BAB II

LANDASAN TEORI

2.1. Profil Kesehatan Kabupaten Bandung

Profil Kesehatan Kabupaten Bandung merupakan sarana penyedia data dan informasi dalam rangka evaluasi tahunan kegiatan-kegiatan dan pemantauan pencapaian Indikator Standar Pelayanan Minimal Bidang Kesehatan, Indeks Kesehatan dan juga sebagai dasar penyusunan rencana pembangunan daerah untuk tahun berikutnya.

Mengingat pentingnya manfaat dari pembuatan Profil Kesehatan Kabupaten maka Dinas Kesehatan Kabupaten Bandung berupaya untuk menyusun "Buku Profil Kesehatan Kabupaten Bandung 2014".

2.1.1 Tujuan

Adapun tujuan dari dinas kesehatan kabupaten bandung adalah :

2.1.1.1 Tujuan Umum

Memberikan gambaran kesehatan yang menyeluruh di wilayah Kabupaten Bandung dalam rangka evaluasi dan pemantauan pencapaian Kabupaten Sehat.

2.1.1.2 Tujuan Khusus

1. Menyediakan data dan informasi umum Kabupaten yang meliputi data lingkungan fisik atau biologic, data perilaku kesehatan masyarakat, data demografik dan data sosial ekonomi.
2. Menyediakan data dan informasi pencapaian pembangunan kesehatan di kabupaten bandung meliputi indicator-indikator derajat kesehatan, perilaku manusia, kesehatan lingkungan dan sumber daya kesehatan.
3. Menyediakan data dan informasi kegiatan-kegiatan multi sector yang dilakukan dalam rangka mencapai kabupaten bandung sehat.
4. Menyediakan data dan informasi untuk penyusunan Profil Kesehatan Propinsi Jawa Barat dan profil Kesehatan Nasional.

2.1.2 Visi Misi Dinas Kesehatan

Mengacu pada visi dan misi kabupaten bandung, gambaran kabupaten di masa depan yang ingin di capai melalui pembangunan kesehatan diajabarkan dalam visi pembangunan kesehatan kabupaten bandung yaitu "terwujudnya masyarakat kabupaten bandung yang sehat mandiri melalui akselerasi indeks kesehatan 78 pada tahun 2015" untuk mencapai visi tersebut ditetapkan misi berikut :

1) Meningkatkan kinerja melalui pemantapan manajemen kesehatan yang dinamis dan akuntabel.
2) Meningkatkan fungsi regulasi dalam menunjang pelayanan kesehatan yang bermutu.
3) Menggerakkan pembangunan berwawasan kesehatan dalam mencapai masyarakat kabupaten bandung sehat mandiri.

2.1.3 Logo Instansi

Berikut adalah Gambar II.1 Logo Dinas Kesehatan Kabupaten Bandung merupakan logo Dinas Kesehatan Kabupaten Bandung :

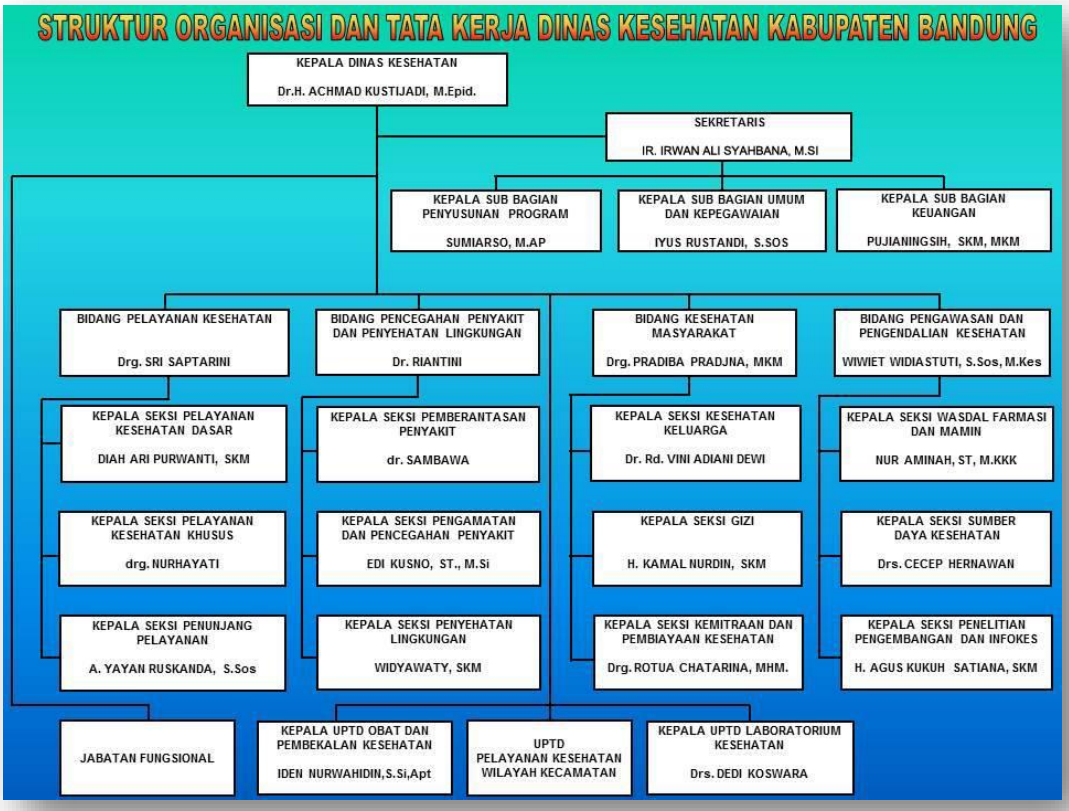


Gambar II.1 Logo Dinas Kesehatan Kabupaten Bandung

2.1.4 Struktur Organisasi Dinas Kesehatan Kabupaten Bandung

Dalam menjalankan tugas-tugas yang terdapat di dalam dinas kesehatan diperlukan suatu organisasi yang tepat, sehingga akan jelas tanggung jawab dan wewenang masing - masing individu. Organisasi merupakan suatu proses kepemimpinan, sehingga menciptakan suatu kerjasama yang baik dan memungkinkan adanya koordinasi dan pengawasan sebaik mungkin.

