BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

1.1 Tinjauan Pustaka

Dalam perkembangan teknologi saat ini, pencarian (*searching)* menjadi aplikasi yang sangat penting bagi pengguna (*user*). Hasil pencarian yang tidak sesuai dengan keinginan pengguna menjadi masalah bagi pengguna dan teknologi informasi. Semantik *search* banyak digunakan dalam masalah memahami arti dari metadata atau informasi. Selain semantik *search web* API juga mempunyai keuntungan bagi programer maupun pengguna, karena *web* API dapat di jalankan oleh sistem operasi mana saja. Banyak penelitian yang menggunakan *web* API atau pun semantik *search* untuk membangun sebuah web.

Jevier Zebua dan Metty Mustikasari (2012) dalam penelitiannya yang berjudul Aplikasi Pencarian Buku Berbasis *Web* Semantik Untuk Perpustakaan SMK YADIKA 7 Bogor menyatakan bahwa dengan *semantic web* data berbasis HTML dapat diubah menjadi format yang dapat dipahami oleh mesin, sehingga dapat melakukan proses pengumpulan informasi dan memahami hubungan antara informasi. Metode yang digunakan oleh Jevier Zebua dan Metty Mustikasari adalah metode studi pustaka. Mereka mempelajari tentang teori-teori yang berkaitan dengan *web* semantik, lalu pembuatan *query* setelah itu uji coba dan melakukan evaluasi.

Carolina, Devina (2016) dalam penelitiannya yang berjudul Pembangunan *Web* API Terintegrasi Untuk Destinasi Pariwisata Yogyakarta, menggunkan *web* sebagai

layanan penyedia data pariwisata Yogyakarta untuk memenuhi kebutuhan *developer* dan menyediakan informasi pariwisata Yogyakarta bagi pengunjung *web*. Selain itu Carolina, Devina juga menggunakan aplikasi mobile Jogja Wisata sebagai aplikasi *client* yang menggunakan *web service*, berjalan pada sistem operasi android, dan dapat berkolaborasi mengunggah data ke aplikasi penyedia *service* tersebut. *Web* API dibangun untuk menyediakan *web service* berupa API tempat wisata, hotel, restaurant, fasilitas umum lainnya, kabupaten, kecamatan, fasilitas di sekitar tempat wisata di Yogyakarta bagi *developer* dan informasi pariwisata Yogyakarta bagi pengunjung *web* JogjaParadise.

Dalam penelitian ini menggunakan MySQL sebagai *database*. MySQL digunakan karena menurut Carolina, Devina MySQL cepat, dipercaya dan mudah dalam penggunaannya sehingga cocok digunakan untuk mengakses data di internet. Selain MySQL penelitian ini juga menggunakan PHP, JSON dan juga *CodeIgniter*.

Sirojul Munir (2016) dalam penelitiannya berjudul Perancangan Sistem Informasi Pencatatan Meteran Air PDAM Berbasis *Web* Menggunakan *Framework* MVC, memiliki tujuan untuk mengetahui implementasi penggunaan *web framework* berbasis *Model View Controller*. MVC (*Model View Controller*) digunkan untuk membangun aplikasi web *service*. Selain itu menggunakan MVC karena MVC mempunyai beberapa manfaat yakni perubahan pada kode program pada *user* dapat dilakukan tanpa mempengaruhi yang lain, dan desainer dapat bekerja tanpa mengkhawatirkan pengolahan dan penyimpanan data. Pada penelitian ini menggunakan *tools plugin POSTMAN* untuk proses pengujian *web service*.

Sedangkan pada penelitian yang akan dilakukan menggunakan *web* API sebagai pengujian *service*nya.

Nava'atul Fadillah, Novrido Charibaldi dan Herlina Jayadianti (2010) dalam penelitiannya yang berjudul Penerapan Teknologi *Semantic Web* Pada Aplikasi Pencarian Koleksi Perpustakaan (Studi Kasus: Perpustakaan FTI UPN "Veteran" Yogyakarta) menggunakan metodologi penelitian *waterfall* yaitu analisis, *design*, implementasi serta pengujian. Pada penelitian ini menggunakan bahasa pemprograman JSP (*Java Server Pages*) untuk merancang *interface*-nya dan menggunakan *protege* 3.4 dan JENA API sebagai *framework*-nya.

