BAB II
TINJAUAN PUSTAKA

2.1. Rumah Sakit

2.1.1. Pengertian Rumah Sakit

Pengertian Rumah sakit menurut Keputusan Menteri Kesehatan Republik Indonesia No. 340/MENKES/PER/III/2010:

1. Rumah sakit adalah institusi pelayanan kesehatan yang menyelenggarakan pelayanan kesehatan perorangan secara paripurna yang menyediakan pelayanan. rawat inap, rawat jalan, dan gawat darurat.
2. Rumah sakit umum adalah rumah sakti yang memberikan pelayanan pada semua bidang dan jenis penyakit.
3. Rumah sakit khusus adalah rumah sakit yang memberikan pelayanan utama pada satu bidang atau satu jenis penyakit tertentu, berdasarkan disiplin golongan ilmu,golongan umur, organ atau jenis pemyakit.
4. Klasifikasi rumah sakit adalah penggolongan kelas rumah sakit berdasarkan fasilitas dan pelayanan rumah sakit,
5. Fasilitas adalah segala sesuatu yang menyangkut sarana dan prasarana maupun alat (baik alat medik maupun non-medik) yang dibutuhkan rumah sakit dalam memberikan pelayanannya yang sebaik-baiknya bagi pasien.
6. Sarana adalah segala sesuatu benda baik secara fisik yang dapat tervisualisasi oleh mata maupun teraba oleh panca-indera dan mudah dapat dikenali oleh pasien dan (umumnya) merupakan bagian dari suatu bangunan gedung maupun gedung itu sendiri.
7. Prasarana adalah benda maupun jaringan / instansi yang membuat suatu sarana dapat berfungsi yang sesuai tujuan yang diharapkan.
8. Tenaga adalah tenaga yang bekerja di rumah sakit secara purna waktu dan berstatus pegawai tetap. [1]

2.1.2. Tugas Rumah Sakit

Rumah sakit tidak hanya menyediakan fasilitas perawatan saja, tetapi juga keperluan administrasi dan surat menyurat yang dibutuhkan pasien, seperti kelengkapan data rekam medis. Berikut ini adalah tugas sekaligus fungsi dari rumah :

1. Melaksanakan pelayanan medis, dan pelayanan penunjang medis.
2. Melaksanakan pelayanan medis tambahan, dan pelayanan tambahan penunjang medis.
3. Melaksanakan pelayanan dokter kehakiman.
4. Melaksanakan pelayanan medis khusus.
5. Melaksanakan pelayanan rujukan kesehatan.
6. Melaksanakan pelayanan kedokteran gigi.
7. Melaksanakan pelayanan kedokteran social.
9. Melaksanakan pelayanan penyuluhan kesehatan.
10. Melaksanakan pelayanan rawat jalan atau rawat darurat dan rawat tinggal (observasi).
11. Melaksanakan pelayanan rawat inap.
12. Melaksanakan pelayanan administrasi.
13. Melaksanakan pendidikan medis.
14. Membantu pendidikan tenaga medis umum.
15. Membantu pendidikan tenaga medis spesialis.
16. Membantu penelitian dan pengembangan kesehatan.
17. Membantu kegiatan penyelidikan epidemilogi. [2]

2.2. Rawat Jalan

Pelayanan rawat jalan adalah salah satu bentuk dari pelayanan kedokteran. Secara sederhana yang dimaksud dengan pelayanan rawat jalan adalah pelayanan kedokteran yang disediakan untuk pasien tidak dalam bentuk rawat inap (*hospitalization*).

Menurut surat Keputusan Mentri Kesehatan RI no.560/MENKES/SK/IV/2003 tentang tarif perjan rumah sakit bahwa rawat jalan adalah pelayanan pasien untuk

observasi, diagnosis, pengobatan, rehabilitasi medik dan pelayanan kesehatan lainnya tanpa menginap di rumah sakit.

