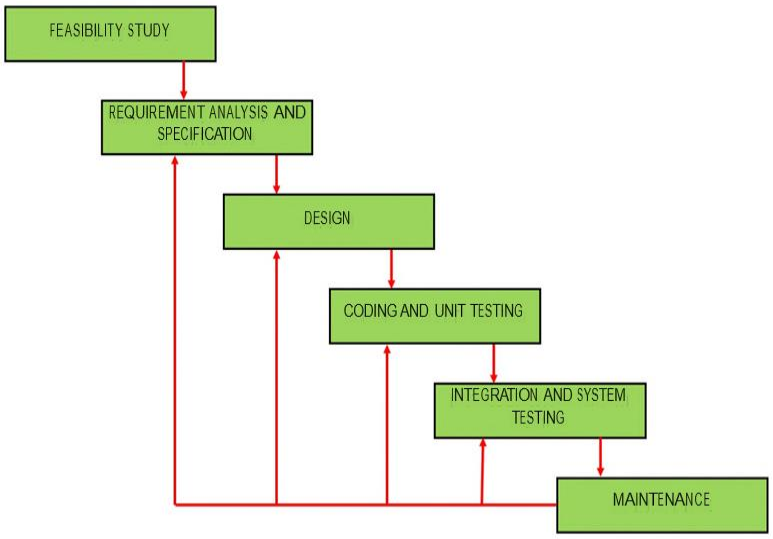
Membangun komponen-komponen perangkat lunak. Melakukan verifikasi clan uji coba terhadap sistem yang telah selesai dibangun. Melakukan konversi data. Melatih para pengguna untuk berinteraksi serta menyelesaikan tugas kerjanya dengan menggunakan sistem. Membuat dokumentasi terhadap sistem yang telah selesai dibangun, yang dapat berupa manual book. Meng-install sistem di terminal-terminal PC yang membutuhkan.

e. Activities

Melakukan pemeliharaan sistem dengan pengecekan secara berkala/periodik. Memperkaya atau mengembangkan sistem dengan penambahan fitur-fitur baru yang dapat meningkatkan kinerja kerja user guna mendukung kinerja bisnis. Memberikan pelayanan kepada para users, seperti dalam bentuk call center ataupun IT support.

Model *Waterfall* adalah proses pengembangan perangkat lunak berurutan, yang prosesnya terus mengalir ke bawah (seperti air terjun) melewati fase-fase perencanaan, pemodelan, implementasi (konstruksi), dan pengujian. Sedangkan metode *Iterative waterfall* adalah pengembangan dari model *waterfall* yang memberikan jalur umpan balik / perubahan pada setiap

fase ke fase sebelumnya. Dapat dilihat pada gambar 2.1 fase sekuensial dari model Iterative waterfall.



Gambar 2. 1 Model Iterative waterfall (PAL, 2018)

2.2.5 Pemrograman Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat suatu permasalahan dan sistem baik (sistem perangkat lunak, sistem informasi atau sistem lainnya)[6]. Ada banyak cara untuk melakukan pengabstraksian dan memodelkan objek, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat melakukan pengabstraksian dan memodelkan objek, data dan proses yang dipunyai akan dienkapsulasi menjadi suatu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan sesuai dengan aturan dan alat bantu pemodelan tertentu. Sistem berorientasi objek merupakan sebuah sistem yang komponennya dibungkus atau dienkapsulasi menjadi kelompok data dan fungsi. Setiap komponen ini nantinya dalam suatu sistem akan mewarisi atribut dan sifat dari komponen lain dan dapat berinteraksi satu sama lain.

Berikut adalah konsep dasar yang harus dipahami terkait pemrograman berbasis objek:

1. Kelas (*class*)

   Kelas adalah kumpulan objek dengan karakteristik yang sama. Sebuah kelas akan memiliki sifat atau atribut, metode atau operasi, hubungan dan arti. Suatu kelas dapat diturunkan dalam kelas lain, dimana atribut dan kelas dapat diwariskan pada kelas yang baru. Secara teknis, kelas

merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. objek (*object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia atau hal lainnya. Objek merupakan suatu entitas yang mampu menyimpan informasi dan operasi yang dapat berpengaruh pada status objeknya. Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat suatu objek.