Berikut adalah Gambar II.2 Struktur Organisasi Dinas Kesehatan merupakan Struktur Organisasi Dinas Kesehatan Kabupaten Bandung :



Gambar II.2 Struktur Organisasi Dinas Kesehatan

2.2. *Landasan Teori*

2.2.1 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis *Linux*. *Android* menyediakan *platform* terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. *Android*

pertama kali dikembangkan oleh perusahaan bernama *Android Inc* yang kemudian pada tahun 2005 di akuisisi oleh raksasa Internet *Google* [***5***]. *Android* dibuat dengan basis *kernel linux* yang telah dimodifikasi, dan untuk setiap release-nya diberi kode nama berdasarkan nama hidangan makanan. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google*, *HTC*, Intel, *Motorola*, *Qualcomm*, *T-mobile*, dan *Nvidia*.

Keunggulan utama *Android* adalah gratis dan *open source*, yang membuat *smartphone Android* dijual lebih murah dibandingkan dengan *blackberry* atau *iPhone* meski fitur (*hardware*) yang ditawarkan *Android* lebih baik. Beberapa fitur utama dari *Android* antara lain *WiFi hotspot*, *Multi-touch*, *Multitasking*, *GPS*, *support java*, mendukung banyak jaringan *(GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth,* Wi-Fi, *LTE*, and *WiMAX*) dan juga kemampuan dasar handphone pada umumnya [***5***]. Di dunia ini terdapat dua jenis distributor sistem operasi *Android*. Pertama yang mendapat dukungan penuh dari *Google* atau *Google Mail Serv*ices (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung *Google* atau dikenal sebagai *Open Handset Distribution* (OHD). Dengan *platform Android* inilah aplikasi rujukan dibuat.

Pemanfaatan sistem operasi *android* dianggap oleh peneliti cocok sebagai media pengembangan aplikasi ini, dikarenakan banyaknya *smartphone* yang beredar dikalangan masyarakat menggunakan aplikasi ini. Sistem operasi *mobile* sudah cukup umum di gunakan oleh kalangan muda, dan tua. Maka atas dasar pemikiran yang telah dipaparkan diatas peneliti memutuskan untuk menggunakan sistem operasi ini sebagai wadah aplikasi yang akan dibangun.



Gambar II.3 *Logo Android* [*5*]

2.2.2 JSON (Java Script Object Notation)

JSON adalah standar berbasis teks ringan yang diracang terbuka dan mudah untuk dibaca dan ditulis manusia serta mudah bagi mesin untuk mengurai dan menghasilkan dalam pertukaran data. *JSON* merupakan format *text* bahasa independen tetapi menggunakan konvensi bahasa yang akrab dari bahasa C, meliputi C, C++, C#, *Java*, *Java*Script, Perl, Python, dan lainnya [***6***]. Properti ini yang membuat *JSON* sebagai bahasa pertukaran yang ideal. Format *JSON* awalnya ditentukan oleh Douglas Crockford dan dijelaskan dalam RFC4627. Jenis media internet yang resmi untuk *JSON* adalah aplikasi/*JSON*. Format *JSON* sering digunakan untuk serialisasi dan mengirimkan data terstruktur melalui koneksi jaringan. Terutama untuk pengiriman data antara *server* dan aplikasi *web*, melayani sebagai alternative ke XML. Dengan *JSON* pertukaran data dari *webserv*ice ke aplikasi *mobile* menjadi lebih ringan dalam menampilkan data. [***6***]

2.2.3 Google Maps API

Google Maps API merupakan pengembangan teknologi dari *google* yang digunakan untuk menanamkan *Google Maps* di suatu aplikasi yang tidak dibuat oleh *Google*. API itu sendiri adalah Kumpulan fungsi-fungsi atau perintah-perintah untuk menggantikan bahasa yang digunakan dalam *system call* dengan bahasa yang lebih terstruktur dan mudah dimengerti [***7***].

Google Maps API adalah suatu library yang berbentuk *javascript* yang berguna untuk memodifikasi peta yang ada di *Google Maps* sesuai kebutuhan. Dalam perkembangannya *Google Maps API* diberikan kemampuan untuk

mengambil gambar peta statis. Melakukan *geocoding*, dan memberikan penuntun arah. *Google Maps API* bersifat gratis untuk publik [***8***].

Penggunaan *Google Maps API* pada pengembangan aplikasi *android* dengan menggunakan *Eclipse* dan komputer menggunakan sistem operasi *windows*.

Kekurangan yang ada pada *Google Maps API* yaitu jika ingin melakukan akses harus terdapat layanan internet pada perangkat yang digunakan. Sedangkan kelebihan yang ada pada *Google Maps API* yaitu:

a. Dukungan penuh yang dilakukan *Google* sehingga terjamin dan bervariasi fitur yang ada pada *Google Maps API*.
b. Banyak pengembang yang menggunakan *Google Maps API* sehingga mudah dalam mencari referensi dalam pengembangan aplikasi.

