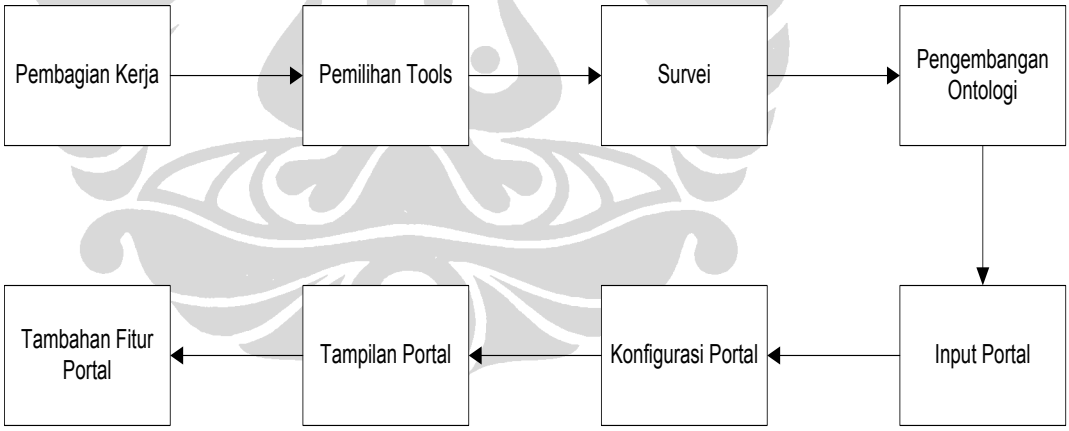
BAB 3

METODOLOGI PENELITIAN

Bab ini menjelaskan tentang metodologi penelitian yang digunakan selama pengembangan proyek mahasiswa. Hal-hal yang dibahas dalam bab ini mencakup pembagian kerja kelompok, survei yang dilakukan ke Pemerintah Kota Depok, kerangka pengembangan proyek, pemilihan *tools*, pengembangan ontologi, *input* portal, konfigurasi portal, tampilan portal, dan struktur direktori yang digunakan dalam portal.

3.1 Kerangka Pengembangan

Untuk membantu pengembangan portal, diperlukan suatu kerangka pengembangan yang menjadi panduan dalam pengembangan portal. Gambar 3.1 merupakan kerangka pengembangan yang digunakan. Secara umum, kerangka pengembangan yang digunakan adalah sebagai berikut.



Gambar 3.1 Kerangka Pengembangan

- **Pembagian kerja**

  Terdiri dari pembagian kerja masing-masing anggota kelompok dalam mengerjakan proyek ini. Pembagian kerja dilakukan mulai dari awal pengembangan proyek sampai dengan penulisan laporan proyek.

- **Pemilihan *tools***

  Terdiri dari pemilihan *tools* Protégé 3.3.1 untuk pengembangan ontologi, RDF123 sebagai *converter* data dari *spreadsheet* ke dalam bentuk RDF, dan portalCore untuk pengembangan portal yang dikerjakan.

- **Survei**

  Wawancara dengan Kepala Subbagian Humas. Di samping itu, dilakukan juga pengambilan data bagian Humas Protokol Pemerintah Kota Depok yang menjadi studi kasus dari pengerjaan proyek ini.

- **Pengembangan ontologi**

  Pengembangan ontologi yang dilakukan dengan menggunakan *tool* Protégé 3.3.1. Tahap-tahap pengembangan ontologi sesuai dengan yang dijelaskan pada subbab 2.2.3 mengenai pengembangan ontologi. Pengembangan ontologi dilakukan dari awal dan tidak me-*reuse* ontologi yang sudah ada.

- ***Input* portal**

  Tahapan ini terdiri dari pendefinisian data dan *rules* yang akan menjadi *input* dari portal yang dikembangkan.

- **Konfigurasi portal**

  Tahapan ini terdiri dari pendefinisian *datasources*, *facets*, dan *templates*.

