Jurnal Strategi
Volume 1 Nomor 1 Mei 2019

APLIKASI PENCARIAN SPESIFIKASI MOTOR BERBASIS WEB SEMANTIK

Edward Kurniawan[#1], Bernard R. Suteja[*2]

[#]*Jurusan S1 Teknik Informatika, Universitas Kristen Maranatha*
Jl.Prof. Drg. Surya Sumantri No 65, Bandung
[1]elstein.ed@gmail.com

[*]*Jurusan S1 Teknik Informatika, Universitas Kristen Maranatha*
Jl.Prof. Drg. Surya Sumantri No 65, Bandung
[2]bernard.rs@it.maranatha.edu

Abstract — ***Finding something in digital media has been closely stick in our lives. Information that is spread on digital media is very much and very complicated that is used for things. For example, in searching for a motorbike, the data generated when searching may not match the keyword. One way to get the most that can be processed is to use the semantic web method. With the semantic web, large-scale data can be processed easily and more structured. Semantic web can provide a knowledge about something such as someone's biography. This knowledge can be used in various applications. With this knowlegde, it will be easier to group information, which allows a search to produce results that match a keyword. This Semantic web-based motorcycle specification search application will help users find various motors that have specifications, with ontology as a data source and SPARQL as data manager.***

Keywords— **knowledge, semantic web**

I. PENDAHULUAN

Banyak orang yang ingin membeli motor tetapi tidak mengetahui spesifikasi yang sesuai dengan yang diinginkan. Pada saat orang ingin mencari spesifikasi motor di media digital, informasi yang dihasilkan mungkin tidak sesuai dengan kata kunci yang diketikkan. Motor memiliki spesifikasi dan fitur yang berbeda-beda. Informasi yang tersebar di media digital sudah sangat banyak yang mengakibatkan permasalahan yang sangat rumit diantaranya dimana sebuah kata dapat memiliki banyak arti atau makna mengenai suatu hal. Pengguna akan kesulitan dalam memlakukan pencarian spesifikasi-spesifikasi motor. Karena informasi di media digital sudah terlalu banyak maka dibutuhkan sebuah alat pencarian yang dapat menyaring informasi-informasi tersebut.

Dengan menggunakan metode web semantik data dalam skala besar dapat diproses dengan mudah oleh mesin. Web semantik mengacu pada kemampuan mesin untuk dapat memahami bahasa manusia [1]. Web semantik dapat melakukan pencarian yang lebih terstruktur sehingga informasi yang dihasilkan untuk pengguna lebih spesifik sesuai dengan kata kunci yang diketikkan pengguna.

Dengan adanya permasalahan ini, akan dibuat sebuah aplikasi berbasis web semantik yang dapat memudahkan orang yang ingin mencari spesifikasi motor. Diharapkan aplikasi ini dapat mempermudah orang agar mendapatkan informasi yang lebih akurat.

II. KAJIAN TEORI

A. *Web Semantik*

Web Semantik menyediakan kerangka umum yang memungkinkan data dibagi dan digunakan kembali di seluruh batasan aplikasi, perusahaan, dan masyarakat. Ini adalah upaya kolaborasi yang dipimpin oleh *W3C* dengan partisipasi dari sejumlah besar peneliti dan mitra industri. Hal ini didasarkan pada *Resource Description Framework (RDF*) *[2]*

Web semantik merupakan suatu aplikasi web yang mempunyai *knowledge base* tertentu sehingga bisa dikatakan web semantik memiliki sifat yang lebih pintar dari web biasa.

B. *OWL*

Web Ontology Language (*OWL*) adalah bahasa Semantik yang dirancang untuk mewakili pengetahuan yang kaya dan kompleks tentang berbagai hal, kelompok, dan saling hubungan. *OWL* adalah bahasa berbasis logika komputasi sehingga

pengetahuan yang diungkapkan dalam *OWL* dapat dimanfaatkan oleh program komputer, misalnya, untuk memverifikasi konsistensi informasi atau untuk membuat informasi yang ada didalamnya tidak berbelit-belit. Dokumen *OWL*, yang dikenal sebagai ontologi, dapat dipublikasikan di *World Wide Web* dan dapat merujuk atau dirujuk dari ontologi *OWL* lainnya. *OWL* adalah bagian dari tumpukan teknologi Semantic *Web W3C*, yang mencakup *RDF, RDFS,* dan *SPARQL* [3].

