# BAB II LANDASAN TEORI

Penulis akan mencoba mengkaji informasi dari penelitian sebelumnya untuk dijadikan bahan perbandingan, dengan mengetahui kelebihan dan kekurangan dari peneliti sebelumnya. Selain itu peneliti menggunakan informasi dari *e-book* dan laporan tugas akhir sebelumnya, tentang teori yang berkaitan dengan judul tugas akhir ini.

## Penelitian Terdahulu

### Analisis Perhitungan Harga Pokok Produksi Sarung Tenun

Menurut Wulansari Dkk (2016), hasil evaluasi hasil perhitungan harga pokok produksi dalam menentukan harga jual dengan metode *Activity Based Costing*, untuk studi kasus pada perusahaan Edytex Jaya Pekalongan. Menunjukan hasil perhitungan harga pokok produksi dengan metode *Activity Based Costing* terdapat selisih perhitungan, yaitu pada sarung tenun dewasa terjadi terjadi selisih harga sebesar Rp.8.428 dan sarung sarung tenun anak sebesar Rp.9.234 dengan tingkat laba yang diharapkan kan dengan nominal 35% untuk sarung tenun dewasa, sedangkan nominal 30% untuk sarung tenun anak.

### Aplikasi Sistem Pendukung Keputusan Layanan Teknologi Informasi

Hasil dari penelitian ini, sistem pendukung keputusan untuk perhitungan biaya produksi jasa layanan teknologi informasi berdasarkan aktivitas menggunakan metode *Activity Based Costing*. Dengan menggunakan metode nilai bersih saat ini yaitu (*Net Present Value*), arus kas yang masuk dari hasil penjualan produk teknologi informasi dikurangi dengan beban operasional departemen TI dapat digunakan untuk mengambil keputusan terhadap pelaksanaan rencana investasi TI. Harga jual jasa layanan TI menjadi tinggi dikarenakan adanya biaya beban operasional departemen TI, departemen TI harus menambah jenis jasa layanan TI yang sesuai dengan jalur bisnis organisasi (*Line of Business*). Dikarenakan biaya *overhead* yang timbul akan dibagi dengan jumlah produk yang dihasilkan. Semakin banyak jenis-jenis layanan akan semakin besar pemakai layanan, dan harga jual akan semakin rendah.

Sistem jual beli layanan antar departemen dengan beberapa organisasi untuk menghasilkan produk akhir, akan mudah ditelusuri. Hal ini berdampak langsung pada penentuan biaya produksi, sehingga secara keseluruhan akan meningkatkan daya saing antar perusahaan karena dapat meningkatkan efisiensi biaya (Edi Widodo, 2017).

### Penerapan Activity Based Costing Untuk Menentukan Harga Pokok Produksi

Putu Pande Yudiastra (2018). Dari penelitian yang dilakukan, maka dapat disimpulkan penerapan metode *Activity Based Costing* untuk penentuan harga produksi dengan menggunakan variable beberapa diantaranya adalah: setiap jenis produk membutuhkan jumlah jam kerja yang disesuaikan, membutuhkan jumlah tanah liat yang cukup, jumlah laba atau keuntungan, batasan jam kerja, dan batasan bahan baku. Hasil penerapan *Activity Based Costing* yang dihasilkan adalah hasil dari perhitungan harga pokok produksi per jenis produknya.

## Sistem Pendukung Keputusan

Sistem Pendukung keputusan adalah sistem yang digunakan untuk dapat mengambil keputusan pada situasi terstruktur dan tidak terstruktur, dibantu menggunakan komputer dengan menggunakan data dan model tertentu untuk menyelesaikan beberapa suatu masalah yang ada. Sistem Pendukung Keputusan diterapkan juga pada perusahaan atau organisasi bukan untuk menggantikan tugas-tugas pengambil keputusan, tetapi merupakan sarana yang membantu bagi mereka dalam pengambilan keputusan lebih tepat.

Dalam penerapan Sistem Pendukung Keputusan, hasil dari beberapa keputusan sistem bukanlah hal yang menjadi dasar dalam pengambilan keputusan. Semua keputusan tetap berada pada kendali seorang, sistem hanya menghasilkan keluaran yang mengkalkulasi data-data sebagaimana pertimbangan seorang dalam pengambil keputusan. (Wibowo, 2011).