2.3. Rawat Inap

Menurut surat Keputusan Mentri Kesehatan RI no.560/MENKES/SK/IV/2003 tentang pola tarif perjan (perusahaan jawatan) rumah sakit bahwa rawat jalan adalah pelayanan pasien untuk observasi, diagnosis, pengobatan, rehabilitasi medik dan pelayanan kesehatan lainnya tanpa menginap di rumah sakit. Pengertian rawat inap lainnya adalah satu bentuk proses pengobatan atau rehabilitasi oleh tenaga pelayanan kesehatan profesional pada pasien yang menderita suatu penyakit tertentu, dengan cara di inapkan di ruang rawat inap tertentu sesuai dengan jenis penyakit yang dialaminya.

Fasilitas Rawat inap disediakan dan dijalankan secara sistematis oleh tenaga medis dan nonmedis, disediakan oleh pihak penyedia pelayanan kesehatan (klinik, rumah sakit, puskesmas). Berikut ini adalah beberapa tujuan rawat inap :

1. Untuk memudahkan pasien mendapatkan pelayanan kesehatan yang komprehensif
2. Untuk memudahkan menegakkan diagnosis pasien dan perencanaan terapi yang tepat
3. Untuk memudahkan pengobatan dan terapi yang akan dan harus didapatkan pasien.
4. Untuk mempercepat tindakan kesehatan.
5. Memudahkan pasien untuk mendapatkan berbagai jenis pemeriksaan penunjang yang diperlukan
6. Untuk mempercepat penyembuhan penyakit pasien
7. Untuk memenuhi kebutuhan pasien sehari-hari yang berhubungan dengan penyembuhan penyakit, termasuk pemenuhan gizi dan lain lain.[3]

2.4. Analisis Terstruktur Perancangan dan Implementasi Sistem Informasi (STRADIS)

Merupakan salah satu metodologi berorientasi proses, biasa juga disebut sebagai metodologi terstruktur. Metodologi ini pertama sekali di tulis pada buku *Structured*

System Analysis yang ditulis oleh Gane dan Sarson, pada tahun 1979. STRADIS dianggap dapat diterapkan dalam pengembangan Sistem informasi, tanpa memperhatikan ukuran sistem dan apakah sistem akan diotomasikan atau tidak. Dalam praktiknya, STRADIS sering dimanfaatkan dalam lingkungan di mana sebagian dari sistem informasi diotomasikan. Berikut ini adalah tahapan - tahapan dalam pengembangan menggunakan metodologi STRADIS :

1. *Initial Study* (Studi awal)

   Tahap pertama dalam metodologi ini adalah studi awal. Dimana pada tahap ini yakni meyakinkan bahwa sistem yang dipilih dalam pengembangan adalah benar – benar di butuhkan. Apakah nantinya sistem yang dikembangkan dapat memberi kontribusi terhadap peningkatan pendapatan, menekan biaya, dan meningkatkan layanan.

2. *Detailed Study* (Studi detail)

   Pada tahapan ini lebih berfokus pada sistem yang sedang berjalan. Tahapan ini dilakukan dengan menginvestigasi sistem yang sedang berjalan melalui identifikasi pengguna yang relevan. Terdapat 3 (tiga) tingkatan pengguna, yaitu :

a) *Senior managers* yang berkaitan dengan tanggung jawab terhadap profit.
b) *Middle managers* yang bertanggung jawab atas unit yang terkain.
c) *End users*, pengguna yang secara lansung berinteraksi dengan atau bekerja lansung dengan sistem. Estimasi biaya dan keuntungan pada tahapan studi awal akan disempurnakan lebih lanjut pada tahapan studi detail.

Analis perlu menginvestigasi asumsi asumsi yang berkaitan dengan estimasi, dan meyakinkan aspek – aspek yang telah di perkirakan. Secara keseluruhan, studi detail terdiri dari :

a) Definisi pengguna dan unit yang terkait dengan sistem baru.
b) Pemodelan logis dari sistem berjalan, diagram aliran data, tampilan antar muka sistem (jika relevan), detail diagram aliran data untuk setiap proses yang penting, definisi data pada level detail.

c) Pernyataan terhadap peningkatan yang diharapkan. Seperti pendapatan, biaya, dan layanan.

d) Estimasi dari biaya dan waktu yang telah diuraikan sebelumnya.