2.2.4 Web Service

*Web Serv*ice adalah sebuah teknologi yang bisa digunakan untuk membuat sebuah aplikasi yang bersifat *platform-independent*. *Web Serv*ice dapat menghubungkan antar *web* yang berbeda, aplikasi yang berbeda, bahkan sampai *dev*ice yang berbeda dalam sistem operasi yang berbeda pula [***9***]**.** *Web Serv*ice dapat menjalankan operasi-operasi termasuk akses data, *update database*, mengatur integritas data, mengirim respon dari *request* yang dikirim oleh *user*. *Web serv*ice tidak memiliki tampilan karena *web serv*ice termasuk dalam *Bussiness-Serv*ice tier. Artinya di dalam *web serv*ice hanya tersedia fungsi-fungsi yang nantinya dapat digunakan oleh aplikasi lainnya. Pada aplikasi Rujukan berguna sebagai perantara untuk mendapatkan data yang akan diambil dari *database server* yang diakses melauli *mobile dev*ice. Data yang diminta akan berupa nilai balikan dengan format *JSON*.

2.2.5 PHP

PHP adalah bahasa scripting language yang dirancang secara khusus untuk penggunaaan pada *web*. PHP kependekan dari *HyperText Preprocessor*. Seperti bahasa pemrogramman *web* lainnya PHP memproses seluruh perintah yang berada dalam skrip PHP didalam *web browser* dan menampilkan *output*nya ke dalam *web*

browser klien. PHP adalah bahasa *scripting* yang menghasilkan *output* HTML ataupun *output* lain sesuai keinginan program yang dijalankan pada *server* side [***10***].

Bahasa pemrograman ini digunakan pada aplikasi antarmuka *back-end* aplikasi rujukan. Lebih tepatnya aplikasi ini akan ditempatkan pada setiap apotek untuk memantau pesanan yang dilakukan dari *sub-system mobile* / aplikasi *front-end* yang digunakan oleh konsumen serta untuk memasarkan stok obat yang terdapat di setiap apotek.

2.2.6 Mysql

MySQL merupakan *software database open source* yang paling populer di dunia, di mana saat ini digunakan lebih dari 100 juta pengguna di seluruh dunia. Dengan kehandalan, kecepatan dan kemudahan penggunaannya, MySQL menjadi pilihan utama bagi banyak pengembang *software* dan aplikasi baik di *platform web* maupun *desktop*. Pengguna MySQL tidak hanya sebatas pengguna perseorangan maupun perusahaan kecil, namun perusahaan seperti Yahoo!, *Alcatel-Lucent, Google, Nokia, Youtube, Wordpress* dan *Facebook* juga merupakan pengguna MySQL. MySQL pertama kali dibuat dan dikembangkan di Swedia, yaitu oleh David Axmark, Allan Larsson dan Michael "*Monty*" Widenius. Mereka mengembangkan MySQL sejak tahun 1980-an. Saat ini versi MySQL yang sudah stabil mencapai versi 5x, dan sedang dikembangkan versi 6x [***11***].

Database ini digunakan sebagai penyedia data *serv*ice yang digunakan untuk aplikasi *front-end* dalam melakukan *request* data pada pusat.

2.2.7 XML (Extensible Markup Language)

Ron Schmelzer et al. (2002:9) menjelaskan “dengan adanya XML, Anda dapat menyandi informasi dalam sebuah dokumen yang tidak hanya berisi data saja di dalamnya, tetapi juga berisi informasi yang menjelaskan maksud dari data tersebut, dan dalam struktur yang dapat dibaca manusia.” Secara sederhana, XML adalah dokumen yang memperbolehkan pemakainya untuk menyimpan data

secara terstruktur dan juga menyandi informasi. Beberapa keuntungan XML menurut Conolly dan Begg (2010:1074) antara lain:

a. XML didesain secara sederhana (*simplicity*) dengan menggunakan bahasa berbasis teks yang dapat dimengerti oleh manusia.
b. Tidak bergantung pada *platform* atau vendor tertentu (*platform*/*vendor independent*).
c. *User* dapat membuat *tag* sendiri sesuai kebutuhan (*extensibility*) dan dapat digunakan pada aplikasi lain (*reuse*).

Penggunaan XML untuk mendefinisikan banyak atribut di dalam infrastruktur aplikasi. Ini menyebabkan tidak diperlukannya bekerja di dalam bahasa programming *Java* untuk menangani atribut-atribut ini. Sekian ratus kode *Java* bisa menjadi hemat. Segala yang ada di dalam aplikasi – mulai dari layout *User Interface*, *string* teks, animasi, hingga komunikasi antar proses dengan layanan sistem operasi *Android* (seperti menggetarkan gadget atau memutar *ringtone*) – dapat dilakukan via XML.

Apa yang membuat XML ideal untuk pengembangan *Android*, terutama untuk pemula, adalah kemudahan penggunaannya. Penggunaannya tidak lebih sulit dari HTML. Jadi jika sudah mengetahui bagaimana menggunakan tag untuk menebalkan teks atau menyisipkan gambar di *web*site, berarti dapat menggunakan XML.

XML memberikan fleksibilitas yang luar biasa untuk mengakomodasi beragam hal di dalam aplikasi, seperti ukuran layar yang berbeda, bahasa, dan rancangan UI.

2.2.8 Rumus Haversine

Rumus *Haversine* digunakan untuk menghitung jarak *great-circle* antara dua point, yaitu jarak terdekat, terhadap permukaan bumi. Formula *Haversine* masih dianggap baik untuk perhitungan numeris, bahkan untuk permukaan yang kecil.Rumus *Haversine* dipulikasikan oleh Roger Sinnott pada *majalah Sky & Telescope* pada tahun 1984 (“Virtues of the *Haversine*”) [*7*].