- **Tampilan portal**

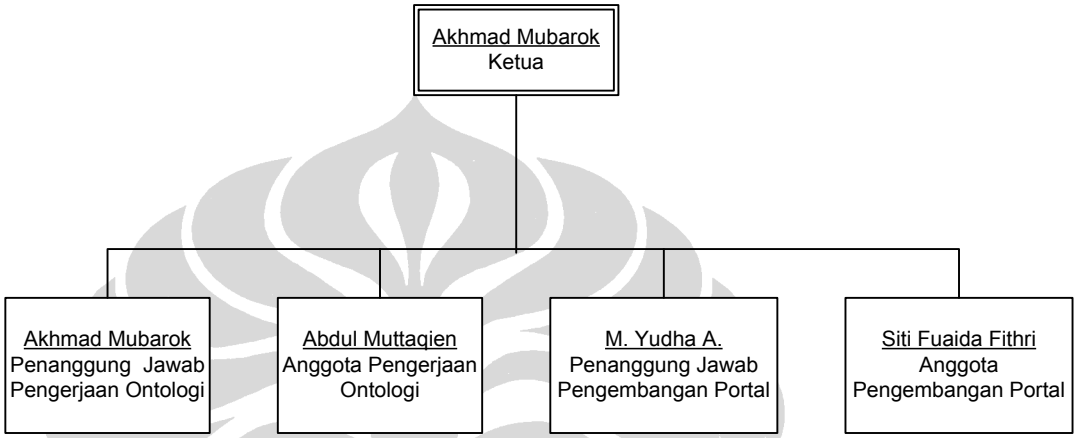
  Tahapan ini terdiri dari pembuatan *templates* atau visualisasi data pada portal.

- **Tambahan fitur portal**

  Tahapan ini terdiri dari penambahan fitur *add*, *update*, dan *delete* data pada portal.

3.2 Pembagian Kerja

Proyek mahasiswa ini dikerjakan oleh empat orang. Untuk memudahkan pengerjaan proyek, dibentuklah struktur organisasi kelompok. Struktur organisasi kelompok tersebut dapat dilihat pada Gambar 3.2. Berikut ini adalah penjelasan mengenai pembagian kerja yang dilakukan dalam kelompok.



Gambar 3.2 Struktur Organisasi Kelompok

Akhmad Mubarok (Ketua Kelompok)

Berikut ini adalah daftar tugas yang dikerjakan oleh Akhmad Mubarok.

- Melakukan survei ke Pemerintah Kota Depok.
- Menyiapkan daftar pertanyaan wawancara.
- Melakukan *coding* ontologi *e-government* dengan menggunakan Protégé 3.3.1.
- Menjelaskan dan mendiskusikan ontologi yang dihasilkan kepada anggota kelompok lainnya.
- Membuat *inference rule*.
- Mengimplementasikan, menyesuaikan, dan mengujicobakan ontologi beserta data-data hasil survei yang telah diolah ke dalam *semantic portal* yang dikerjakan.

- Membuat laporan proyek mahasiswa subbab 2.1, 2.2, 3.1, 3.2, 3.3, 3.4, 3.5, 4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.1.5, 4.2, 4.3, dan mengintegrasi seluruh bagian laporan.

Abdul Muttaqien

Berikut ini adalah daftar tugas yang dikerjakan oleh Abdul Muttaqien.

- Melakukan survei ke Pemerintah Kota Depok.
- Mengolah data hasil survei.
- Melakukan *coding* ontologi *e-government* dengan menggunakan Protégé 3.3.1.
- Membuat *inference rule*.
- Merancang tampilan *semantic portal* yang dikerjakan.
- Membuat laporan proyek mahasiswa bab 1, subbab 3.6, 3.7, 3.8, 3.10, dan bab 5.

M. Yudha A.

Berikut ini adalah daftar tugas yang dikerjakan oleh M. Yudha A..