3. *Defining and Designing Alternative Solutions* (Definisi dan Perancangan Solusi Alternatif)

Tahap selanjutnya adalah mendifinisikan solusi alternatif terhadap masalah pada sistem yang sedang berjalan. Analis dengan menggunakan objek yang telah ada untuk menghasilkan sebuah DFD logis baru, atau rancangan sistem usulan. DFD baru tersebut dikembangkan menjadi level yang lebih detail yang menunjukkan hal – hal penting pada tujuan sistem yang sudah terpenuhi. Laporan pada tahap ini harus menunjukkan pembuat keputusan yang relevan, dan sebuah komitmen pada setiap alternatif yang sudah dibuat. Laporan tersebut terdiri

1) DFD sistem berjalan.
2) Keterbatasan sistem berjalan, termasuk estimasi biaya dan keuntungan.
3) DFD logis sistem usulan.

Untuk setiap alternatif yang diidentifikasi, rancangan harus mencakup pernyataan yang meliputi :

1) Bagian – bagian DFD yang akan diimplementasikan.
2) Tampilan tatap muka sistem (Terminal, laporan, fasilitas query, dan lainnya.)
3) Estimasi biaya dan keuntungan.
4) Garis besar jadwal pengimplementasian.
5) Resiko yang akan terlibat.

4. *Physical Design* (Rancangan Fisik)

Tim perancang akan meningkatkan alternatif rancangan yang dipilih menjadi rancangan fisik, yang mencakup beberapa aktifitas paralel :

a) Menghasilkan detil dari DFD termasuk penanganan kesalahan, semua logika proses, kamus data, format layar dan laporan yang harus disetujui pengguna.

b) Menormalisasikan simpanan data yang didefinisikan dalam DFD.

c) Merancang file fisik atau basis data berdasarkan pada simpanan data.

d) Mendapatkan hierarki modular dari fungsi-fungsi dalam DFD (gunakan analisis *transform-centred* dan *transaction-centred*). Estimasi biaya untuk mengembangkan dan mengoperasikan sistem baru harus dilakukan.

5. Tahapan lainnya

Tahapan selanjutnya tidak jelas didefinisikan pada metodologi ini, namun berikut ini beberapa hal yang perlu ditambahkan menurut Gane dan Sarson :

1) Membuat rencana implementasi, termasuk rencana pengujian dan penerimaan sistem.
2) Mengembangkan program aplikasi dan fungsi komunikasi data/basis data secara bersamaan.
3) Mengkonversi dan memuat basis data.
4) Menguji dan memastikan penerimaan tiap bagian sistem.
5) Memastikan bahwa sistem memenuhi kriteria kinerja (waktu tanggap dan *throughput*) dalam beban yang realistis.
6) Mengoperasikan sistem dan membenahi beberapa hambatan.
7) Bandingkan fasilitas dan kinerja sistem keseluruhan dengan objektif awal, dan benahi perbedaan yang ada.
8) Analisis berbagai permintaan untuk perbaikan, buat prioritasnya, dan tempatkan sistem dalam status *maintenance*.[5]

2.5. Data Flow Diagram

Data Flow Diagram (DFD) merupakan diagram yang mengunakan notasi-notasi atau simbol-simbol untuk mengambarkan sistem jaringan kerja antar fungsi-fungsi yang berhubungan satu sama lain dengan aliran dan penyimpanan data.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan

fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan. Salah satu keuntungan menggunakan diagram aliran data adalah memudahkan pemakai (*user*) yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan.

DFD terdiri dari diagram konteks (*context diagram*) dan diagram rinci (level diagram). Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh input ke sistem atau output dari sistem. Dalam diagram konteks biasanya hanya ada satu proses. Tidak boleh ada store dalam diagram konteks. Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram level di atasnya.[4]

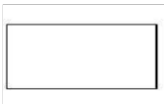
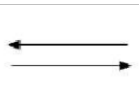
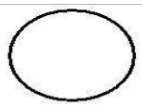
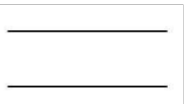
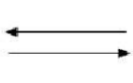
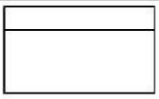
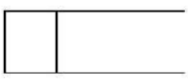
2.5.1. Konsep *Data Flow Diagram*

Dalam perancangannya, DFD memiliki beberapa cara penulisan notasi yang berbeda, namun tetap saja tidak mengubah fungsi dan tujuan utamanya. Berikut ini adalah notasi DFD :

1. Entitas Eksternal (*External Entity*).

Entitas Eksternal (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan *input* atau menerima *output* dari sistem.