Sistem Pendukung Keputusan dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasikan masalah, memilih data yang relevan, dan menentukan pendekatan yang digunakan dalam proses pengambilan keputusan sampai dengan mengevaluasi pemilihan yang ada (Fitriani, 2012).

## Harga Pokok Produksi

Harga Pokok Produksi adalah daftar biaya produksi yang harus dikeluarkan perusahaan pada periode tertentu. Di dalamnya meliputi biaya bahan baku, alat produksi, dan bahan pendukung produksi lainnya. Jika harga pokok sudah ditentukan, pihak perusahaan akan lebih mudah untuk menentukan harga produk.

Menurut Supriyono (2013), menyatakan kalau harga pokok produksi merupakan jumlah uang yang akan dibayarkan dalam rangka untuk memiliki produk atau jasa yang diperlukan perusahaan.

## Metode *Activity Based Costing*

*Activity Based Costing* adalah sebuah metode yang bisa dilakukan untuk menghitung suatu biaya dan dapat mengidentifikasi biaya berdasarkan aktivitas. Fungsi lain dari metode tersebut adalah untuk mengalokasikan seluruh biaya dari berbagai sumber daya yang dimanfaatkan ketika menjalankan kegiatan produksi untuk menentukan harga yang lebih akurat. Sistem *Activity Based Costing* juga dapat menetapkan biaya tidak langsung suatu pabrik untuk memproduksi produk dengan cara yang lebih logis daripada pendekatan tradisional.

**Kelebihan *Activity Based Costing***

1. Dapat memperluas jumlah himpunan biaya yang bisa digunakan untuk mengumpulkan biaya *overhead*.
2. Dana yang dialokasikan akan berdasarkan kegiatan yang dihasilkan, bukan pada tingkat volumenya
3. Mampu mengubah beberapa sifat biaya tidak langsung, sehingga akan membuat biaya tersebut menjadi lebih mudah dilacak

**Langkah-Langkah penyelesaian *Activity Based Costing***

1. Mengklasifikasi dan mengidentifikasi beberapa aktivitas dan biaya sesuai kebutuhan. Berbagai aktivitas diklasifikasi ke dalam beberapa kelompok, disesuaikan proses produksi yang dikelola. Sementara itu berbagai biaya akan dihubungkan dengan setiap kelompok aktivitas, yaitu aktivitas berlevel unit, *batch*, produk, dan fasilitas.
2. Menghitung dan menentukan *cost driver* untuk masing – masing aktivitas.
3. Penentuan *Cost pool* akan masuk ke dalam suatu kelompok biaya aktivitas overhead yang harus berhubungan secara logis dan harus mempunyai rasio konsumsi yang untuk setiap produk.
4. Menentukan *Pool rate* untuk masing - masing *Cost pool*.

*Pool rate* = A : B

*Pool rate*, A = *Cost pool,* B = Kapasitas *Cost driver* untuk semua produk

1. Menelusuri biaya tiap *cost pool* homogen ke produk yang dibebankan.

Biaya *overhead* pabrik = *Pool rate* x *Cost pool*

*Pool rate,* C = *Cost driver* per produk tertentu

1. Setelah menghitung semua biaya *overhead*, harga pokok produksi untuk masing-masing jenis produk dapat dicari.

Harga Pokok Produksi = X1 + X 2 + X3

Dimana Harga Pokok

X1 = Biaya bahan langsung, X 2 = Biaya tenaga kerja langsung

X3 = Biaya *overhead* pabrik

## UML (*Unified Modeling Language*)

UML atau “*Unified Modeling Language*” adalah suatu metode pemodelan visual yang berfungsi untuk sarana perancangan sistem berorientasi objek atau yang kita kenal OOP.

Menurut Windu Gata, Grace (2013:4), *Unified Modeling Language* (UML) merupakan spesifikasi standar yang digunakan untuk membangun sebuah perancangan perangkat lunak. Metode tersebut merupakan suatu susunan dalam pembangunan suatu alat yang mendukung pengembangan sistem.