- Melakukan survei ke Pemerintah Kota Depok.
- Mempelajari portalCore yang digunakan sebagai template *semantic portal* yang dikerjakan.
- Mengajarkan cara penggunaan portalCore ke anggota yang lain.
- Memodifikasi portalCore dengan menambahkan fitur *add, update*, dan *delete*.
- Membuat laporan proyek mahasiswa subbab 2.3, 3.9, 4.1.6, 4.1.7, dan 4.1.8.

Siti Fuaida Fithri

Berikut ini adalah daftar tugas yang dikerjakan oleh Siti Fuaida Fithri.

- Melakukan survei ke Pemerintah Kota Depok.
- Mempelajari portalCore yang digunakan sebagai template *semantic portal* yang dikerjakan.

- Mengajarkan cara penggunaan portalCore ke anggota yang lain.
- Memodifikasi portalCore dengan menambahkan fitur *add, update*, dan *delete*.
- Membuat laporan proyek mahasiswa subbab 2.4, 3.9, 4.1.6, 4.1.7, dan 4.1.8.

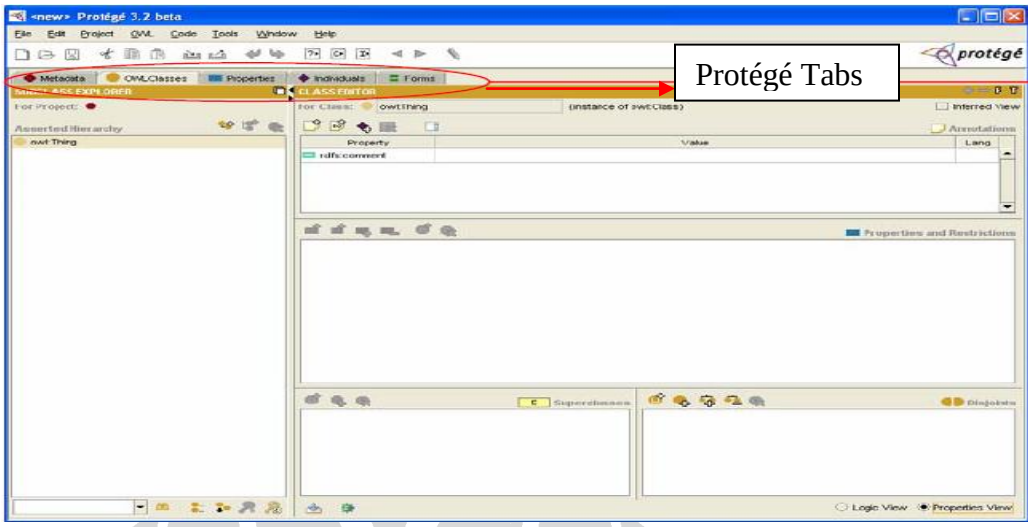
3.3 Pemilihan *Tools*

Sebelum melakukan pengembangan portal, dilakukan pemilihan *tools* yang akan digunakan. Terdapat tiga *tools* utama yang dipilih untuk membantu pengerjaan proyek ini, yaitu Protégé 3.3.1, RDF123, dan portalCore. Di samping itu, untuk menjalankan portal yang dikerjakan, digunakan *web server Apache Tomcat* 5.5. Berikut ini akan dijelaskan mengenai ketiga *tools* utama yang digunakan.

3.3.1 Protégé 3.3.1

Sebelum memulai proyek, pembimbing terlebih dahulu menyarankan untuk menggunakan *tool* ini untuk mengembangkan ontologi. Selain *powerful*, *tool* ini telah banyak digunakan dalam proyek penelitian ataupun aplikasi yang menerapkan ontologi (Ojo & Janowski, 2005). Pelaksana proyek kemudian mencoba untuk membuat ontologi sederhana dengan *tool* ini. Setelah mencoba *tool* tersebut, ternyata penggunaannya cukup mudah dan memiliki banyak fitur yang dapat membantu pengembangan proyek. Oleh karena itu, *tool* ini dipilih untuk membantu pengembangan ontologi.