Metode *Haversine* digunakan untuk menghitung jarak antara titik di permukaan bumi menggunakan garis lintang (*longitude*) dan garis bujur (*lattitude*) sebagai variabel inputan. *Haversine* formula adalah persamaan penting pada navigasi, memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang.

Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari R 6.367, 45 km, dan lokasi dari 2 titik di koordinant bola (lintang dan bujur) masing-masing adalah lon1, lat1, dan lon2, lat2, maka rumus *Haversine* dapat ditulis dengan persamaan sebagai berikut :

```
x = (lon2-lon1) * cos ((lat1+lat2)/2);
y= (lat2-lat1);
d= sqrt(x*x+y*y)*R
```

Gambar II.4 *Rumus Havershine* [*7*]

Keterangan:

x = Longitude (Lintang)

y= Lattitude (Bujur)

d= Jarak

R= Radius Bumi =6371 km

1 derajat= 0.0174532925 radian

Contoh Perhitungan *Haversine* Formula

Lokasi 1: lon1= 119.800801, lat1= -0.790175

Lokasi 2: lon2= 119.8428 , lat2= -0.8989

lat1 = -0.790175 * 0.0174532925 radian = -0.013791155 radian

lon1= 119.800801 * 0.0174532925 radian = 2.090918422 radian

lat2 = -0.8989 * 0.0174532925 radian = -0.01569 radian

lon2= 119.8428 * 0.0174532925 radian= 2.091651 radian

x = (lon2-lon1) * cos ((lat1+lat2)/2)

= (2.091651-2.090918422) * cos ((-0.013791155 + -0.01569)/2)

= 0.0007329412

y = (lat2-lat1)

= (-0.01569- (-0.013791155))

= -0.001897609

d = sqrt (x*x + y*y) * R

= sqrt((0.0007329412*0.0007329412)+(-0.001897609*-0.001897609))*6371

= sqrt (0.0000041381) * 6371

=12.96012927 km

2.2.9 Client Server

Client-server merupakan arsitektur komputasi yang memisahkan antara klien dengan *server*, dan kebanyakan arsitektur ini diimplementasikan pada jaringan komputer. Namun bisa juga diimplementasikan antar perangkat lain seperti juga handphone dengan handphone, ataupun handphone dengan komputer. Setiap klien atau *server* yang terhubung ke jaringan disebut sebagai *node*. Umumnya arsitektur ini memiliki 2 tipe *node* yaitu: klien dan *server*. Tipe ini biasanya disebut pula two-tier. Dengan arsitektur ini, memungkinkan *dev*ice untuk melakukan sharing files dan *resources*. Setiap *instance* dari *software* pada klien dapat mengirimkan *request* kepada satu atau beberapa *server* yang terhubung kepadanya. *Server* kemudian menerima *request* tersebut, memprosesnya, setelah itu mengirimkan informasi yang dihasilkan kepada klien. Meskipun konsep ini diaplikasikan pada berbagai jenis aplikasi, namun secara garis besar pola *request* dan response nya tetap sama. Beberapa penerapan arsitektur *Client Server*, yaitu pada *web* browser sebagai klien, serta *web server*, *database server*, dan *mail server* sebagai *server*. [***12***]

Keuntungan dari penerapan arsitektur *Client Server* antara lain:

1. Semua data disimpan pada *server*, sehingga manajemen sekuritas penuh berada pada *server*, hal ini menjadi baik dibandingkan dengan manajemen sekuritas data yang tersebar, *server* menjamin siapa-siapa saja yang berhak mengakses data [***12***].
2. Oleh karena peyimpanan terpusat, proses *update* data menjadi mudah untuk dilakukan daripada melakukan *update* data yang tersebar. Pada kasus *peer*

to peer, *update* data dilakukan di setiap peer, hal ini tentunya sangat memakan waktu dan rentan terjadinya inkonsistensi data [***12***].

2.2.10 GLOBAL POSITIONING SYSTEM

GPS (*Global Positioning System*) adalah sebuah sistem navigasi berbasiskan satelit yang dibangun dari sebuah jaringan yang terdiri dari 24 satelit yang diletakkan dalam orbit. *GPS* bekerja pada berbagai kondisi cuaca, di manapun posisi di dunia, dan 24 jam satu hari [***3***]

GPS (*Global Positioning System*) adalah sistem satelit navigasi dan penentuan posisi yang dimiliki dan dikelola oleh Amerika Serikat. Sistem ini didesain untuk memberikan posisi dan kecepatan tiga dimensi serta informasi mengenai waktu. *GPS* terdiri dari 3 segmen yaitu segmen angkasa, kontrol/pengendali, dan pengguna. Segmen angkasa terdiri dari 24 satelit yang

Beroperasi dalam 6 orbit pada ketinggian 20.200 km dengan periode 12 jam (satelit akan kembali ke titik yang sama dalam 12 jam). Segmen Kontrol/Pengendali terdapat pusat pengendali utama yang terdapat di *Colorodo Springs*, dan 5 stasiun pemantau lainnya dan 3 antena yang tersebar di bumi ini. Pada sisi pengguna dibutuhkan penerima *GPS* yang biasanya terdiri dari penerima, prosesor, dan antena. [***3***]

Sebuah *GPS* receiver harus terkunci dengan sinyal dari setidaknya 3 satelit untuk menghitung posisi 2D (*latitude dan longitude*) dan pergerakan *track*. Dengan 4 atau lebih satelit, *receiver* dapat menentukan posisi 3D *user* (*latitude, longitude, dan altitude*). Begitu posisi *user* sudah ditentukan, unit *GPS* dapat menghitung informasi lain, seperti kecepatan, pelacakan, jarak tempuh, jarak tujuan, waktu matahari terbit dan terbenam, dan sebagainya [***13***].