C. SPARQL

SPARQL adalah *semantic web relational database*. Jika web semantic adalah sebuah koleksi dari basis data global, maka *SPARQL* dapat membuat sebuah koleksi tersebut menjadi sebuah basis data yang sangat besar. *SPARQL* dibangun berdasarkan standar lain termasuk *RDF, XML, HTTP,* dan *WSDL. SPARQL* memungkinkan pengguna untuk menulis query terhadap data yang dapat disebut "*key-value*" secara bebas atau, lebih spesifik lagi, data mengikuti spesifikasi *RDF* dari *W3C*. Seluruh *database* dengan demikian merupakan satu set "subjek-predikat-objek" [4].

D. RDF

RDF adalah model standar untuk pertukaran data di *Web. RDF* memiliki fitur yang memudahkan penggabungan data bahkan jika skema dasarnya berbeda, dan secara khusus mendukung perubahan skema dari waktu ke waktu tanpa memerlukan semua data untuk diubah.

RDF memperluas struktur *link Web* untuk menggunakan *URI* untuk memberi nama hubungan antara hal-hal dan juga dua ujung *link* (ini biasanya disebut sebagai *"triple"*). Dengan menggunakan model sederhana ini, memungkinkan data terstruktur dan semi-terstruktur untuk dicampur, terbuka, dan dibagi ke berbagai aplikasi.

Struktur ini membentuk grafik berlabel, dan mewakili link yang dinamai di antara dua sumber, yang ditunjukkan oleh *graph nodes.* Tampilan grafik ini adalah model termudah untuk *RDF* dan sering digunakan dalam penjelasan visual yang mudah dipahami *[5]*.

E. Apache Jena

Jena adalah *framework Java* untuk membangun aplikasi *Semantic Web.* Jena menyediakan *library Java* yang luas untuk membantu *developers* mengembangkan kode yang menangani *RDF, RDFS, RDFa, OWL* dan *SPARQL* sesuai dengan rekomendasi *W3C* yang dipublikasikan.

Jena menyertakan *rule-based inference engine* untuk melakukan penalaran berdasarkan ontologi *OWL dan RDFS,* dan berbagai strategi penyimpanan untuk menyimpan *RDF triples* dalam memori atau *disk [6]*. Cara untuk mengakses apache jena dengan mengetikan localhost:8080 pada browser.

F. FUSEKI

Fuseki adalah server *SPARQL* yang dapat berjalan sebagai layanan sistem operasi, seperti aplikasi web Java (file WAR), dan sebagai server mandiri. FUSEKI menyediakan keamanan (menggunakan Apache Shiro) dan memiliki antarmuka pengguna untuk pemantauan dan administrasi server. FUSEKI menyediakan protokol *SPARQL* 1.1 untuk *query dan update* serta protokol *SPARQL Graph Store.*

Fuseki terintegrasi dengan *TDB* untuk menyediakan lapisan penyimpanan persisten yang kuat dan transaksional, dan menggabungkan Jena *text query* dan *Jena spatial query.* Ini dapat digunakan untuk menyediakan mesin protokol untuk *query* dan sistem penyimpanan *RDF* lainnya *[7]*. Cara untuk mengakses FUSEKI dengan mengetikan localhost:3030 pada browser.

G. HTML

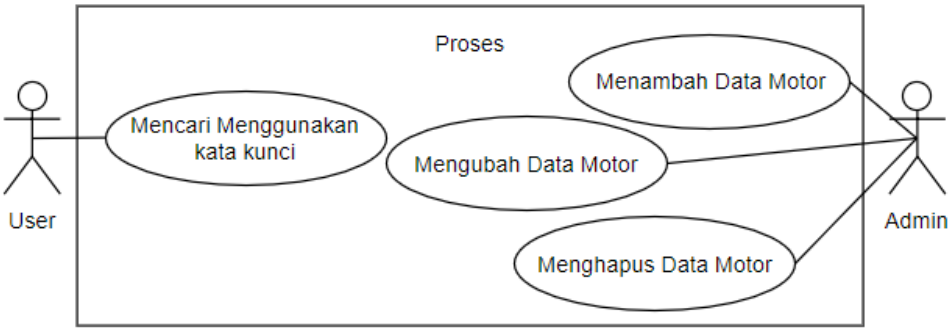
HTML (*Hyper Text Markup Language*) adalah bahasa markup standar untuk membuat halaman Web. HTML menggambarkan struktur halaman Web menggunakan markup. Elemen HTML adalah blok bangunan dari halaman HTML, Elemen HTML diwakili oleh tag. Tag HTML memberikan label konten seperti "judul", "paragraf", "tabel", dan seterusnya. Browser tidak menampilkan tag HTML, tapi menggunakannya untuk menampilkan konten halaman [8].