Pada penelitian yang di lakukan di perpustakaan Universitas Muhammadiyah Yogyakarta berfokus pada aplikasi pencarian buku dengan memanfaatkan teknologi *web* API sebagai *client service*-nya. Menggunakan MVC *web* API karena jika menggunakan API programnya dapat dijalankan di sistem operasi mana saja asalkan sudah ter-*install* API tersebut. Selain itu penggunaan MVC karena MVC memisahkan berdasarkan fungsinya, sehingga memudahkan *developer* dalam mengembangkan aplikasi yang akan dibangun. Dalam penelitian ini selain menggunakan teknologi *web* API dan MVC, juga menggunakan teknologi *ADO.Net* sebagai *framework database*nya. *Database* yang digunakan adalah *database stored procedure.*

1.2 Landasan Teori

2.2.1 Perkembangan *Web*

Web saat ini menjadi kumpulan data yang besar dan sangat berguna bagi pengguna internet, karena di dalam *web* terdapat dokumen-dokumen yang saling terhubung dan dapat di akses oleh pengguna melalui koneksi internet. Fenomena perkembangan web diawali dengan adanya *web* 1.0. web 1.0 merupakan *website* generasi pertama. Dalam *website* ini pengunjung internet hanya bisa melakukan pencarian (*searching*) dan melihat (*browsing*) informasi di internet. Menurut (Maulana Firdaus, 2013) Jadi didalam *web* 1.0 hanya terjadi komunikasi 1 arah dimana pembuat *website* dan penikmat (pengunjung) *website* hanya sebagai pembaca. Bahasa yang digunakan pada *web* 1.0 masih bahasa HTML saja.

Web 2.0 ini merupakan *web* kedua yang dimana pada *web* ini pengunjung sudah mulai dapat melakukan interaksi dengan diatur oleh sistem yang ada di *web*. Ciri dari *web* ini adalah *Share, Collaborate* dan *Exploit*. (Adadarmawan, 2014). Jadi pada *web* ini pengunjung sudah dapat merasakan kehidupan sosial di dunia maya. *Web* 2.0 hadir seiring maraknya pengguna *blog*, *Friendster*, *Myspace*, *Youtube* dan *Fickl*. Jadi pada web ini kehidupan sosial didunia maya benar-benar terasa.

Web 3.0 ini merupakan *web* generasi ketiga yang merupakan perkembangan lebih maju dari *web* 2.0. Menurut (Dias Taufik Rahman, 2015), didalam *web* ini kita bisa melakukan aktivitas di dunia maya seperti layaknya di dunia nyata. *Web* 3.0 mempunyai ciri seperti *Happen*, *Provide* dan *Suggest*. *Web* 3.0 merupakan sebuah realisasi pengembangan sistem kecerdasan buatan (*artifical intelegence*)

untuk menciptakan global metadata yang dapat dimengerti oleh sistem, sehingga sistem dapat mengartikan kembali data tersebut kepada pengunjung dengan baik.

2.2.2 Mesin Pencari (*Search Engine*)

Mesin pencari (*search engine*) adalah sebuah program komputer yang berfungsi untuk memudahkan pengguna dalam mencari informasi yang ada di dalam *web server* seperti layanan WWW (*World Wide Web*), FTP (*File Transfer Protocol*), *Mailing List*, atau *News Group*. Menurut (Doni Alip, 2016), hasil dari pencarian akan menampilkan data informasi yang tersimpan dalam *website*, *blog* dan forum tertentu. Hasil pencarian pun memiliki variasi data seperti bentuk tulisan, gambar dan video. Mesin pencari umumnya menampilkan data yang dibutuhkan oleh pengguna sesuai dengan kriteria yang dimasukkan oleh pengguna. Manfaat dari mesin pencari adalah untuk mempermudah *user* dalam mencari informasi secara cepat dan tepat. Karena *user* hanya perlu mengetikkan kata kunci yang di inginkan kemudian *website* akan menampilkan hasil dari kata kunci tersebut secara cepat.

2.2.3 Semantic Search

Semantic search adalah sebuah teknik pencarian yang memahami arti dari sebuah informasi berdasarkan metadata dan menghasilkan informasi yang akurasi. *Semantic search* juga tidak hanya menampilkan informasi berdasarkan tingkat populer tetapi menghasilkan data secara *semantic*. *Semantic* adalah makna kata. Jadi ketika kita mengetikkan kata kunci atau kalimat didalam *search engine* maka *semantic search* akan menampilkan hasil yang paling relavan dengan kalimat

tersebut. *Semantic search* berbeda dengan *search engine*. *Search engine* umumnya menggunakan *page rangking* untuk kriteria menampilkan hasil pencarian.