Gambar II.5 *Global Positioning System* [*3*]

2.2.11 Location Based Service

Ide menggunakan *mobile* phone adalah untuk memberikan layanan lain yang juga berguna selain layanan komunikasi dasar dan telah dimulai sejak 1990 semenjak adanya Internet pada *Vo*ice *Telephony*. *Location based serv*ice (LBS) mengacu pada kumpulan aplikasi yang memberdayakan pengetahuan posisi geografis *mobile dev*ice untuk memberikan layanan-layanan [***14***]. *Location based serv*ice banyak memberikan keuntungan pada client, seperti:

a. Untuk menentukan lokasi bisnis atau layanan terdekat seperti Bank atau Hotel.
b. Memperoleh peringatan, misalnya notifikasi *Sale* pada *Shopping Mall* atau berita dari area kemacetan terdekat.
c. Pencari teman atau pencari lokasi phone yang tercuri.

Lokasi dapat ditentukan dari terminology spasial ataupun dalam deskripsi teks. Lokasi spasial dapat direpresentasikan pada penggunaan sistem kordinat *latitude longitude-altitude*. *Latitude* didefinisikan sebagai 0-90 derajat utara atau selatan ekuator. *Longitude* sebagai 0-180 derajat barat atau timur dari meridian utama yang melewati *Greenwich*, Inggris. *Altitude* direpresentasikan sebagai satuan meter di atas permukaan laut. Deksripsi teks didefinisikan sebagai lokasi jalan, yang mengandung nama kota, kode pin, dan sebagainya [***14***].

2.3. *Structure Analysis and Design*

Structured analysis and design adalah pendekatan sistem analisis dan desain sistem informasi secara hierarki. SADM diproduksi untuk CCTA, kantor pemerintah Inggris yang berkaitan dengan penggunaan teknologi di pemerintahan dari tahun 1980 dan seterusnya [***15***].

2.3.1 Diagram Konteks

Diagram Konteks adalah data flow diagram tingkat atas (DFD *Top Level*), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. (CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem) [***15***]**.**

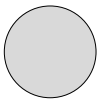
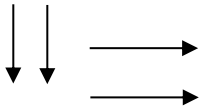
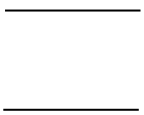
2.3.2 Data Flow Diagram

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. DFD terdiri dari empat komponen utama diantaranya adalah :

1. Entitas.
2. Proses.
3. Data Store.
4. Data Flow.

DFD dapat dikatakan merupakan breakdown dari diagram konteks. Analisis mengenai sistem yang lebih detail dapat dilakukan dengan melakukan breakdown pada DFD level 1, 2 dan seterusnya. Berikut adalah Tabel II.1 Simbol Diagram Konteks dan DFD .

Tabel II.1 Simbol Diagram Konteks dan DFD [*15*]

Simbol	Nama Simbol	Fungsi
	Entitas Luar	Menggambarkan entitas eksternal yang berhubungan dengan sistem
	Sistem (Konteks) / Proses (DFD)	Menggambarkan proses yang ada dalam suatu sistem
	Aliran Data/Informasi	Menggambarkan aliran data antar proses, data store dan entitas luar
	*Data Storage*	Mengambarkan tempat penyimpanan data di dalam sistem

2.3.3 Spesifikasi Proses

Spesifikasi Proses adalah tabel yang berisi keterangan atau deskripsi dari semua proses yang terdapat pada DFD. Logika proses yang ditulis pada spesifikasi proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau pseudecode tetapi tidak boleh mengkombinasikan kedua bahasa tersebut [***15***]. Berikut adalah Tabel II. 2 Contoh Spesifikasi Proses

Tabel II. 2 Contoh Spesifikasi Proses [*15*]

No	Proses	Keterangan
1	No Proses	Menyatakn no proses
	Nama Proses	Menyatakan nama proses
	Source (Sumber)	(Menyatakan sumber data *input* menuju proses)
	Input	(menyatakan isi data yang masuk ke proses)

No	Proses	Keterangan
	Output	(menyatakan informasi yang keluar dari proses)
	Destination	(menatakan tujuan informasi *output* dari proses)
	Logika Proses	(menyatakan algoritma dari proses)

2.3.4 Kamus Data

Kamus data adalah tabel yang berisi deskripsi dari data yang mengalir pada DFD, Penjelasan struktur data dapat berupa *field* tiap data harus sama dengan yang sudah dimodelkan di ERD [***15***]. Berikut adalah Tabel II. 3 Contoh Kamus Data Tabel II. 3 Contoh Kamus Data .

Tabel II. 3 Contoh Kamus Data [*15*]

Nama aliran data	Data Siswa Nama dari data yang digunakan
Where used/How used	Daftar pada poin proses apa data digunakan
Keterangan	Uraian singkat dari data yang digunakan
Struktur Data	Daftar komponen data yang ada pada data yang digunakan
Deskripsi	Jenis data dalam representasi komputer untuk masing-masing data.

2.4. *Object Oriented Analysis Design*

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut *analisis berorientasi objek* (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau *subsistem* [***16***].

2.4.1 Unified Modelling Language (UML)

Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO).