Tabel 2 1 Simbol dan notasi pada DFD

Keterangan	Entitas Eksternal	Aliran Data	Proses	Penyimpanan Data
Komponen DFD menurut Yourdon dan DeMarco				
Komponen DFD menurut Gene dan Serson				

2. Aliran Data (*Data flow*).

Aliran data mengalir diantara proses (*process*), simpanan data (*data store*) dan kesatuan luar (*External entity*). Aliran data ini menunjukkan arus dari data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem.

3. Proses (*Process*).

Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu aliran data yang masuk ke dalam proses untuk dihasilkan aliran data yang akan keluar dari proses.

4. Penyimpanan Data (Data Store).

Penyimpan data (*data store*) merupakan penyimpan data yang dapat berupa: suatu file atau basis data di sistem komputer, suatu arsip atau catatan manual, suatu tabel acuan manual, suatu agenda atau buku.

DFD sangat berbeda dengan bagan alir (*flowchart*). Perbedaannya adalah sebagai berikut:

1. Proses di DFD dapat beroperasi secara parallel, sehingga beberapa proses dapat dilakukan serentak sedangkan bagan alir cenderung menunjukkan proses yang urut.
2. DFD lebih mencerminkan arus dari data di suatu sistem, sedang bagan alir sistem lebih menunjukkan arus dari prosedur dan bagan alir program lebih menunjukkan arus dari algoritma.
3. DFD tidak menunjukkan proses perulangan *(loop)* dan proses keputusan *(decision)*, sedang bagan alir menunjukkanya.

   Ada 3 (tiga) jenis DFD, yaitu ;

   i. *Context Diagram (CD)*
   ii. DFD Fisik
   iii. DFD Logis[4]

2.5.2. DFD Level

DFD dapat digambarkan dalam Diagram Context dan Level n. Huruf n dapat menggambarkan level dan proses di setiap lingkaran. Berikut ini adalah penjelasannya :

1. *Context Diagram* (CD)

Jenis pertama *Context Diagram*, 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).

Beberapa hal yang harus diperhatikan dalam menggambar CD;

a) Terminologi sistem :
   i. Batas Sistem adalah batas antara "daerah kepentingan sistem".
   ii. Lingkungan Sistem adalah segala sesuatu yang berhubungan atau mempengaruhi sistem tersebut.
   iii. *Interface* adalah aliran yang menghubungkan sebuah sistem dengan linkungan sistem tersebut.
b) Menggunakan satu simbol proses,
c) Nama/keterangan di simbol proses tersebut sesuai dengan fungsi sistem tersebut,
d) Antara Entitas Eksternal/Terminator tidak diperbolehkan komunikasi langsung
e) Jika terdapat termintor yang mempunyai banyak masukan dan keluaran, diperbolehkan untuk digambarkan lebih dari satu sehingga mencegah penggambaran yang terlalu rumit, dengan memberikan tanda asterik (*) atau garis silang (#).
f) Jika Terminator mewakili individu (personil) sebaiknya diwakili oleh peran yang dipermainkan personil tersebut.
g) Aliran data ke proses dan keluar sebagai output keterangan aliran data berbeda.

2. Diagram Level n / *Data Flow Diagram Levelled*

Dalam diagram n DFD dapat digunakan untuk menggambarkan diagram fisik maupun diagram diagram logis. Dimana Diagram Level *n* merupakan hasil pengembangan dari *Context Diagram* ke dalam komponen yang lebih detail tersebut

disebut dengan *top-down partitioning.* Jika melakukan pengembangan dengan benar, maka akan mendapatkan DFD-DFD yang seimbang.