2.2.4 MVC *(Model View Controller)*

MVC merupakan arsitektur pemprograman yang terdiri atas tiga bagian, yakni *model*, *view* dan *controller*. Arsitektur MVC dipisahkan ke dalam tiga layer. Untuk membangun sebuah aplikasi seperti manipulasi data, *interface* atau tampilan dan bagian yang menjadi *control* dari sebuah aplikasi *website*. Berikut ini merupakan fungsi dan penjelasan *model*, *view* dan *controller* (Aguzrybudy, 2016):

a. *Model*

   Model berfungsi untuk menangani masalah yang berkaitan dengan *database* dan untuk memanipulasi data (*insert*, *update*, *delete*, *search*). Selain itu *model* menangani *validasi* dari *controller*. Model tidak dapat terhubung secara langsung dengan bagian *view*.

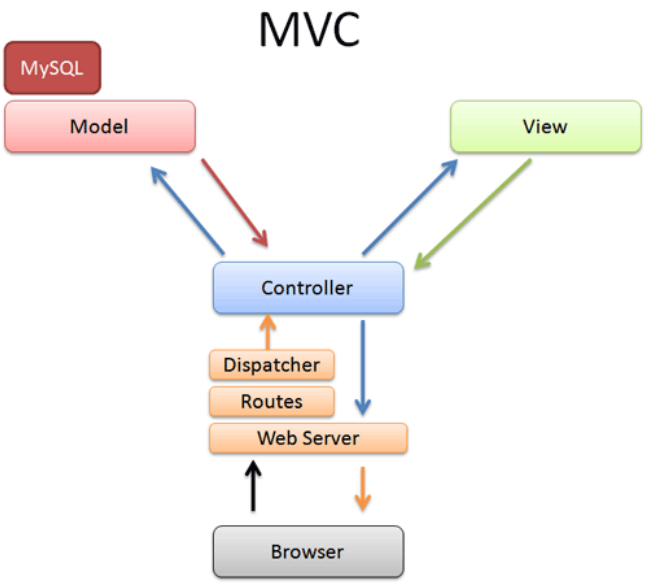
b. *View*

   View berfungsi untuk menangani masalah tampilan yang nantinya akan dilihat oleh *user* atau pengguna *website*. *View* tidak memiliki akses langsung ke *model* dan *view* diatur oleh *controller*.

c. *Controller*

   Controller berfungsi untuk menerima *request* dan kemudian menentukan apa yang akan di proses oleh aplikasi. *Controller* juga berfungsi mengatur hubungan antara *view* dan *model*.

Penjelasan MVC (*Model View Controller*) juga dapat dilihat pada gambar 2.1.



Gambar 2.1 Arsitektur *Model View Controller*

2.2.5 Web API (*Application Programming Interface*)

API (*Application Programming Interface*) merupakan sekumpulan perintah, fungsi, protokol yang digunakan *programmer* untuk membangun sebuah perangkat lunak atau (*software*) tertentu (Wirasetiawan, 2014). API juga digunakan untuk menggantikan bahasa yang digunakan dalam *system call* dengan bahasa yang lebih mudah dimengerti. API terdiri dari 2 (dua) *type*, yaitu *private* dan *public*. *Private* yang berarti hanya dapat di akses secara internal. *Public* yang berarti dipublikasikan secara umum.

Keuntungan menggunakan API adalah (Bayusetiawan, 2013):

1. Portabilitas: API dapat dijalankan dalam sistem operasi mana saja asalkan paket-paket API sudah ter*install*.

2. Lebih mudah dimengerti: API menggunakan bahasa yang lebih terstruktur dan mudah dipahami, hal ini penting dalam hal *editing* dan pengembangan.
3. Mudah dikembangkan: Dengan adanya API, memudahkan programmer untuk mengembangkan suatu *system*.

Web API yang bekerja di atas teknologi *web* yang menerima *request* dari *client* dan memberikan *response* yang sesuai dengan protokol tertentu.(Yanabatuwael, 2010) API berfungsi melengkapi bagian dari *web service. Web service* juga disebut *web* API. *Web service* merupakan sekumpulan data (*database*), *software* atau pun bagian dari *software* yang dapat diakses secara *remote*.(Wahidin Alambiyah, 2015). W*eb service* mempunyai layanan yang memungkinkan dua buah sistem yang saling *independent* dapat saling berkomunikasi seperti halnya *client server* melalui *protokol* HTTP.