Protégé merupakan sebuah *tool* yang digunakan untuk membuat domain ontologi, menyesuaikan *form* untuk *entry* data, dan memasukkan data (Protégé, 2009). Format penyimpanan Protégé dapat dalam bentuk OWL, RDF, XML, dan HTML. Protégé menyediakan kemudahan *plug and play* yang membuatnya fleksibel untuk pengembangan prototipe. Protégé dibuat dengan menggunakan bahasa pemrograman Java. Semua menu dalam Protégé dapat digunakan melalui *Graphical User Interface* (GUI) dengan menyediakan *tab* untuk masing-masing bagian dan fungsi standar. Contoh tampilan dari Protégé dapat dilihat pada Gambar 3.3 (Ontology, n.d.).



Gambar 3.3 Contoh Tampilan Protege

Protege merupakan *free software* dan membutuhkan SDK Java untuk menjalankannya. Protégé dapat berjalan di berbagai *platform* sistem operasi, antara lain Windows, Mac OS, Solaris, Linux, HP-UX, UNIX, dan AIX. Protégé dapat membuka berbagai macam format berkas, terdapat tiga format berkas umum yang dapat dibuka dengan Protégé, yaitu XML, RDF, dan OWL. Untuk dapat membuka berkas tersebut, maka perlu dilakukan pembuatan *project* baru pada Protégé, *project* tersebut memiliki format berkas .pprj (Ontology, n.d.).

Pada umumnya, ontologi yang dikembangkan dalam proyek ini menggunakan beberapa menu yang disediakan Protégé. Sebelumnya telah dijelaskan bahwa menu-menu tersebut dapat digunakan melalui *tab* yang terdapat dalam GUI Protégé. Berikut penjelasan dari masing-masing tab tersebut.

Tab OWL *Classes*

Tab ini berfungsi untuk mendefinisikan kelas dan hirarki kelas. Pada saat dijalankan, sudah terdapat sebuah kelas, yaitu owl:Thing yang merepresentasikan semua individu yang terdapat dalam ontologi yang dibuat. Hal ini disebabkan semua kelas yang dibuat akan menjadi subkelas dari owl:Thing. Terdapat dua bagian dalam *tab* ini, yaitu *subclass explorer* dan *class editor*.

a. *Subclass Explorer*

Subclass explorer akan menampilkan kelas-kelas dan hirarki kelas yang dibuat. Di samping itu, terdapat menu-menu yang digunakan untuk menambah kelas ataupun menghapus kelas. Menu-menu tersebut antara lain:

- *Create subclass*, digunakan untuk membuat subkelas.
- *Create sibling class*, digunakan untuk membuat *sibling class*.
- *Delete class*, digunakan untuk menghapus kelas.

b. *Class Editor*

Class editor digunakan untuk mendefinisikan kelas, properti dari kelas, dan juga relasi dengan kelas lain. Menu-menu yang terdapat pada *class editor* cukup banyak. Beberapa menu yang digunakan antara lain:

- *Create expression*, digunakan untuk membuat relasi antar kelas.
- *Create restriction*, digunakan untuk menentukan batasan keterlibatan kelas dalam suatu relasi dengan kelas lain, seperti ∀ (*AllValuesFrom*) dan ∃ (*SomeValuesFrom*).
- *Create disjoint class*, digunakan untuk kelas yang tidak beririsan dengan kelas lain.
- *Create annotation*, digunakan untuk menambahkan *metadata* (data tentang data).

Tab Properties

Tab ini berfungsi untuk merepresentasikan relasi antara dua *instance*. Terdapat dua tipe utama dari *tab* ini, yaitu *object properties* dan *datatype properties*. *Object properties* menghubungkan antara dua *instance*, sedangkan *datatype properties* menghubungkan satu *instance* dengan *XML Schema Datatype value* atau *rdf literal* (atribut yang dimiliki *instance* tersebut). Terdapat dua bagian dalam *tab* ini, yaitu *property browser* dan *property editor*.

a. *Property Browser*

Pada *property browser* terdapat empat *tab*, yaitu *tab object*, *datatype*, *annotation*, dan *all*. *Tab object* digunakan untuk membuat *object properties*, sedangkan *tab datatype* digunakan untuk membuat *datatype properties*. *Tab annotation* digunakan untuk menambahkan *metadata*, sedangkan *tab all* akan menampilkan seluruh properti yang dibuat.

b. *Property Editor*

Property editor digunakan untuk mendefinisikan *domain* dan *range* dari suatu properti. Selain itu, dapat juga ditambahkan *inverse property* dalam *object property*. Pendefinisian *domain* dan *range* diperlukan untuk menentukan di mana suatu properti terlibat.