Beberapa hal yang harus diperhatikan dalam membuat DFD ialah:

1) Pemberian Nomor pada diagram level n dengan ketentuan sebagai berikut:
   i. Setiap penurunan ke level yang lebih rendah harus mampu merepresentasikan proses tersebut dalam sepesifikasi proses yang jelas. Sehingga seandainya belum cukup jelas maka seharusnya diturunkan ke level yang lebih rendah.
   ii. Setiap penurunan harus dilakukan hanya jika perlu.
   iii. Tidak semua bagian dari sistem harus diturunkan dengan jumlah level yang sama karena yang kompleks bisa saja diturunkan, dan yang sederhana mungkin tidak perlu diturunkan. Selain itu, karena tidak semua proses dalam level yang sama punya derajat kompleksitas yang sama juga.
   iv. Konfirmasikan DFD yang telah dibuat pada pemakai dengan cara *top-down*.
   v. Aliran data yang masuk dan keluar pada suatu proses di level n harus berhubungan dengan aliran data yang masuk dan keluar pada level n+1. Dimana level n+1 tersebut mendefinisikan sub-proses pada level n tersebut.
   vi. Penyimpanan yang muncul pada level n harus didefinisikan kembali pada level n+1, sedangkan penyimpanan yang muncul pada level n tidak harus muncul pada level n-1 karena penyimpanan tersebut bersifat lokal.
   vii. Ketika mulai menurunkan DFD dari level tertinggi, cobalah untuk mengidentifikasi *external events* dimana sistem harus memberikan respon. *External events* dalam hal ini berarti suatu kejadian yang berkaitan dengan pengolahan data di luar sistem, dan menyebabkan sistem memberikan respon.
2) Jangan menghubungkan langsung antara satu penyimpanan dengan penyimpanan lainnya (harus melalui proses).
3) Jangan menghubungkan langsung dengan tempat penyimpanan data dengan entitas eksternal / terminator (harus melalui proses), atau sebaliknya.
4) Jangan membuat suatu proses menerima input tetapi tidak pernah mengeluarkan output yang disebut dengan istilah "*black hole*".

5) Jangan membuat suatu tempat penyimpanan menerima input tetapi tidak pernah digunakan untuk proses.
6) Jangan membuat suatu hasil proses yang lengkap dengan data yang terbatas yang disebut dengan istilah "*magic process*".
7) Jika terdapat terminator yang mempunyai banyak masukan dan keluaran, diperbolehkan untuk digambarkan lebih dari satu sehingga mencegah penggambaran yang terlalu rumit, dengan memberikan tanda asterik (*) atau garis silang (#), begitu dengan bentuk penyimpanan.
8) Aliran data ke proses dan keluar sebagai output keterangan aliran data berbeda.[9]

2.5.3. DFD Fisik

Adalah representasi grafik dari sebuah sistem yang menunjukan entitas-entitas internal dan eksternal dari sistem tersebut, dan aliran-aliran data ke dalam dan keluar dari entitas-entitas tersebut. Entitas-entitas internal adalah personel, tempat (sebuah bagian), atau mesin (misalnya, sebuah komputer) dalam sistem tersebut yang mentransformasikan data. Maka DFD fisik tidak menunjukkan apa yang dilakukan, tetapi menunjukkan dimana, bagaimana, dan oleh siapa proses-proses dalam sebuah sistem dilakukan.

Perlu diperhatikan didalam memberikan keterangan di lingkaran-lingkaran (simbol proses) dan aliran-aliran data (simbol aliran data) dalam DFD fisik menggunakan label/keterangan dari kata benda untuk menunjukan bagaimana sistem mentransmisikan data antara lingkaran-lingkaran tersebut.[10]

2.5.4. DFD Logis

Adalah representasi grafik dari sebuah sistem yang menunjukkan proses-proses dalam sistem tersebut dan aliran-aliran data ke dalam dan ke luar dari proses-proses tersebut. Kita menggunakan DFD logis untuk membuat dokumentasi sebuah sistem informasi karena DFD logis dapat mewakili logika tersebut, yaitu apa yang dilakukan oleh sistem tersebut, tanpa perlu menspesifikasi dimana, bagaimana, dan oleh siapa proses-proses dalam sistem tersebut dilakukan