H. PHP

PHP (*Hypertext Preprocessor*) adalah bahasa *open source scripting* yang banyak digunakan dengan tujuan umum yang sangat cocok untuk pengembangan web dan dapat ditanamkan ke dalam HTML [9].

III. ANALISIS DAN PERANCANGAN SISTEM

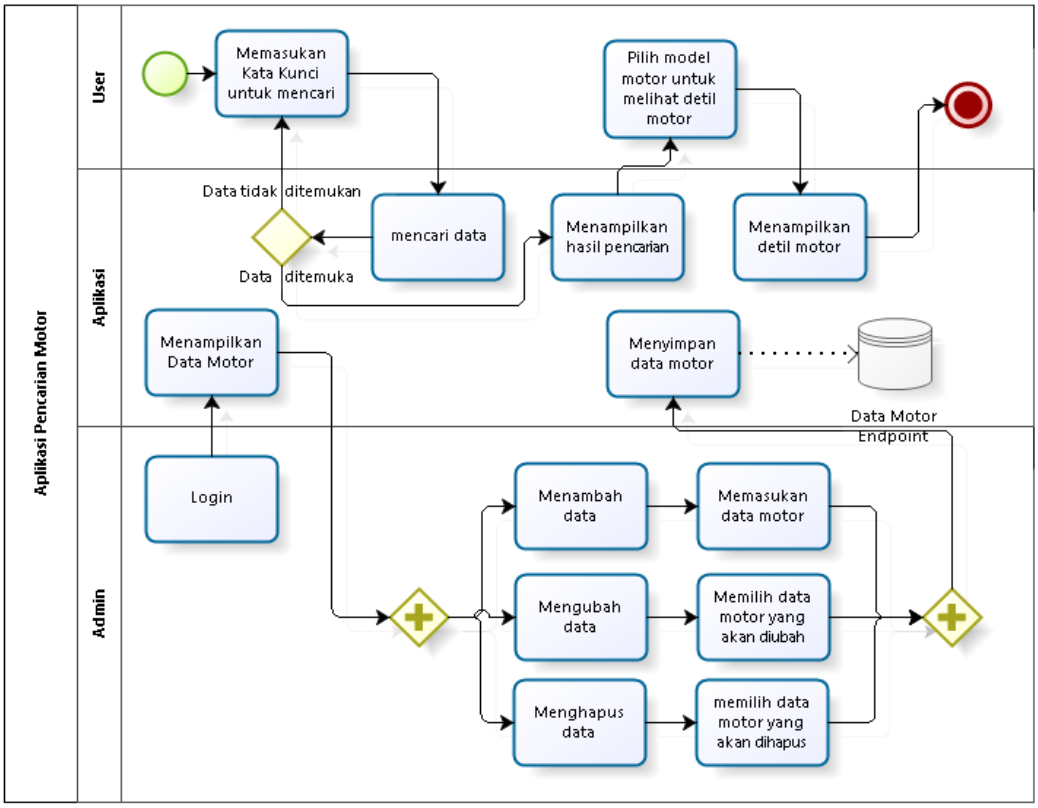
A. Use Case Diagram



Gambar 1 Use Case Diagram

Pada Gambar 1 pengguna hanya dapat melakukan pencarian dengan memasukan kata kunci untuk mencari spesifikasi motor, sedangkan Admin dapat melakukan proses tambah data, hapus data, atau mengubah data.