UML (*Unified Modelling Language*) merupakan salah satu standar bahasa yang banyak digunakan didunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain (Rosa Dan Shalahuddin, 2015:133)

### *Use Case Diagram*

*Use case diagram* adalah suatu urutan interaksi yang saling berkaitan antara sistem dan aktor. Menggunakan sebuah teknik dalam perancangan sebuah *software* maupun sebuah sistem informasi untuk melengkapi sebuah sistem yang bersangkutan, *use case* menunjukkan interaksi yang dilakukan antara aktor, dari interaksi sistem dengan sistem yang sudah ada, sebuah *use case* didesain sesuai dengan urutan langkah yang sederhana.

Sistem beraksi dan bereaksi terhadap perilaku sistem, yang dimaksud perilaku ini merupakan aktifitas sistem yang dapat dilihat dari luar dan dapat diuji. *Use case* sendiri menjelaskan sebuah lingkungan sistem, serta hubungan antara sistem dengan sistem yang lainnya. Manfaat *use case* sendiri adalah:

1. Memudahkan hubungan dengan domain *expert* dan *end user.*
2. Memastikan pemahaman yang tepat tentang kebutuhan sistem.
3. Adanya sebuah *interface* yang dimiliki sistem.
4. Dapat dipergunakan untuk verifikasi.

Sedangkan dari segi karakteristik, karakteristik dari *use case* adalah sebagai berikut:

1. *Use case* merupakan hubungan antara sistem dengan aktor, beserta tindakan yang dilakukan saling berhubungan.
2. *Use case* dipelopori oleh pengguna dan mungkin melibatkan peran aktorlain.
3. *Use case* mempunyai perluasan yang menjelaskan tindakan khusus dalam interaksi.

Untuk mendukung pembuatan *use case*, penjelasan mengenai komponen yang terdapat pada *use case* sebagai berikut:

**Tabel 2.1** Komponen *Use Case*

|  |  |  |
| --- | --- | --- |
| **No** | **Komponen** | **Keterangan** |
| 1 |  | ***Actor*,** seorang yang akan berinteraksi dengan sistem, dimana hanya dapat menginputkan informasi dan menerima informasi dari sistem dan tidak memegang kendali pada *use case. Actor* tersebut di gambarkan dengan *stickman*. |
| 2 |  | ***Use case*,** fungsional sistem yang akan di buat, dengan ini *user* akan lebih paham penggunaan sebuah sistem. |
| 3 |  | ***Association*,** penghubung diatara elemen yang menghubungkan antara a*ctor* dan u*se case*. |
| 4 |  | ***Generalization*,** objek anak berbagi perilaku dan struktur data dari objek yang ada diatasnya induk. |
| 5 |  | ***Dependency*,** hubungan pada suatu elemen mandiri yang akan mempengaruhi elemen yang bergantung pada elemen tidak mandiri. |
| 6 |  | ***Include*,** spesifikasikan bahwa *use case* adalah fungsionalitas dari *use case* lainnya*.* |
| 7 |  | ***Extend*,** bahwa target *use case* diperluas perilaku dari *use case* lainnya pada suatu titik yang diberikan. |
| 8 |  | ***System*,** penggambaran menunjukan paket yang akan ditampilkan sistem secara terbatas. |
| 9 |  | ***Collaboration*,** hubungan dari aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah elemennya. |
| 10 |  | ***Note*,** elemen fisik saat aplikasi dijalankan dan menampilkan suatu sumber daya komputasi. |

### *Activity Diagram*

***Activity diagram* sebuah teknik untuk menggambarkan prosedural, aktifitas sistem, jalur kerja dan aliran secara berurutan dari aktifitas tersebut. Dalam beberapa hal, diagram aktifitas satu jenis dengan UML atau sama persis dengan sebuah aliran data yang terkait dengan pendekatan berorientasi objek, akan tetapi perbedaan prinsip lebih mendukung kepada *behavior parallel*.**

***Diagram aktifitas*** adalah penggambaran tentang aktifitas pada prosedur logika. Alur diawal hingga diakhir, *activity diagram* menunjukkan langkah - langkah apa saja yang terjadi dalam proses sistem yang akan dibuat. Sehingga dengan menggunakan diagram ini dapat memudahkan dalam proses pembuatan sistem (Sukamto dan Shalahuddin, 2013).