2.2.6 XML

XML singkatan dari *Extended Markup Language* adalah bahasa yang digunakan untuk menyimpan data (tidak ada program) dan tidak tergantung dengan *tools* tertentu (seperti *editor, dbms,* dan *compiler*). XML merupakan suatu bahasa *Markup*. *Markup* yaitu bahasa yang berisikan kode-kode berupa tanda-tanda tertentu dengan aturan untuk memformat *dokumen* teks dengan *tag* sendiri agar dapat dimengerti. (Novia Asrumiati, 2013). XML merupakan kelanjutan dari HTML (*HyperText Markup Language*) yang merupakan bahasa standar untuk melacak internet.

XML mempunyai tiga tipe file:

1. XML, merupakan standar format dari struktur berkat (*file*) yang ada.
2. XSL, merupakan standar untuk memodifikasi data yang diimpor atau diekspor.
3. XSD, merupakan standar yang mendefinisikan struktur *database* dalam XML.

Keunggulan XML adalah sebagai berikut:

1. Pintar (*Intelligence*). XML dapat menangani berbagai tingakat (*level*) kompleksitas.
2. Dapat beradaptasi. Dapat mengadaptasi untuk membuat bahasa sendiri. Seperti *Microsoft* membuat bahasa MSXML atau *Macromedia* mengembangkan MXML.
3. Mudah Pemeliharaannya.
4. Sederhana. XML lebih sederhana.
5. Mudah dipindah-pindahkan (*Portability*). XML mempunyai kemudahan perpindahan (portabilitas) yang lebih bagus.

2.2.7 JSON

JSON (*JavaScript Object Natation*) adalah format pertukaran data yang ringan, mudah dibaca dan diimplementasikan oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. JSON merupakan bagian dari bahasa pemprograman *JavaScript* (Standar ECMA-262 Edisi ke-3 –Desember 1999). JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman

apapun karena menggunakan bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, *Java*, *JavaScript*, *Perl*, *Python* dan sebagainya. Kelebihan inilah yang membuat JSON menjadi ideal sebagai bahasa pertukaran data.

JSON dibuat dari dua struktur:

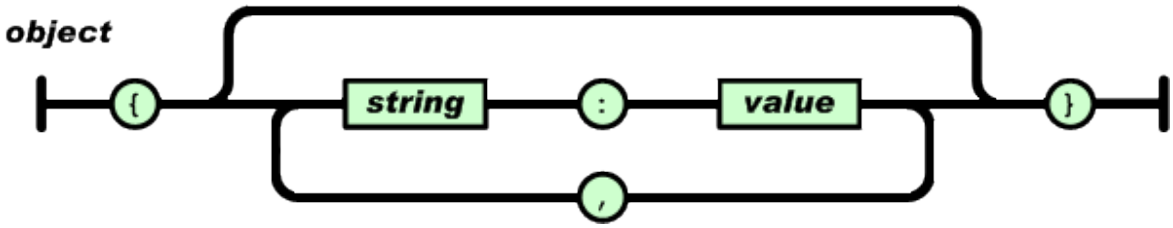
1. Kumpulan pasangan nama atau nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.

2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*). Vaktor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Hampir semua bahasa pemprograman mendukung penuh JSON dalam berbagai format. Hal ini memungkinkan format data yang dapat diperuntukan menggunakan bahasa pemprograman juga menggunakan dasar dari struktur JSON.

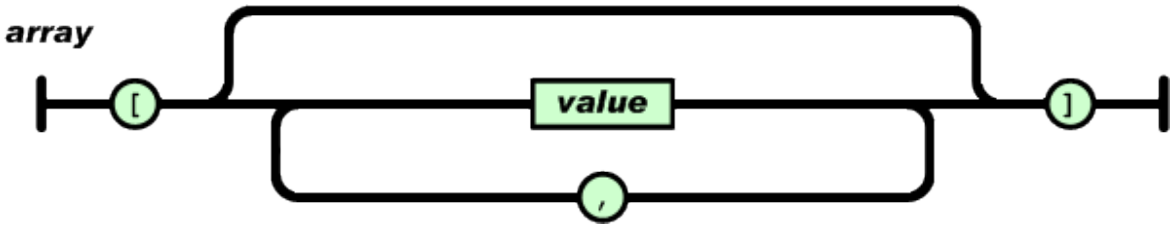
JSON menggunakan bentuk sebagai berikut:

- *Object* adalah sepasang nama atau nilai yang tidak berurutan. Objek 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).