Tab Metadata

Tab ini digunakan untuk mendefinisikan *namespace* dari ontologi yang dikerjakan. Untuk membuat *namespace* baru, cukup mengetik *namespace* tersebut pada *box edit* yang terdapat pada bagian atas *individual editor*. *Namespace* yang dibuat harus merupakan URI yang valid dan harus diakhiri dengan '/' atau '#'. Salah satu contoh *namespace* yang valid adalah sebagai berikut.

```
http://www.w3.org/2000/01/rdf-schema#
```

Untuk mempersingkat *namespace*, dapat juga dibuat *prefix* dari *namespace* tersebut. Hal ini memungkinkan untuk merujuk pada kelas, properti, ataupun *instance* yang terdapat pada ontologi lainnya. Membuat *prefix* dapat dilakukan dengan menggunakan menu *add new prefix* yang terdapat pada bagian bawah *individual editor*. Contoh dari *prefix namespace* adalah sebagai berikut.

```
rdfs - http://www.w3.org/2000/01/rdf-schema#
```

***Tab* Ontoviz**

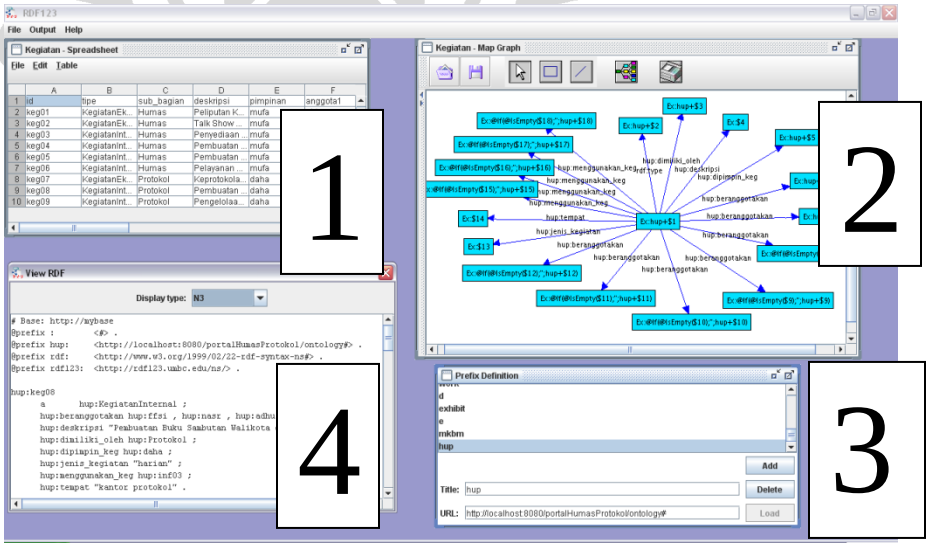
Tab ini digunakan untuk memvisualisasikan ontologi yang dibuat ke dalam bentuk graf. Dengan menggunakan menu yang terdapat dalam tab ini, graf yang

dihasilkan dapat disimpan dalam format gambar GIF. Graf yang dihasilkan tersebut kemudian ditampilkan dalam portal yang dikerjakan.