**Tabel 2.2** Komponen Activity Diagram

|  |  |  |
| --- | --- | --- |
| **No** | **Komponen** | **Keterangan** |
| 1 |  | ***Activity*,** digambarkan dengan aktifitas yang dilakukan sistem, bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. |
| 2 |  | ***Initial status,*** sebuah diagram aktivitas dengan objek sebuah status awal. |
| 3 |  | ***Initial final,*** gambaran sebuah diagram aktivitas yang memiliki status di akhir. |
| 4 |  | ***Decision,*** digunakan untuk suatu percabangan keputusan atau tindakan dalam pemilihan aktivitas yang lebih dari satu. |
| 5 |  | ***Line Connector,*** berguna untuk menghubungkan antara satu komponen dengan komponen lainnya. |

### *Sequence Diagram*

*Sequence diagram* merupakan diagram yang menggambarkan kolaborasi dinamis antara beberapa objek. Berguna untuk penjabaran rangkaian pesan yang dikirim ke objek yang berhubungan antara objek lain atau sesuatu yang terjadi pada titik tertentu dalam menjalankan sistem.

**Tabel 2.3** Komponen *Sequence Diagram*

|  |  |  |
| --- | --- | --- |
| **No** | **Komponen** | **Keterangan** |
| 1 |  | ***Actor***, merepresentasikan entitasyang berada diluar sistem dan saling berinteraksi dengan sistem lain. |
| 2 |  | ***Lifeline,*** objek antarmuka yang saling berinteraksi dan tempat mulai dan berakhirnya sebuah *message*. |
| 3 | Message() | ***Message,*** berfungsi sebagai komunikasi antar objek yang memuat informasi tentang aktivitas yang terjadi. |
| 4 | Message() | ***Self Message,*** simbol ini menunjukkan pesan objek hendak memanggil dirinya sendiri untuk menunjukan urutan kejadian. |
| 5 |  | ***LifeLine Boundary,*** Digunakan untuk menggambarkan sebuah penggambaran *form*. |
| 6 |  | ***Control class,*** menggambarkan objek yang digunakan untuk penghubung *boundary* dengan *table*. |
| 7 |  | ***Entity Class,*** Digunakan untuk menghubungkan setiap kegiatan yang akan dilakukan. |

### Class Diagram

Sukamto dan Shalahuddin (2013:141), “*Class diagram* merupakan penjelasan tentang struktur sistem dari pendeskripsian yang akan dibangun dari tiap-tiap kelas untuk sistem. Adapun *class diagram* mempunyai sebuah metode dan atribut. Diagram *class* menyediakan tampilan aplikasi statis dan kemampuan pemetaan dengan berorientasi objek.

* 1. Variabel-variabel yang dimiliki sebuah kelas biasa diartikan dengan atribut.
  2. Metode merupakan fungsi fungsi yang dimiliki oleh suatu kelas.

Adapun beberapa susunan struktur yang harus memiliki jenis-jenis kelas sebagai berikut:

1. Kelas dapat memiliki fungsi terdahulu yang dilakukan ketika sistem dijalankan.
2. Fungsi kelas dapat menampilkan sistem *(view)* yang menjelaskan dan mengatur beberapa tampilan terhadap user.
3. Menjelaskan *use case (controller)* kelas yang harus berfungsi menangani pengambilan dari penjelasan sebuah *use case*, kelas ini disebut juga sebagai kelas proses untuk melayani proses aktivitas pada perangkat lunak.