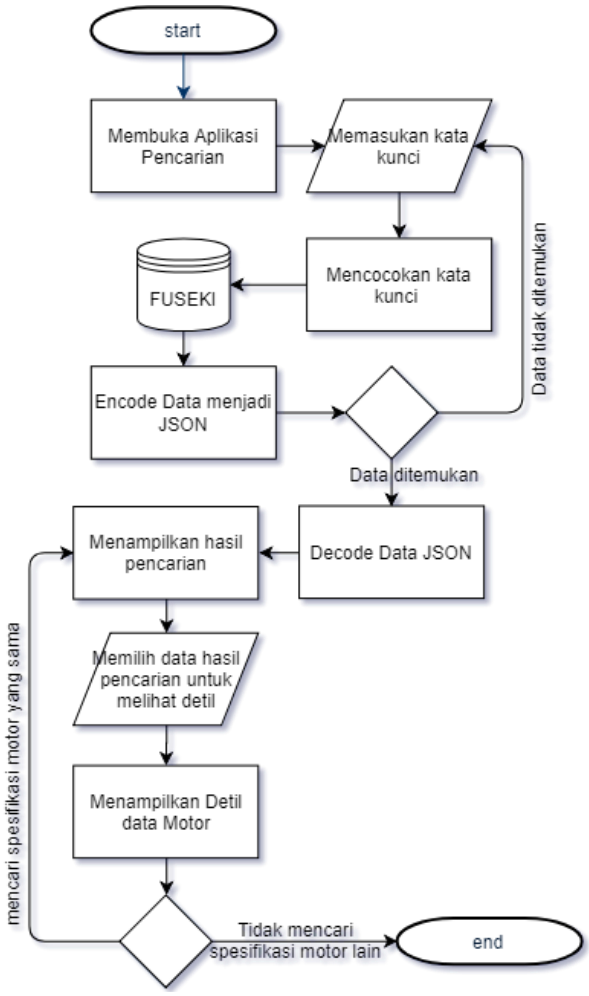
B. BPMN



Gambar 2 BPMN

Pada Gambar 2 proses yang terjadi didalam rancangan ini user memasukan kata kunci untuk mencari spesifikasi motor, lalu aplikasi akan mencari data sesuai dengan kata kunci yang dimasukan. Admin diharuskan untuk login dengan menggunakan username dan password agar dapat menambah data motor, mengubah data motor, dan menghapus data motor.

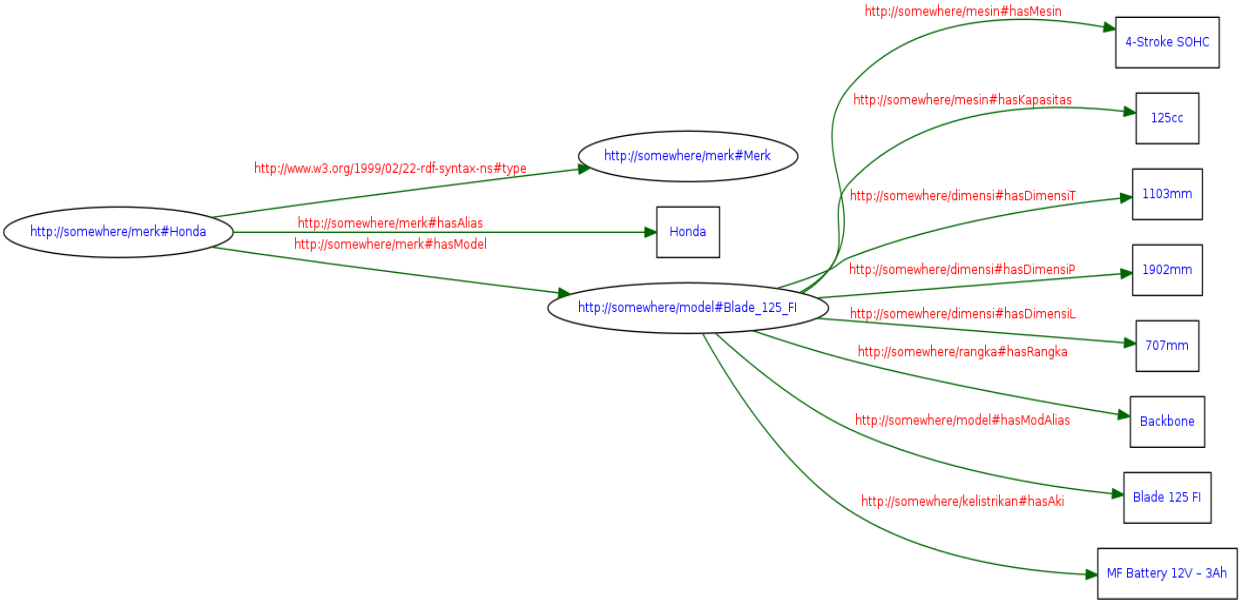
C. Flowchart



Gambar 3 Flowchart

Pada Gambar 3 proses dimulai saat pengguna membuka aplikasi pencarian, lalu memasukan kata kunci untuk mencari data motor. Aplikasi akan mencocokan kata kunci yang diketikan pengguna menggunakan SPARQL query yang akan dicocokan pada data yang tersimpan pada FUSEKI. Data yang akan ditampilkan akan di *encode* kedalam JSON, lalu akan di *decode* oleh PHP untuk ditampilkan.