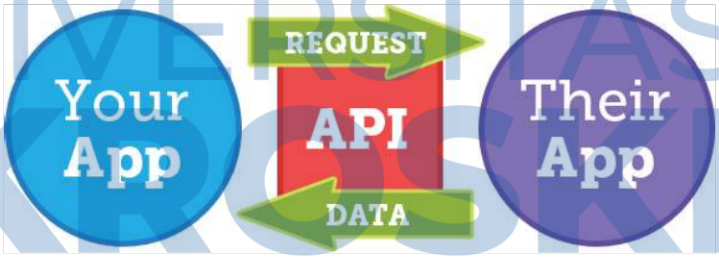
Keuntungan dari DFD logis dibandingkan dengan DFD fisik adalah dapat memusatkan perhatian pada fungsi-funsi yang dilakukan sistem.

Beberapa hal yang umum yang mendapat perhatian dalam mendesain baru tersebut ialah:

1. Menggabungkan beberapa tugas menjadi Satu
2. Master Detail Update
3. Meminimalkan tugas-tugas yang tidak penting
4. Menghilangkan tugas-tugas yang duplikat
5. Menambahkan proses baru
6. Meminimalkan proses input
7. Menetapkan bagian mana yang harus dikerjakan komputer dan bagian mana yang harus dikerjakan manual [10]

2.6. API (*Application Programming Interface).*

API merupakan sekumpulan sintak yang berisi perintah atau fungsi yang dapat digunakan untuk berinteraksi dengan sistem operasi tertentu atau program pengendalian lainnnya misalnya *Database Management System* (DBMS).



Gambar 2 1 Ilustrasi cara kerja API

Sebuah API dapat diimplementasikan dengan menulis sintaks dalam program yang menyediakan sarana untuk meminta layanan program tersebut. Sebagai contoh, facebook menyediakan API sehingga para pengembang website dapat mengintegrasikan komentar di websitenya langsung melalui komentar akun facebook si pengunjung, atau yang lebih

spesifik lagi API facebook juga dapat digunakan untuk membuat fungsi *auto post* artikel ke facebook saat artikel di website ditambahkan. Semua ini dapat dilakukan karena facebook menyediakan program API untuk dapat mengakses sebagian atau beberapa fungsi dari program facebook. Begitu juga untuk API dari layanan lainnya seperti JNE, POS dst.

Lahirnya konsep API telah membawa banyak perubahan pada proses kemudahan dalam mengintegrasikan berbagai layanan aplikasi, namun perlu dicatat bahwa konsep API adalah antarmuka *software-to-software*, bukan merupakan sebuah *user interface*. API memungkinkan sebuah aplikasi berbicara satu sama lain tanpa sepengetahuan pengguna.

Terdapat beberapa jenis API yang sering digunakan dalam pengembangan perangkat lunak. Berikut ini adalah beberapa jenis API :

1. *Document Object Model* (DOM)

Document Object Model (DOM) adalah antarmuka pemrograman (API) untuk HTML, XML dan SVG dokumen. Ini memberikan representasi terstruktur dokumen sebagai pohon. DOM mendefinisikan metode yang memungkinkan akses ke pohon, sehingga mereka dapat mengubah dokumen struktur, gaya dan konten. DOM menyediakan representasi dokumen sebagai kelompok terstruktur *node* dan objek, memiliki berbagai properti dan metode. *Node* juga dapat memiliki *event handler* yang melekat pada mereka, dan setelah peristiwa dipicu, *event handler* dijalankan. Pada dasarnya, menghubungkan halaman web untuk *script* atau bahasa pemrograman.

Meskipun DOM sering diakses menggunakan *JavaScript*, ini bukan merupakan bagian dari bahasa *JavaScript*. Hal ini juga dapat diakses oleh bahasa lain.

2. Device API

Merupakan seperangkat APIs yang memungkinkan kita untuk mengakses ke seluruh fitur *hardware* yang tersedia ke halaman Web dan aplikasi. Misalnya :

a. *Ambient Light Sensor API*

Berfungsi untuk mendeteksi perubahan intensitas cahaya.

b. *Battery Status* API

Berfungsi untuk mengakses informasi tentang jumlah daya baterai pada perangkat.

c. *Geolocation* API.