**Tabel 2.4** Komponen *Class Diagram*

|  |  |
| --- | --- |
| **Gambar** | **Keterangan** |
|  | ***Class,*** gambaran dari objek yang berbagi atribut serta operasi yang sama. |
|  | ***Interface,*** objek ini digunakan sebagai konsep *interface* dalam tampilan pemrograman. |
|  | ***Association,*** relasi penghubung antar kelas dengan kelas lainnya, atau asosiasi *multiplicity* |
|  | ***Directed association,*** Relasi antar kelas digunakan oleh makna kelas yang lain |
|  | **Aggregation,** objek relasi penghubung antar kelas. |

## *Database*

### Pengertian *Database*

*Database* merupakan sekumpulan informasi atau data yang tersimpan secara sistematis di dalam komputer yang menggunakan suatu program untuk mendapatkan informasi dari basis data komputer. *Database* adalah kumpulan fakta representasi yang saling berkaitan, disimpan secara bersama sehingga memudahkan dalam pengelolaannya dan tanpa pengulangan yang tidak perlu, yang dapat memenuhi kebutuhan berbagai data (setiawan, 2020). Beberapa alasan mengapa diperlukan pembuatan atau membangun sebuah database diantaranya, membuat penyimpanan dan pengelolaan data menjadi lebih efisien, dapat memudahkan dalam mengatur data dalam jumlah banyak dan selalu bertambah, mempercepat dan mempermudah dalam mengidentifikasi data, serta mudah membuat sistem data dengan mengelompokan dan menyimpan data secara terstruktur.

### Entity Relationship Diagram

*Entity Relationship Diagram* (ERD) adalah suatu model statis suatu sistem struktural yang menjelaskan hubungan antar data dan biasa digunakan dalam desain sebuah *database*. Banyak komponen ERD seperti simbol dan objek dasar yang dapat menghubungkan antar relasi data.

Menurut Yanto (2016:32) “ERD merupakan suatu diagram untuk menggambarkan desain dari model konseptual suatu *database*. Dalam gambaran antara objek yang dapat berelasi dengan objek yang lain, objek ini seperti di dunia nyata yang sering dikenal dengan hubungan antar entitas.

*Entity Relationship Diagram* mempunyai kegunaan dalam dimodelkan struktur dan hubungan antar data, dan memiliki beberapa komponen penyusun sebagai berikut:

* 1. Entitas

Merupakan kumpulan objek yang dapat diidentifikasikan secara unik dan berbeda, objek data tersebut yang terangkum sebagian dari data utama yang diperjelas data-data lainnya Sutanta (2011). Saat membuat ERD, biasanya entitas digambarkan dalam simbol persegi panjang. Sedangkan disisi lain, ada juga entitas yang dinilai lemah digambarkan dengan simbol persegi panjang kecil di dalam persegi panjang besar. Keberadaan setiap entitas pasti memiliki perbedaan, jika ada kesamaan satu dengan lainnya, maka entitas tidak bisa dituliskan.

* 1. Atribut

Semua entitas selalu mempunyai elemen atau atribut untuk mendeskripsikan karakteristik dari setiap entitas. Gambaran atribut diwakili sebagai simbol elips, didalam ERD ada beberapa atribut contohnya adalah atribut *simple*, atribut kunci, atribut multinilai, atribut *derivative* dan atribut gabungan.

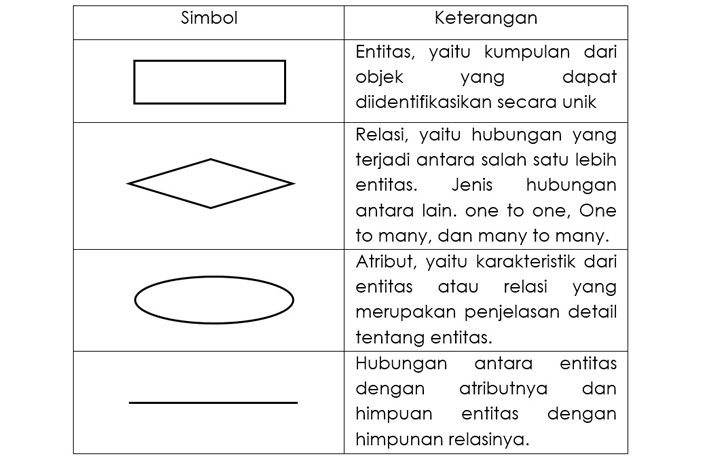
* 1. Relasi

Relasi atau hubungan merupakan sebuah keterkaitan dari sejumlah atau beberapa entitas yang berasal dari himpunan lain. Gambaran relasi diwakili oleh simbol belah ketupat, relasi juga terbagi menjadi beberapa jenis seperti *One to One*, *One to Many*, dan yang terakhir *Many to Many*.