D. RDF-Graph



Gambar 4 RDF-Graph

Pada Gambar 4 merupakan ontologi yang digunakan untuk menyimpan data-data motor seperti merk, mesin, model, tipe rangka, dimensi, kelistrikan motor.

IV. IMPLEMENTASI

A. Tampilan Halaman Utama

Pada Gambar 5 merupakan tampilan utama saat membuka aplikasi pencarian. Dihalaman ini pengguna dapat menggunkan kata kunci untuk mencari spesifikasi motor.

Gambar 5 Halaman Utama

Misalnya pengguna mengetikan “Blade 125” maka aplikasi akan menampilkan data motor model Blade dan mempunyai mesin 125cc. Contoh kode untuk mengambil data sesuai kata kunci dapat dilihat pada Kode 1

B. Tampilan Hasil Pencarian

Pada Gambar 6 merupakan tampilan hasil pencarian setelah pengguna menggunakan kata kunci untuk mencari data motor.

Merk	Model Motor
Honda	Blade 125 FI
Honda	Supra GTR 150

Gambar 6 Hasil Pencarian

Pengguna dapat melihat detil spesifikasi motor dengan menekan hasil pencarian model motor. Contoh kode program untuk menampilkan data sesuai kata kunci dapat dilihat pada Kode 1.

```
String search = request.getParameter("search");
String sparqlendpoint = "http://localhost:3030/Motor/query";
String perintahSPARQL =" "+
    "PREFIX mod:<http://somewhere/model#> "+
    "PREFIX mtm:<http://somewhere/mesin#> "+
    "PREFIX mtr:<http://somewhere/merk#> "+
    "SELECT DISTINCT "+
    "   ?Umerk ?Umodel  "+
    "   ?merk ?model ?mesin "+
    "WHERE {"+
    "   ?Umerk mtr:hasModel ?Umodel. "+
    "   ?Umerk mtr:hasAlias ?merk.  "+
    "   ?Umodel mod:hasModAlias ?model. "+
    "   ?Umodel mtm:hasMesin ?mesin.    "+
    "FILTER("+
    "   regex(?merk, \""+search+"\", 'i')"+
    "   || regex(?model, \""+search+"\", 'i')"+
    ")"+
    "}";


Query query = QueryFactory.create(perintahSPARQL);
QueryExecution qe = QueryExecutionFactory.sparqlService(sparqlendpoint,query);
ResultSet hasil = qe.execSelect();

ByteArrayOutputStream b = new ByteArrayOutputStream();
ResultSetFormatter.outputAsJSON(b,hasil);
String json = b.toString();
out.println(json);
```

Kode 1 Query Search

Contoh String search = request.getParameter("search"); digunakan untuk mengambil kata kunci pada *text box "search"* pada Gambar 5 yang akan dicocokan pada *query FILTER.*