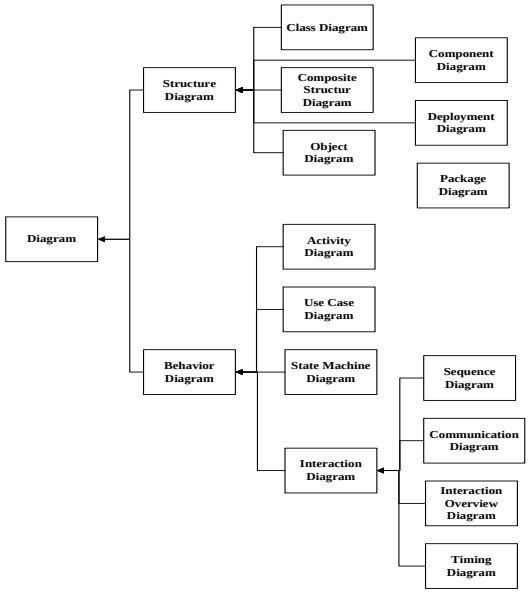
UML merupakan standar yang relatif terbuka yang dikontrol oleh *Object Management Group* (OMG), sebuah konsorsium terbukayang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas, khususnya interoperabilitas sistem berorientasi objek. OMG mungkin lebih dikenal dengan standar-standar CORBA (*Common Object Request Broker Architecture*) [***16***].

Berikut adalah Tabel II.4 Jenis Diagram Resmi UML beserta dengan kegunaannya :

Tabel II.4 Jenis Diagram Resmi UML [*16*]

Diagram	**Kegunaan**
Activity	*Behavior* prosedural dan paralel
Class	*Class*, fitur, dan hubungan-hubungan
Communication	Interaksi antar objek; penekanan pada jalur
Component	Struktur dan koneksi komponen
Composite structure	Dekomposisis *runtime* sebuah *class*
Deployment	Pemindahan artifak ke node
Interaction overview	Campuran *sequence* dan *activity* diagram
Object	Contoh konfigurasi dari contoh-contoh
Package	Struktur hirarki *compile-time*
Sequence	Interaksi antar objek; penekanan pada *sequence*
State machine	Bagaimana *event* mengubah objek selama aktif
Timing	Interaksi antar objek; penekanan pada timing
Use case	Bagaimana pengguna berinteraksi dengan sebuah sistem

Berikut adalah Gambar II.6 Klarifikasi Jenis Diagram UML [***16***]



Gambar II.6 Klarifikasi Jenis Diagram UML [*16*]

Adapun penjelasan dari *Class* Diagram, *Use case* Diagram dan *Sequence* Diagram adalah sebagai berikut [***16***]:

1. *Class* Diagram

Class diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat di antaranya. *Class* diagram juga menunjukan properti dan operasi sebuah *class* dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut. Kotak-kotak yang terdapat di dalam diagram merupakan *class*, yang dibagi menjadi tiga bagian : nama *class* (cetak tebal), atributnya dan operasinya.

a. Properti

Properti mewakili fitur-fitur struktural dari sebuah *class*. Properti merupakan sebuah konsep tunggal, tetapi tampak seperti dua notasi yang sedikit berbeda: atribut dan asosiasi.

b. Atribut

Notasi atribut mendeskripsikan properti dengan sebaris teks di dalam kotak *class* tersebut.

c. Operasi

Operasi merupakan suatu kegiatan yang dimengerti sebuah *class* untuk dilakukan. Operasi secara jelas berhubungan dengan metode dalam sebuah *class*.

2. *Use case* Diagram

Use case diagram menampilkan aktor, *use case* dan hubungan antar keduanya :

a. Aktor mana yang menggunakan *use case* mana
b. *Use case* mana yang memasukkan *use case* lain

3. *Sequence* Diagram

Sebuah *sequence* diagram secara khusus menjabarkan *behaviour* sebuah *skenario* tunggal. Diagram tersebut menunjukkan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam *use case*.

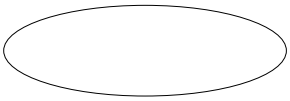
4. *Activity* Diagram

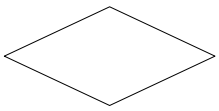
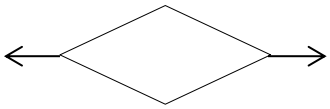
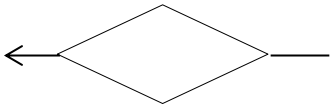
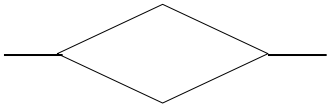
Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. dalam beberapa hal, diagram ini seperti sebuah diagram alir, tetapi perbedaan prinsip antar diagram ini dan notasi diagram alir adalah diagram ini mendukung *behaviour* paralel.

2.5. *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) adalah sekumpulan cara atau peralatan untuk mendeskripsikan data-data atau objek-objek yang dibuat berdasarkan dan berasal dari dunia nyata yang disebut entitas (*entity*) serta hubungan (*relationship*) antar entitas-entitas tersebut dengan menggunakan beberapa notasi [***16***]. Komponen-komponen pembentuk ERD dapat di lihat pada Tabel II.5 Komponen-komponen ERD .