3. Metode (*method*)

Metode atau operasi pada suatu kelas hampir sama dengan fungsi pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode. Metode atau operasi dapat berasal dari *event*, aktivitas atau aksi keadaan dan fungsi.

4. Atribut (*attribute*)

Atribut adalah suatu variabel yang dimiliki oleh sebuah kelas. Atribut dapat berupa nilai atau elemen data yang dimiliki objek dalam suatu kelas. Atribut ini dimiliki secara individual oleh sebuah objek dan sebaiknya bersifat privat untuk menjaga konsep enkapsulasi.

5. Abstraksi (*abstraction*)

Prinsip untuk merepresentasi dunia nyata yang komplek menjadi satu bentuk model yang sederhana dengan mengabaikan aspek lainnya yang tidak sesuai dengan masalah.

6. Enkapsulasi (*encapsulation*)

Pembungkusan atribut data dan layanan untuk menyembunyikan implementasi sehingga objek tidak mengetahui cara kerjanya.

7. Pewarisan (*inheritance*)

Mekanisme yang memungkinkan objek mewarisi sebagian atau seluruh definisi dan objek lainnya.

8. Antarmuka (*interface*)

Antarmuka sangat mirip dengan kelas, tetapi tanpa atribut kelas memiliki metode yang dideklarasikan tanpa ada isi. Deklarasi metode sebuah antarmuka dapat diimplementasi oleh kelas lainnya. Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas akan mendeklarasikan metode antarmuka yang dibutuhkan oleh kelas itu. Antarmuka atau *interface* biasanya digunakan agar kelas lain tidak mengakses langsung ke suatu kelas.

9. *Reusability*

Pemanfaatan kembali suatu objek sudah didefinisikan untuk suatu masalah yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan objek dan kelas.

11. 1Komunikasi Antar Objek

Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dari suatu objek ke objek yang lainnya.

12. Polimorfisme (*polymorphism*)

Kemampuan objek yang digunakan untuk banyak tujuan yang berbeda dengan nama yang sama sehingga dapat menghemat baris program.

13. *Package*

Package adalah sebuah kemasan yang digunakan untuk mengelompokan kelas sehingga memungkinkan kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.2.6 UML (*Unified Modelling Language)*

Pada perkembangan teknik pemrograman berorientasi objek, Sebuah standarisasi bahasa pemodelan untuk membangun sebuah perangkat lunak, yaitu *Unified Modelling Language* (UML)[6]. UML merupakan bahasa visual untuk komunikasi suatu sistem dengan menggunakan diagram dan teks pendukung. UML hanya berfungsi untuk melakukan pemodelan, jadi penggunaannya tidak terbatas, meski pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

UML terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori:

A. *Structure diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
B. *Behavior diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
C. *Interaction diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

1. *Use case Diagram*

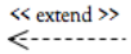
Use case Diagram merupakan pemodelan untuk kelakuan suatu sistem yang dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang dibuat dan digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi tersebut. Syarat penamaan pada *use case* adalah nama didefinisikan sesederhana mungkin dan dapat dipahami. Ada dua hal utama dalam *use case*, yaitu [8]:

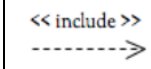
A. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dalam sistem yang akan dibuat diluar sistem itu sendiri, jadi walaupun simbol aktor adalah orang tetapi aktor itu belum tentu orang.

B. *Use case* merupakan fungsionalitas yang ada pada sistem sebagai unit untuk bertukar pesan atau berkomunikasi.

Berikut adalah simbol-simbol yang ada pada *use case diagram*, sesuai pada Tabel 2.1 :

Tabel 2. 1 Simbol-simbol Use case

No.	Simbol	Nama	Keterangan
1.		*Use case*	Disediakan sistem sebagai unit yang saling bertukar pesan antar aktor. Biasanya dinyatakan dengan kata kerja.
2.		*Actor*	Merupakan orang, sistem, ataupun proses yang berinteraksi dengan sistem yang akan dibuat diluar dari sistem itu sendiri.
3.		*Association*	Komunikasi antara aktor dengan *use case* atau *use case* yang memiliki interaksi dengan aktor.
4.		*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* dapat berdiri sendiri tanpa *use case* tambahan.
5.		*Generalization*	Hubungan antara dua buah *use case* dimana fungsi yang satu lebih umum dari lainnya.