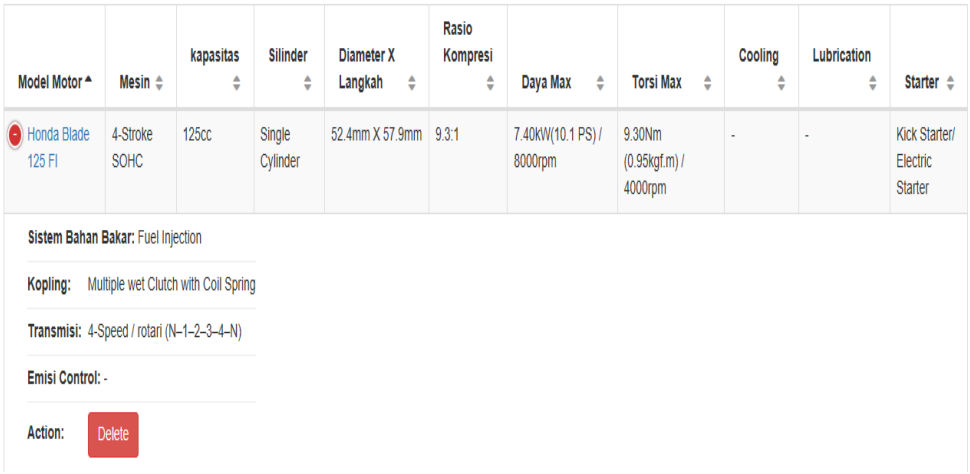
C. TAMPILAN DETIL SPESIFIKASI MOTOR

Gambar 7 merupakan potongan tampilan dari halaman detil motor. Pengguna dapat melihat data motor yang memiliki spesifikasi yang sama. Misalnya pengguna menekan data pada spesifikasi mesin "125cc" maka aplikasi akan menampilkan hasil data spesifikasi motor yang memiliki mesin "125cc".

Gambar 7 Detil Spesifikasi Mesin

D. Halaman Utama Admin

Gambar 8 merupakan tampilan halaman utama untuk admin setelah melakukan login. Admin dapat menambahkan data motor, mengubah data motor dan menghapus data motor.



Gambar 8 Halaman Utama Admin

Admin dapat mengubah data motor dengan menekan data model motor yang akan diubah, dan menghapus data motor dengan menekan tombol “Delete”.

E. Tampilan Tambah Data

Gambar 9 merupakan potongan tampilan halaman tambah data. Admin diharuskan mengisi semua *field* untuk menambahkan data motor baru.

Gambar 9 Tambah Data Motor

Gambar 9 merupakan bagian form merk dan model motor. Namun form ini masih belum dapat digunakan karena adanya beberapa kendala saat memanggil fungsi query SPARQL dan memasukan data ke dalam FUSEKI, tetapi query untuk menambahkan data dapat dijalankan secara manual dengan cara mengetikan localhost:8080/namaFolder/namaFile.jsp pada browser. Contoh kode program untuk menambah data motor baru dapat dilihat pada Kode 2.

```
<%
String merk = request.getParameter("merk");
String model = request.getParameter("model");

String sparqlinsertendpoint = "http://localhost:3030/coba/update";
String perintahSPARQL =" "+
    "PREFIX mod:<http://somewhere/model#> "+

    "PREFIX mtr:<http://somewhere/merk#> "+

    "INSERT DATA{"+
    "   mtr:"+merk+" a mtr:Merk.    "+
    "   mtr:"+merk+"    mtr:hasAlias \""+merk+"\";  "+
    "   mtr:hasModel mod:"+model+". "+
    "   mod:"+model+" mod:hasModAlias \""+model+"\".    "+

UpdateRequest query = UpdateFactory.create(perintahSPARQL);
UpdateProcessor reg = UpdateExecutionFactory.createRemote(query,sparqlinsertendpoint);
reg.execute();
%>
```

Kode 2 Query Tambah Data

Contoh String merk = request.getParameter("merk"); digunakan untuk mengambil data dari *text box* "merk" pada Gambar 9 yang akan disimpan kedalam sebuah URI `<http://somewhere/merk#>`.

F. Tampilan Ubah Data

Gambar 10 merupakan potongan halaman ubah data motor setelah admin memilih dan menekan model motor. Admin dapat mengubah data spesifikasi motor yang akan diubah saja tidak diharuskan untuk mengubah seluruhnya.

Gambar 10 Ubah Data Motor

Gambar 10 merupakan bagian form merk dan model motor. Namun form ini masih belum dapat digunakan karena adanya beberapa kendala saat memanggil fungsi query SPARQL dan memasukan data ke dalam FUSEKI, tetapi query untuk mengubah data dapat dijalankan secara manual dengan cara mengetikan localhost:8080/namaFolder/namaFile.jsp pada browser. Contoh kode program untuk menambah data motor baru dapat dilihat pada Kode 3.

```
<%
String merk = request.getParameter("merk");
String model = request.getParameter("model");
String mesin = request.getParameter("mesin");

String sparqldeleteendpoint = "http://localhost:3030/coba/update";
String perintahSPARQL =" "+
    "PREFIX mod:<http://somewhere/model#> "+
    "PREFIX mtr:<http://somewhere/merk#> "+

    "DELETE data{"+
    "   mtr:"+merk+" a mtr:Merk.    "+
    "   mtr:"+merk+"    mtr:hasAlias \""+merk+"\";  "+
    "   mtr:hasModel mod:"+model+". "+
    "};"+

    "INSERT DATA{"+
    "   mtr:"+merk+" a mtr:Merk.    "+
    "   mtr:"+merk+"    mtr:hasAlias \""+merk+"\";  "+
    "   mtr:hasModel mod:"+model+". "+
    "   mod:"+model+" mod:hasModAlias \""+model+"\".    "+
    "}";

UpdateRequest query = UpdateFactory.create(perintahSPARQL);
UpdateProcessor reg = UpdateExecutionFactory.createRemote(query,sparqldeleteendpoint);
reg.execute();
%>
```