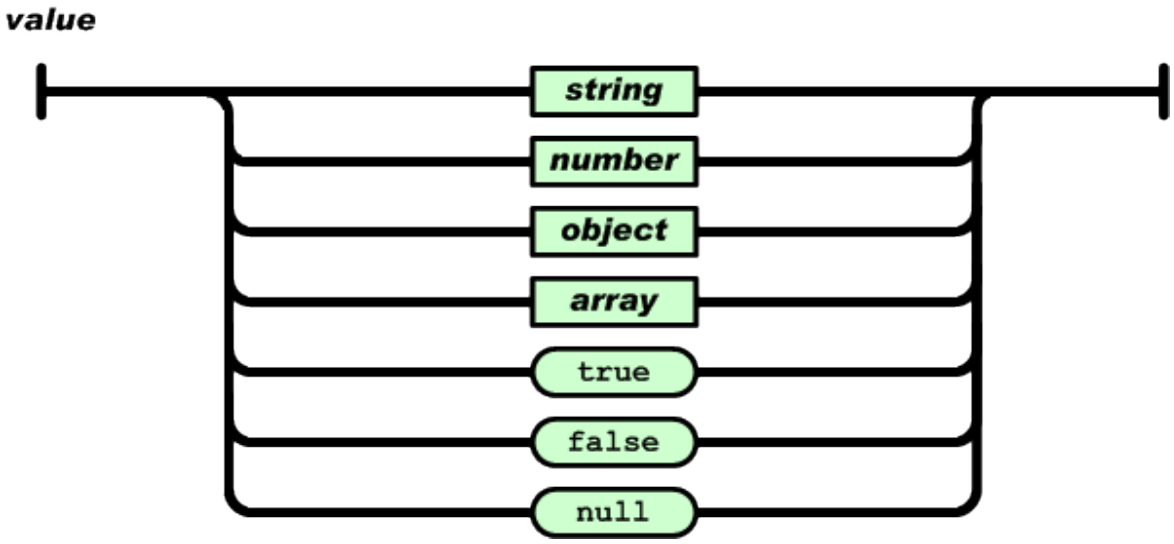
Gambar 2.2 Objek Dalam JSON

- *Array* adalah sekumpulan nilai yang teratur. *Array* dimulai dengan "[" (kurung kotak buka) dan diakhiri dengan "]" (kurung kotak tutup). Setiap nilai dipisah dengan "," (koma).