* 1. Garis

Setiap entitas mempunyai garis penghubung antar atribut untuk menunjukan hubungan entitas pada diagram ER. Garis ini dapat memudahkan pengguna untuk melihat dan mengetahui bagaimana flow suatu ERD dimulai dan sampai *flow* berakhir

**Tabel 2.5** Tabel *Entity Relationship Diagram*



## Bahasa Pemrogaman

### PHP

PHP (*Hypertext Preprocessor)* adalah salah satu bahasa *open source* bagian dari bahasa *script* yang dikhususkan untuk pengembangan web dan dapat ditanamkan atau disisipkan kedalam HTML. Bisa dikatakan PHP sebagai sebuah *server-side script language* artinya sintaks dan perintah yang *user* berikan akan dijalankan oleh web *server* menjadi kode HTML

Menurut Hikmah, dkk (2015:1) “*Hypertext Preprocessor* atau yang disebut dengan PHP, tergolong sebagai perangkat lunak *open source* diatur dalam aturan *general purpose licences*. Bahasa pemrograman PHP cocok digunakan dalam lingkungan web, karena dapat diletakan pada *script* HTML atau sebaliknya. Beberapa keuntungan menggunakan pemrograman PHP adalah:

1. PHP bisa diintegrasikan dengan beberapa *database* umum seperti MySQL, Oracle, Sybase, Informix, dan Microsoft SQL Server.
2. Bahasa Pemrograman PHP dapat dijalankan pada sisi server yang dapat digunakan untuk mengelola konten dinamis, *database*, bahkan dapat membangun sebuah situs web besar.
3. Mendukung banyak protokol besar seperti IMAP, POP3, dan LDAP. Sintaks pada PHP juga sama seperti bahasa pemrogaman C.

### JAVA

Java ini merupakan salah satu bahasa pemrogaman yang bisa dibilang *powerful* karena bahasa pemrogaman ini bersifat *multiplatform*. Pemograman Java ini berorientasi pada objek, sementara programnya tersusun dari kelas-kelas atau bagian tertentu. Fitur pemrogaman objek ini dapat memudahkan dalam pembuatan aplikasi kompleks, namun di sisi lain membuat bahasa java relatif lebih susah dipelajari terutama bagi pemula. Berbeda dengan bahasa pemrogaman yang mendukung pemrogaman procedural seperti C++, PHP atau Python, bahasa java hanya menyediakan konsep objek.

### JSON

*JavaScript Object Notation* atau JSON merupakan bagian (subset) dari Javascript. JSON adalah format pertukaran dan penyimpanan data yang ringan, bagi penggunanya JSON mudah dibaca dan ditulis, serta mudah diuraikan dan dibuat (*generate*) oleh komputer. Selain itu JSON berguna untuk mengirimkan data melalui jaringan sebagai string, meski perlu dikonversi ke objek JavaScript saat anda mengakses data. Format teks JSON sepenuhnya tidak bergantung pada pada bahasa tetapi menggunakan konvensi yang umum bagi pengguna, yaitu menggunakan bahasa pemrogaman C-family, termasuk C, C++, Java, Javascript, Python, dan banyak lainnya.

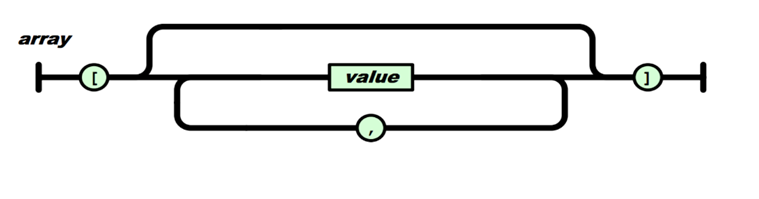
JSON memiliki fungsi format yang menyimpan informasi terstruktur dan biasanya digunakan untuk mengirim data antara server dengan klien. Selain itu JSON terbuat dari dua struktur:

a. Sekumpulan nilai dari beberapa bahasa, yaitu sebagai *object, record*, struktur, tabel *hash*, daftar berkunci, atau *associative array*.