6.		*Include*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya.

2.2.7 *Flowchart*

Flowchart adalah penggambaran secara grafik dari langkah- langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. Kegunaan utama dari *Flowchart* yaitu untuk mendesain program dan merepresentasikan program[9].

Flowchart harus dapat merepresentasikan komponen - komponen dalam bahasa pemrograman seperti:

a. *Relationship*

   Flowchart dapat memberikan gambaran yang efektif, jelas, dan ringkas tentang prosedur *logic*. Teknik penyajian yang bersifat grafis jelas akan lebih baik daripada uraian-uraian yang bersifat teks khususnya dalam menyajikan logika-logika yang bersifat kompleks.

b. *Analysis*

   Dengan adanya pengungkapan yang jelas dalam model atau chart, maka para pembaca dapat dengan mudah melihat permasalahan atau memfokuskan perhatian pada area-area tertentu sistem informasi.

c. *Communication*

   Karena simbol-simbol yang digunakan mengikuti suatu standar tertentu yang sudah diakui secara umum, maka *flowchart* dapat merupakan alat bantu yang sangat efektif dalam mengkomunikasikan logika suatu masalah atau dalam mendokumentasikan logika tersebut

Flowchart disusun dengan simbol. Simbol ini bertujuan untuk membantu dalam penggambaran proses didalam program dan menunjukan jenis operasi pengolahan dalam suatu proses. Berikut adalah simbol-simbol yang ada pada *flowchart*[8], dapat dilihat pada Tabel 2.2 :

Tabel 2. 2 Simbol-simbol *Flowchart*

No.	Simbol	Nama	Keterangan
1.		*Flow*	Menyatakan jalannya arus suatu proses
2.		*Communication Link*	Menyatakan bahwa adanya transisi suatu data dari satu lokasi ke lokasi lainnya.
3.		*Connector*	Menyatakan sambungan dari satu proses ke proses lainnya dalam halaman yang sama.
4.		*Offline Connector*	Menyatakan sambungan dari satu proses ke proses lainnya dalam halaman yang berbeda.
5.		*Manual Operation*	Menyatakan suatu tindakan yang tidak dilakukan oleh komputer.
6.		*Decision*	Menunjukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, ya atau tidak.
7.		*Predefined Proses*	Menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.
8.		*Terminal*	Menyatakan permulaan atau akhir suatu program
9.		*Keying Operation*	Menyatakan segala jenis operasi yang diproses dengan menggunakan mesin yang mempunyai keyboard.
10.		*Off-line Storage*	Menunjukan bahwa data disimpan ke suatu media tertentu.

11.		*Manual Input*	Memasukan data secara manual dengan menggunakan online keyboard.
12.		*Input – Output*	Menyatakan proses input dan output.
13.		*Punched Card*	Menyatakan input yang berasal dari kartu atau output ditulis ke kartu.
14.		*Magnetic-tape Unit*	Menyatakan input yang berasal dari pita magnetic atau output disimpan ke pita magnetic.
15.		*Document*	Merupakan simbol untuk data yang terbentuk informasi.
16.		*Display*	Menyatakan peralatan output yang digunakan berupa video atau komputer (layar).

2.2.8 Basis Data

Basis data merupakan kumpulan informasi yang ada selama periode waktu yang lama, seringkali bertahun-tahun[10]. Basis data merupakan hal yang sangat penting untuk semua bisnis. Basis data berada di belakang layar perusahaan besar maupun perusahaan kecil. Perusahaan menyimpan setiap data – data penting mereka ke dalam basis data. Kekuatan basis data berasal dari pengetahuan dan teknologi yang telah berkembang dan diwujudkan dalam perangkat lunak khusus yang disebut sistem manajemen basis data, atau Database Management System (DBMS), atau disebut juga Sistem Basis Data.