Kode 3 Query Update Data

Contoh String merk = request.getParameter("merk"); digunakan untuk mengambil data dari *text box* "merk" pada Gambar 9 yang kemudian URI `<http://somewhere/merk#>` akan dihapus, kemudian URI: `<http://somewhere/merk#>` akan disimpan kembali kedalam FUSEKI.

G. Hapus Data

Menghapus data motor dapat dilakukan dengan menekan tombol "Delete" yang ada pada tabel data, sama seperti yang ada pada Gambar 8. Namun tombol "Delete" belum dapat digunakan karena adanya beberapa kendala saat memanggil query SPARQL, tetapi query untuk meghapus data dapat dijalankan secara manual dengan cara mengetikan localhost:8080/namaFolder/namaFile.jsp pada browser. Contoh kode program untuk menambah data motor baru dapat dilihat pada

Kode 4

```
<%
String merk = request.getParameter("merk");
String model = request.getParameter("model");
String sparqldeleteendpoint = "http://localhost:3030/coba/update";
String perintahSPARQL =" "+
    "PREFIX mod:<http://somewhere/model#> "+
    "PREFIX mtr:<http://somewhere/merk#> "+
    "DELETE data{"+
    "   mtr:"+merk+" a mtr:Merk.    "+
    "   mtr:"+merk+"    mtr:hasAlias \""+merk+"\";  "+
    "   mtr:hasModel mod:"+model+". "+
    "   mod:"+model+" mod:hasModAlias \""+model+"\".    "+

UpdateRequest query = UpdateFactory.create(perintahSPARQL);
UpdateProcessor reg = UpdateExecutionFactory.createRemote(query,sparqldeleteendpoint);
reg.execute();
%>
```

Kode 4 Query Delete Data

Contoh String merk = request.getParameter("merk"); digunakan untuk mengambil data dari tabel data “merk” pada Gambar 8 yang kemudian URI `<http://somewhere/merk#>` `akan dihapus.`

V. KESIMPULAN

Dengan membuat aplikasi pencarian berbasis web semantik diharapkan data yang telah disediakan dapat digunakan untuk mempermudah aplikasi lain dalam memperoleh data. *Knowledge* mengenai spefikasi motor, data-data motor dapat dengan mudah dikelompokan sesuai dengan merk dan modelnya.

DAFTAR PUSTAKA

[1] B. R. Suteja and H. Toba, Web cerdas itu web semantik, Garudhawaca, 2017.

[2] I. Herman, "W3C Semantic Web," W3C, [Online]. Available: https://www.w3.org/2001/sw/.

[3] O. W. Group, "W3C Semantic Web," W3C, [Online]. Available: https://www.w3.org/2001/sw/wiki/OWL.

[4] S. W. Group, "W3C Semantic Web," W3C, [Online]. Available: https://www.w3.org/2001/sw/wiki/SPARQL.

[5] R. W. Group, "W3C Semantic Web," W3C, [Online]. Available: https://www.w3.org/RDF/.

[6] A. Jena, "What is Jena?," The Apache Software Foundation, [Online]. Available: https://jena.apache.org/about_jena/about.html. [Accessed 2 february 2018].

[7] A. Jena, "Apache Jena FUSEKI," The Apache Software Foundation, [Online]. Available: https://jena.apache.org/documentation/fuseki2/index.html. [Accessed 2 February 2018].

[8] w3schools.com, "w3schools.com," [Online]. Available: https://www.w3schools.com/html/html_intro.asp.

[9] P. Group, "PHP," [Online]. Available: http://php.net/manual/en/intro-whatis.php.