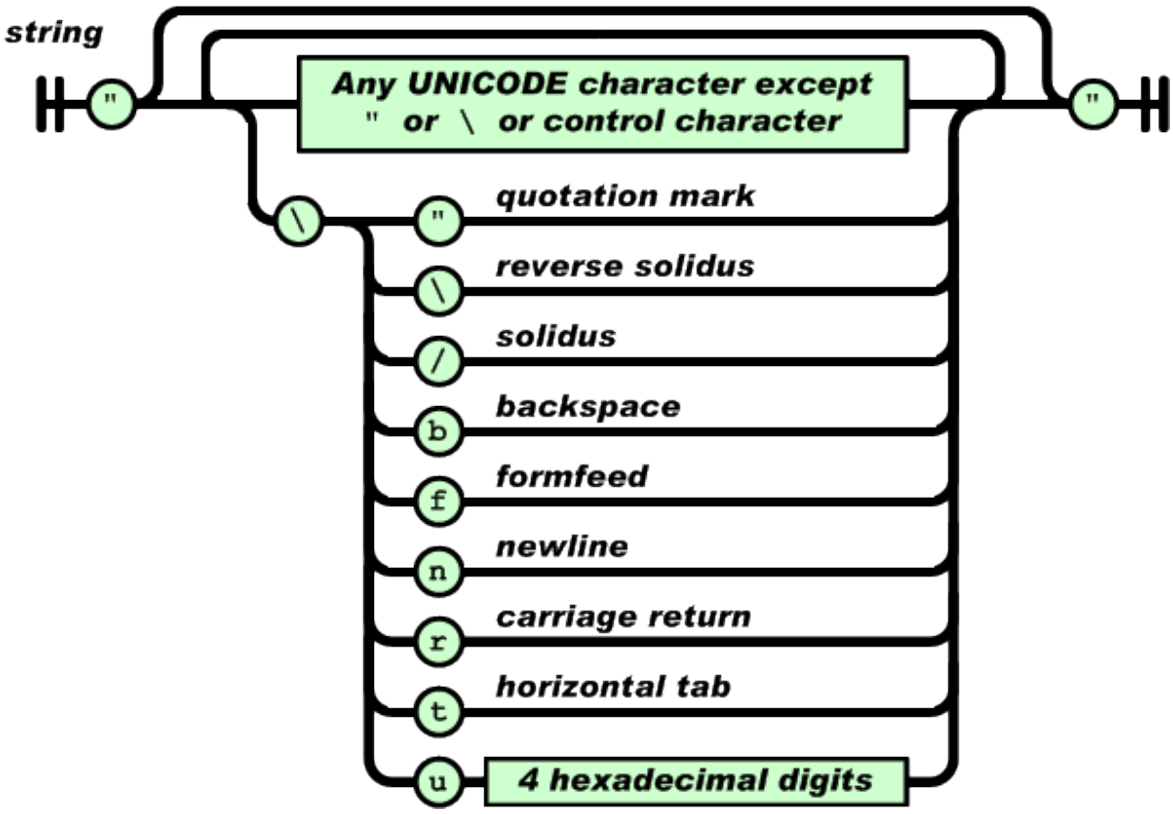
Gambar 2.3 *Array* Dalam JSON

- Nilai (*value*) bisa berupa *string* dalam tanda kutip ganda, atau angka, *TRUE* atau *FALSE* atau *NULL*, sebuah *object* atau sebuah *array*. Struktur ini dapat disusun bertingkat.



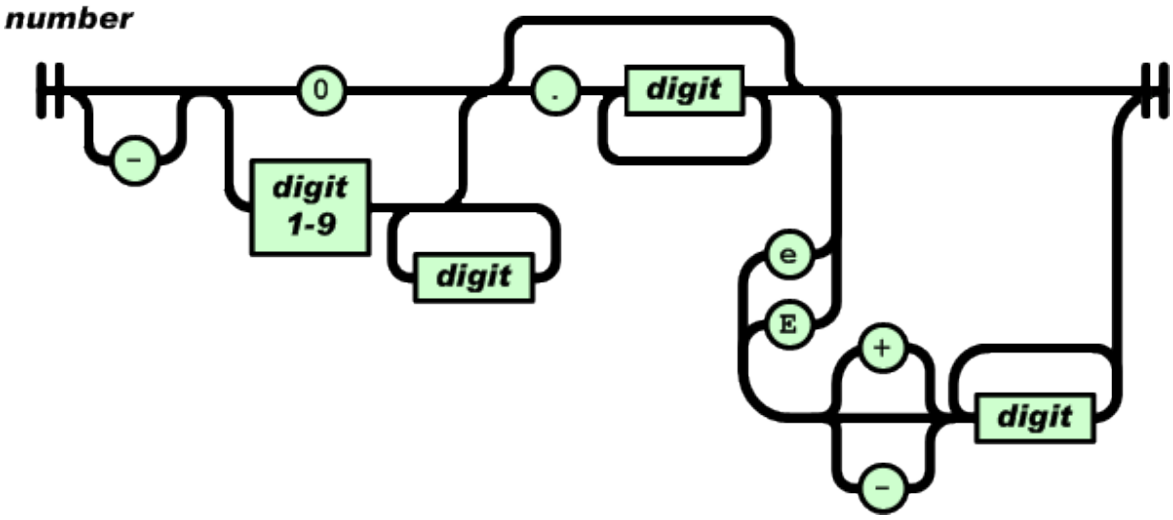
Gambar 2.4 *Value* atau Nilai Dalam Format JSON

- *String* adalah kumpulan dari nol atau lebih karakter *unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam *string* dapat digunakan *blackslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan string C atau *Java*.



Gambar 2.5 Tipe Data *String* Pada Format JSON

- *Number* (angka) sangat mirip dengan angka di bahasa C atau *Java*, kecuali tipe bilangan oktal dan heksadesimal tidak digunakan.