Memungkinkan pengguna untuk memberikan lokasi mereka ke aplikasi web jika mereka menginginkannya. Untuk alasan privasi, pengguna diminta untuk mengizinkan untuk melaporkan informasi lokasi.

d. *Pointer Lock* API.

Menyediakan metode masukan berdasarkan gerakan *mouse* beberapa waktu, bukan hanya posisi absolut dari kursor *mouse* di *viewport*. Ini memberi Anda akses ke gerakan *mouse* secara baku, mengunci target *event mouse* untuk elemen tunggal, menghilangkan batas berapa gerakan mouse jauh bisa pergi dalam satu arah, dan menghilangkan kursor dari pandangan. Ini sangat ideal untuk pengembangan game FPS (*First person shooter)*.

e. *Proximity* API.

Adalah cara praktis untuk mengetahui kapan pengguna berada dekat dengan perangkat. Peristiwa ini memungkinkan untuk bereaksi terhadap perubahan tersebut, misalnya dengan menutup layar smartphone ketika pengguna memiliki panggilan telepon dengan perangkat dekat telinga mereka.

f. *Device Orientation* API.

Berfungsi untuk perangkat agar mampu menentukan orientasi perangkat yaitu, perangkat dapat melaporkan perubahan data yang menunjukkan orientasi mereka dengan hubungan dengan tarikan gravitasi. Secara khusus, perangkat genggam seperti ponsel dapat menggunakan informasi ini untuk secara otomatis memutar layar untuk tetap tegak, menyajikan tampilan layar lebar dari konten web bila perangkat diputar sehingga lebarnya lebih besar dari tingginya.

g. *Screen Orientation* API.

Berbeda dengan orientasi perangkat, walaupun perangkat tidak didukung untuk orientasi perangkat, sebuah perangkat tetap memiliki Orientasi layar. API ini akan

mendeteksi posisi layar, secara umum terbagi dua, yaitu *portrait* maupun *landscapre.*

h. *Vibration* API.

API *Vibration* memungkinkan untuk mengendalikan perngkat agar melakukan getaran (digunakan untuk perangkat *mobile* seperti handphone dan tabel*t*).

3. Communication APIs

API ini memungkinkan halaman Web dan aplikasi berkomunikasi dengan halaman lain atau perangkat tertentu. Misalnya :

a. *Network Information* API

Memberikan informasi tentang koneksi sistem, yang dalam istilah jenis koneksi umum (misalnya, wifi, seluler, dll). Ini dapat digunakan untuk memilih konten definisi tinggi atau konten definisi rendah berdasarkan koneksi pengguna. Seluruh API terdiri dari penambahan antarmuka *Network Information* dan properti tunggal untuk antarmuka *Navigator*

b. *Web Notifications*

Notifications API memungkinkan sebuah halaman web atau aplikasi mengirimkan pemberitahuan yang ditampilkan di luar halaman pada tingkat sistem; ini memungkinkan aplikasi web mengirimkan informasi ke pengguna bahkan jika aplikasi *idle* atau di latar belakang. Artikel ini membahas dasar-dasar penggunaan API ini di aplikasi Anda sendiri.

4. Data management APIs

Data pengguna dapat disimpan dan di kendalikan menggunakan perangkat API ini. Misalnya :

a. *FileHandle* API

Filehandle API memungkinkan untuk memanipulasi *file*, termasuk menciptakan file dan memodifikasi konten mereka (tidak seperti *File* API). Karena file dimanipulasi

melalui API yang dapat secara fisik disimpan pada perangkat, bagian editing menggunakan mekanisme penguncian turn-based untuk menghindari masalah ras.

b. *IndexedDB*.

IndexedDB adalah API tingkat rendah untuk penyimpanan sisi klien dari sejumlah besar data terstruktur, termasuk *file* / *blobs*. API ini menggunakan indeks untuk mengaktifkan pencarian kinerja tinggi data ini.[7]