Tabel II.5 Komponen-komponen ERD [*16*]

Notasi	Komponen	Keterangan
	Entitas	Individu yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain
	Atribut	Properti yang dimiliki oleh suatu entitas, dimana dapat mendeskripsikan karakteristik darri entitas tersebut

	Relasi	Menunjukkan hubungan diantara sejumlah entitas yang berbeda
	Relasi 1 : 1	Relasi yang menunjukkan bahwa setiap entitas pada himpunan entitas pertama berhubungan dengan paling banyak satu entitas pada himpunan entitas kedua
	Relasi 1 : N	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak atau sebaliknya. Setiap entitas dapat berelasi dengan banyak entitas pada himpunan entitas yang lain
	Relasi N : N	Hubungan ini menunjukkan bahwa setiap entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua, demikian juga sebaliknya

2.6. *Tools Pembangunan Aplikasi*

2.6.1 Android Studio

Android Studio merupakan pengembangkan dari *Eclipse* IDE, dan dibuat berdasarkan *IDE Java* populer, yaitu *IntelliJ IDEA*. *Android Studio* direncanakan untuk menggantikan *Eclipse* ke depannya sebagai IDE resmi untuk pengembangan aplikasi *Android*. [***17***]

Sebagai pengembangan dari *Eclipse*, *Android Studio* mempunyai banyak fitur-fitur baru dibandingkan dengan *Eclipse IDE*. Berbeda dengan *Eclipse* yang menggunakan Ant, *Android Studio* menggunakan *Gradle* sebagai *build environment*. Fitur-fitur lainnya adalah sebagai berikut :

- Menggunakan *Gradle-based build system* yang fleksibel.
- Bisa membangun *multiple* APK .
- Template *support* untuk *Google Serv*ices dan berbagai macam tipe perangkat.
- Layout editor yang lebih bagus.
- Built-in *support* untuk *Google* Cloud *Platform*, sehingga mudah untuk integrasi dengan *Google* Cloud Messaging dan *App Engine*.
- Import library langsung dari *Maven repository*

2.6.2 Adobe Dreamweaver

Adobe Dreamweaver adalah sebuah HTML (*Hyper Text Markup Language)* editor professional untuk mendesain *web* secara visual dan mengelola situs atau halaman *web*. *Dreamweaver* merupakan perangkat lunak utama yang digunakan oleh *web designer* maupun *web programmer* dalam mengebangkan situs *web*. Hal ini disebabkan ruang kerja, fasilitas dan kemampuan *Dreamweaver* yang mampu meningkatkan produktivitas dan efektivitas dalam desain maupun membangun suatu situs *web*.

Dalam hal ini penulis menggunakan *Adobe Dreamweaver CS3* karena memiliki kemampuan untuk menyunting kode dengan lebih baik, juga dapat melakukan print kode pada jendela *Code View*, selain itu juga memiliki fasilitas *Code Hints* untuk membantu dalam urusan tag-tag, serta *Tag Inspector* yang sangat berguna dalam menangani tag-tag HTML [***18***].

2.6.3 PHP (Hypertext Preprocessor)

PHP (*Hypertext Preprocessor*) merupakan bahasa pemrograman berbasis *web* yang memiliki kemampuan untuk memroses dan mengolah data secara dinamis. PHP dapat dikatakan sebagai sebuah *sever-side embedded script language*, yang artinya semua sintaks dan perintah program yang anda tulis akan sepenuhnya dijalankan oleh *server*, tetapi dapat disertakan pada halaman HTML biasa. [***18***]

PHP merupakan bahasa pemrograman yang dijalankan di *server web* yang berbentuk *scripting*, sistem kerja program ini adalah sebagai *interpreter* bukan sebagai *Compiler*. PHP menurut Syafii (2004 : v) merupakan bahasa pemrograman berbasis *web* yang terbukti sangat *reliable* penggunaannya dan mempunyai dukungan yang kuat. PHP berjalan pada bermacam sistem operasi (*Windows*, *Linux*, Unix, dll) dan mendukung bermacam *database* (*MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC*, dll). PHP merupakan perangkat lunak yang open *source*, dapat secara bebas didownload dan digunakan serta kompatibel dengan hampir semua *server web* (Apache, IIS, dll).

2.6.4 Php Myadmin

Saat ini banyak sekali perangkat lunak yang dapat dimanfaatkan untuk mengelola basis data dalam *MySQL*, salah satunya adalah *phpMyAdmin*. Dengan *phpMyAdmin*, seseorang dapat membuat *database*, membuat tabel, mengisi data, dan lain-lain dengan mudah, tanpa harus menghafal baris perintahnya.

PhpMyadmin adalah perangkat lunak bebas yang ditulis dalam bahasa pemrograman PHP yang digunakan untuk menangani administrasi MySQL melalui *World Wide Web*. [***18***] *PhpMyAdmin* mendukung berbagai operasi *MySQL*, diantaranya : mengelola basis data, tabel-tabel, bidang (*fields*), relasi (*relations*), indeks, pengguna (*users*), perijinan (*permissions*), dan lain-lain.

2.6.5 XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache* HTTP *Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama *XAMPP* merupakan singkatan dari X (empat sistem operasi apapun),

Apache, *MySQL*, *PHP* dan *Perl*. Program ini tersedia dalam GNU *General Public Li*cense dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis. Untuk mendapatkanya dapat mendownload langsung dari *web* resminya [***18***].

2.7. Metode Penelitian

Metodologi penelitian yang digunakan dalam penelitian ini adalah metode penelitian terapan, metode ini bertujuan untuk menyelesaikan permasalahan yang timbul masyarakat berdasarkan fenomena yang ada. Teknik sampling dengan menggunakan *probability sampling,* dengan menggunakan *proportionate stratified random sampling,* perhitungan sampel menggunakan rumus *slovin*. Obyek penelitian yaitu rujukan ibu hamil di kabupaten bandung.

2.7.1 Populasi

Populasi adalah wilayah yang terdiri atas obyek/subyek yang mempunyai kualitas dan karakteristik tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya (Sugiyono, 2010: 61).