A. *Database Management System* (DBMS)

DBMS (*Database Management System*) adalah sistem aplikasi untuk menyimpan, mengelola dan menampilkan data. Suatu sistem dikatakan DBMS jika memenuhi syarat jika:

1. Menyediakan fasilitas untuk mengelola data.
2. Mampu menangani integritas data.
3. Mampu menangani akses data.

4. Mampu menangani *backup* data.

Pentingnya data bagi suatu organisasi, maka hampir sebagian besar memanfaatkan DBMS dalam mengelola data yang dimiliki. Pengelolaan DBMS ditangani oleh tenaga ahli yaitu DBA (*Database Administrator*).

B. *Entity Relationship Diagram* (ERD)

Pemodelan basis data yang paling banyak digunakan adalah *Entity Relationship Diagram* (ERD). Dalam entity relationship model (atau model E / R), struktur data direpresentasikan secara grafis, sebagai "Entity Relationship Diagram (ERD)" menggunakan tiga prinsip jenis elemen yaitu set entitas, atribut, dan relasi(hubungan).

Berikut adalah simbol-simbol yang ada pada ERD dengan notasi Chen [10], dapat dilihat pada Tabel 2.3 :

Tabel 2. 3 Simbol-simbol ERD

No.	Simbol	Nama	Keterangan
1.		*Entity*	Merupakan data inti yang akan disimpan, penamaan entitas biasanya lebih ke kata benda.
2.		Atribut	*Field* atau kolom yang disimpan dalam entitas.
3.		Atribut kunci primer	*Field* atau kolom yang disimpan dalam entitas dan sebagai kunci akses.
4.		*Multivalue*	*Field* atau kolom yang disimpan dalam entitas yang memiliki nilai lebih dari satu.
5.		Relasi	Relasi menghubungkan antar entitas. Biasanya diawali dengan kata kerja.
6.		*Association*	Penghubung antar relasi dan entitas dimana ujungnya memiliki *multiplicity*.

2.2.9 Jalan Kabupaten

Jalan Kabupaten adalah status atau kondisi jalan yang menjadi tanggung jawab Pemerintah Kabupaten[3]. Jalan Kabupaten umumnya terletak di Wilayah Kabupaten dan menghubungkan antara Desa, Kecamatan, atau Kota dalam Wilayah Kabupaten tersebut[4].

2.2.10 Pemeliharaan Rutin Jalan

Pemeliharaan rutin jalan adalah kegiatan merawat serta memperbaiki kerusakan-kerusakan yang terjadi pada ruas-ruas jalan dengan kondisi pelayanan mantap[5]. Jalan dengan kondisi pelayanan mantap adalah ruas-ruas jalan dengan kondisi baik atau sedang sesuai umur rencana yang diperhitungkan serta mengikuti suatu standar tertentu. Kegiatan pemeliharaan rutin jalan dilakukan pada ruas jalan/bagian ruas jalan dan bangunan pelengkap dengan kriteria sebagai berikut:

1. Ruas jalan dengan kondisi baik dan sedang atau disebut jalan mantap.
2. Bangunan pelengkap jalan yang mempunyai kondisi baik sekali dan baik.

Pemeliharaan rutin jalan dilakukan sepanjang tahun, meliputi kegiatan:

1. Pemeliharaan/pembersihan bahu jalan;
2. Pemeliharaan sistem drainase (dengan tujuan untuk memelihara fungsi dan untuk memperkecil kerusakan pada struktur atau permukaan jalan dan harus dibersihkan terus menerus dari lumpur, tumpukan kotoran, dan sampah);
3. Pemeliharaan/pembersihan rumaja;
4. Pemeliharaan pemotongan tumbuhan/tanaman liar (rumput rumputan, semak belukar, dan pepohonan) di dalam rumija;
5. Pengisian celah/retak permukaan (sealing);
6. Laburan aspal;
7. Penambalan lubang;
8. Pemeliharaan bangunan pelengkap;
9. Pemeliharaan perlengkapan jalan; dan
10. Pembentukan kembali permukaan untuk perkerasan jalan tanpa penutup dan jalan tanpa perkerasan.

~ Halaman ini sengaja dikosongkan ~