b. Daftar nilai terurutkan pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*). Struktur-struktur data ini disebut sebagai struktur data *universal*. Semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama. Struktur dan rangkaian JSON menggunakan bentuk sebagai berikut:

1. Objek

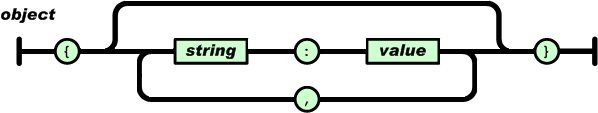
Sepasang nama atau nilai yang tidak terurutkan. Sebuah objek akan dimulai dengan {(kurung kurawal buka) dan diakhiri dengan} (kurung kurawal tutup). Setiap nama diikuti dengan: (titik dua) dan setiap pasangan nama atau nilai dipisahkan oleh , (koma).



Gambar 2.1 Objek

1. Larik

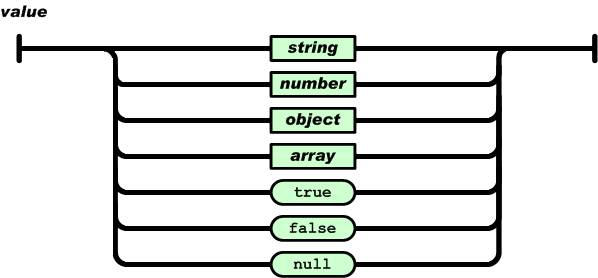
Sekumpulan nilai yang terurutkan. Dimulai dengan [ (kurung kotak buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

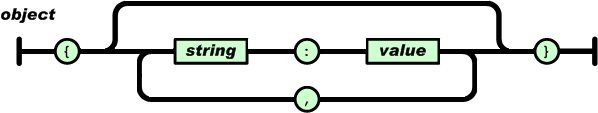


Gambar 2.2 Larik

1. Nilai

Nilai berupa sebuah *string* dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau larik. Struktur tersebut dapat disusun bertingkat.

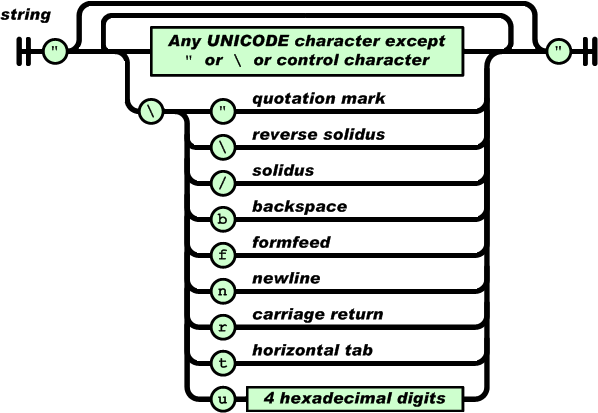




Gambar 2.3 Nilai JSON

1. String

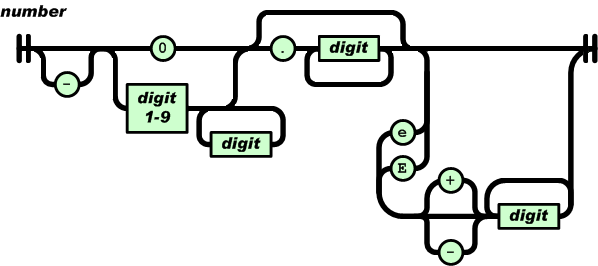
Kumpulan dari nol atau lebih karakter *Unicode*, yang diawali dengan tanda kutip ganda. Dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter tertentu. *String* sangat mirip dengan *string* C atau Java.

**

Gambar 2.4 String JSON

1. Angka

Mirip dengan angka di C atau *Java*, kecuali format heksadesimal dan oktal tidak dapat digunakan.



Gambar 2.5 Angka JSON