Populasi dalam penelitian ini adalah bidan yang ada diseluruh kabupaten bandung yang telah ditentukan dalam batasan masalah, bidan yang diteliti merupakan rata-rata tenaga kesehatan yang menindak lanjuti rujukan dari hasil pemeriksaan pasien atau penderita yang ada di puskesmas atau di BPS tempat pelayan kesehatan.

2.7.2 Sampel

Sampel adalah bagian dari jumlah dan karakteristik yang dimiliki oleh populasi (Sugiyono, 2010: 62). Pengambilan sampel dalam penelitian ini dengan teknik *probability sampling* yaitu *proportionate stratified random sampling* dengan menggunakan rumus *slovin*. Menurut Sugiyono (2010: 63), *Probability sampling* adalah teknik pengambilan sampel yang memberikan peluang yang sama bagi setiap unsur (anggota) populasi untuk dipilih menjadi anggota sampel. Besarnya sampel dalam penelitian ini ditentukan dengan rumus *Slovin* dengan taraf keyakinan sebesar 85% sebagai berikut:

$$n = \frac{N}{1 + Ne^2}$$

n = jumlah elemen / anggota sampel
N = jumlah elemen / anggota populasi
e = error level (taraf keyakinan 85% maka tingkat error signifikan 15% = 0,15.

2.8. Metode Pengujian

2.8.1 User Acceptence Testing (UAT)

Menurut Perry (2006:70), *User Acceptance Testing* merupakan pengujian yang dilakukan oleh *end-user* dimana *user* tersebut adalah *staff*/karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya.

Menurut Lewis (2009:134), setelah dilakukan *system testing, acceptance testing* menyatakan bahwa sistem *software* memenuhi persyaratan. *Acceptance testing* merupakan pengujian yang dilakukan oleh pengguna yang menggunakan teknik pengujian *black box* untuk menguji sistem terhadap spesifikasinya. Pengguna akhir bertanggung jawab untuk memastikan semua fungsionalitas yang relevan telah diuji.

Menurut Black (2002:7), *acceptance testing* biasanya berusaha menunjukkan bahwa sistem telah memenuhi persyaratan-persyaratan tertentu. Pada pengembangan *software* dan *hardware* komersial, *acceptance test* biasanya disebut juga "*alpha tests"* (yang dilakukan oleh pengguna *in-house)* dan "*beta tests"* (yang dilakukan oleh pengguna yang sedang menggunakan atau akan menggunakan sistem tersebut). *Alpha* dan *beta test* biasanya juga menunjukkan bahwa produk sudah siap untuk dijual atau dipasarkan. *Acceptance testing* mencakup data, *environment* dan skenario yang sama atau hampir sama pada saat *live* yang biasanya berfokus pada skenario penggunaan produk tertentu.

Dari definisi di atas, *user acceptance testing* adalah pengujian yang dilakukan oleh pengguna dari sistem tersebut untuk memastikan fungsi-fungsi yang ada pada sistem tersebut telah berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

2.8.2 Blackbox Testing / Fungsional Testing

Menurut Black (2007:45-46), *black box testing* adalah melakukan pengujian terhadap apa yang dilakukan oleh sistem, khususnya perilaku dan juga masalah bisnis. *Black box testing* bertujuan untuk mengidentifikasikan *bug-bug* yang ada pada hasil, kinerja dan juga perilaku sistem. Pengujian ini biasanya dilakukan oleh pihak penguji ketika *integration test*, *system test*, dan *acceptance test*, tetapi juga berguna untuk tahap yang lebih awal untuk membantu membangun *unit test case* dan *component test case* yang lebih baik.

Menurut Perry (2006:69), *functional testing* juga dapat disebut sebagai *black box testing* karena tidak ada pengetahuan dari logika internal sistem yang digunakan untuk membuat *test case.* Biasanya dalam pengujian fungsional, teknik validasi lebih digunakan untuk melakukan pengujian. Tim penguji melakukan validasi terhadap *function key* yang ada dan mengobservasi hasilnya.

Kelebihan dari *functional testing*:

- Melakukan simulasi terhadap kegunaan sistem yang sebenarnya.
- Tidak membuat asumsi terhadap struktur sistem.

Kekurangan dari functional testing:

- Dapat berpotensi menghilangkan logika yang salah dalam *software*.
- Memungkinkan terjadinya pengujian yang redundan.

Kedua metode ini berguna untuk validasi pada seluruh sistem. Seperti contohnya, sebuah *functional test case* dapat diambil dari dokumentasi tentang deskripsi bagaimana melakukan sebuah fungsi seperti menerima inputan *bar code*. Sebuah *structural test case* dapat diambil dari dokumentasi teknikal manual. Untuk pengujian sistem yang efektif diperlukan kedua metode tersebut.

Menurut Lewis (2009:39), pada *black box testing* atau pengujian fungsional, kondisi pengujian dikembangkan berdasarkan fungsionalitas dari program atau sistem yang akan diuji, oleh karena itu, penguji membutuhkan informasi mengenai data input dan output yang diamati, tetapi tidak mengetahui bagaimana program atau sistem tersebut bekerja atau tidak perlu mengetahui bagaimana struktur internal dari program tersebut melakukan eksekusi. Penguji berfokus pada pengujian fungsionalitas dari program terhadap spesifikasi.

Dari definisi di atas, *black box testing* adalah pengujian terhadap apa yang dilakukan oleh sistem, khususnya perilaku dan juga masalah bisnis di mana penguji membutuhkan informasi mengenai data input dan output yang diamati tetapi tidak perlu mengetahui struktur internal dari sistem tersebut.