Gambar 2.6 *Number* Pada Format JSON

Spasi kosong (*Whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detail *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

2.2.8 *System Development Life Cycle* (SDLC)

SDLC adalah suatu kerangka yang menggambarkan beberapa kegiatan yang dilakukan melalui beberapa tahap dalam pembuatan sebuah *software* (Fatwa, 2007). Model SDLC yang digunakan dalam pengembangan aplikasi ini adalah model *Waterfall*. Disebut *waterfall* karena tahapan berjalan secara berurutan, jadi sebelum tahap satu selesai tidak bisa pindah ketahap selanjutnya. Model *waterfall* juga disebut dengan *clasic life cycle.* Model ini membutuhkan pendekatan sistematis dan sekuensial dalam pengembangan perangkat lunak, yang dilakukan secara terurut yang dimulai dari analisis, desain, pengkodean, penggujian dan pemeliharaan (*maintenance*).

2.2.9 *Unified Modeling Laguage* (UML)

UML adalah bahasa pemodelan yang digunakan untuk menspesifikasikan, mendokumentasikan dan membangun sistem perangkat lunak. UML yang digunakan dalam pengembangan aplikasi pencarian buku ini antara lain *Use Case Diagram* dan *Activity Diagram*.

a. Use Case Diagram

Use case diagram adalah suatu diagram yang menggambarkan suatu sistem dan bagaimana sistem tersebut bekerja. Perancangan *use case* digunakan untuk memodelkan proses berdasarkan perspektif *user*. *Use*

case diagram terdiri atas diagram untuk *use case* dan *actor*. *Use case diagram* menggambarkan orang yang berinterkasi dengan sistem. Simbol-simbol yang dalam *use case diagram* dapat dilihat pada tabel 2.1.

Tabel 2.1 Simbol-simbol Dalam *Use Case Diagram*

NO	Gambar	Nama Gambar	Keterangan
1		*Use Case*	Merupakan fungsionalitas yang disediakan sistem sebagai unit yang bertukar pesan dengan *actor*.
2		*Actor*	Merupakan *abstraction* dari orang yang mengaktifkan fungsi dari target sistem dan merupakan orang yang berinteraksi dengan *use case*.
3		*Association*	Digambarkan dengan garis tanpa panah yang mengindikasikan siapa berinteraksi secara langsung dengan sistem.
4		*Generalization*	Mengindikasikan siapa yang berinteraksi secara pasif dengan sistem.

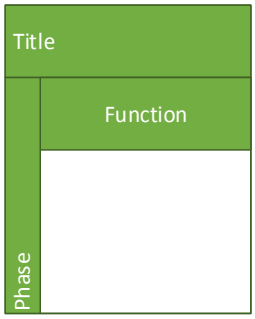
NO	Gambar	Nama Gambar	Keterangan
5		*Include*	Mengidentifikasi hubungan antar dua *use case* dimana satu *use case* memanggil *use case* lainnya.
6		*Extend*	Merupakan perluasan dari *use case* jika kondisi atau syarat terpenuhi.

b. Activity Diagram

Activity diagram merupakan diagram yang digunakan untuk menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel 2.2.

Tabel 2.2 Simbol-simbol Dalam A*ctivity Diagram*

No	Gambar	Nama Gambar	Keterangan
1		*Start Point*	Merupakan awal dalam aktivitas.
2		*End Point*	Merupakan akhir dalam aktivitas.
3		*Activities*	Menggambarkan suatu kegiatan bisnis atau proses.

No	Gambar	Nama Gambar	Keterangan
4		*Decision Point*	Menggambarkan pilihan untuk pengembalian keputusan dalam aktivitas.
5		*Swimlane*	Untuk pembagian *activity diagram* yang menunjukkan siapa yang melakukan aktivitas.

2.2.10 Metode Pengujian Sistem

Pengujian sistem menyajikan anomali yang menarik bagi perekayasa perangkat lunak pada proses perangkat lunak, perekayasa berusaha membangun perangkat lunak dari konsep abstrak ke implementasi yang dapat di lihat, kemudian dilakukan pengujian.

2.2.10.1 Pengujian *Black Box*

Pengujian *Black Box* terfokus pada apakah unit program memenuhi tebutuhan (*requirement*) yang disebutkan dalam spesifikasi. Pada *Black Box Testing* pengujian dilakukan hanya dengan menjalankan atau eksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses yang diinginkan. Jika ada unit yang tidak sesuai dengan *output*nya maka untuk penyelesainnya diteruskan pada metode *white box testing.*