3.3.2 RDF123

RDF123 merupakan *free software* yang digunakan untuk membantu pemrosesan data-data *spreadsheet* yang didapatkan dari hasil survei ke dalam bentuk RDF (RDF123, 2009). Terdapat beberapa format RDF yang dapat dihasilkan dengan *tool* ini, salah satunya adalah format *notation* 3 (N3) yang digunakan dalam proyek ini. Untuk mengolah data *spreadsheet* menjadi N3 diperlukan empat tahap seperti yang terlihat pada Gambar 3.4, antara lain:

1. Mengubah format data *spreadsheet* menjadi format .csv (*comma separated value*) yang akan menjadi *input* dalam RDF123.
2. Membuat *mapping* graf yang digunakan untuk memetakan data .csv tersebut ke dalam bentuk N3.
3. Memilih *prefix* yang akan digunakan. *Prefix* yang terdapat dalam graf harus dibuat terlebih dahulu.
4. Langkah terakhir adalah memilih menu untuk menampilkan hasil pengubahan data menjadi bentuk N3.



Gambar 3.4 Konversi Data dengan RDF123

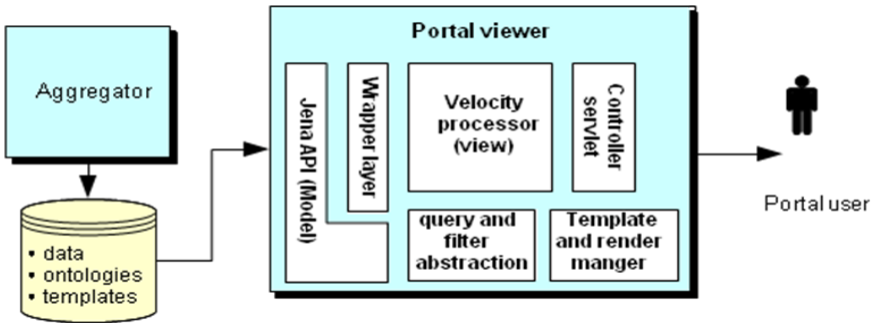
3.3.3 PortalCore

PortalCore merupakan bagian dari proyek SWAD-E yang dikembangkan sebagai *Semantic Web Environmental Directory demonstrator* (SWED Technical Resources, n.d.). *Tool* ini dapat digunakan untuk membuat portal berbasis *semantic web* dalam skala kecil.

PortalCore merupakan *tool* untuk membangun antarmuka dengan *faceted-browsing* berdasarkan kumpulan objek yang dijelaskan di RDF. *Input* yang diberikan pada portalCore berupa kumpulan data RDF yang diklasifikasikan dalam beberapa kategori beserta properti yang dimilikinya. Dengan *tool* ini, dapat dihasilkan:

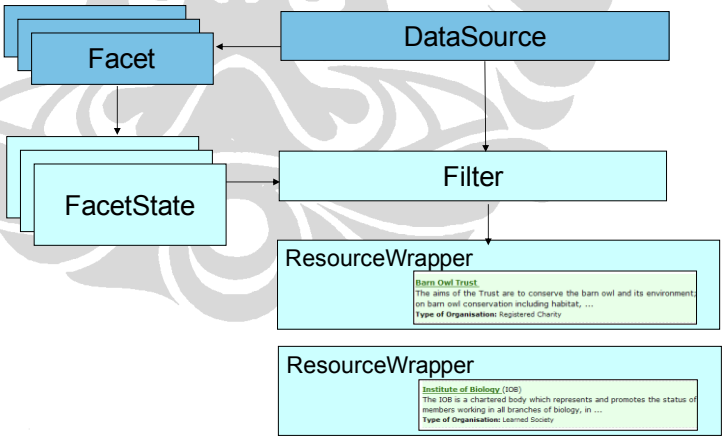
- *Web portal* untuk melihat dan menelusuri data.
- Antarmuka berbasis *faceted-browse*.
- Menampilkan data menggunakan *template* yang dapat dimodifikasi.
- Pencarian teks terintegrasi.
- Konfigurasi yang fleksibel.
- Pengambilan data dari berkas, basis data, dan *harvester*.
- Penelusuran jejak pencarian.

Struktur portalCore menggunakan pendekatan MVC (*Model-View-Controller*). Komponen utama berupa *portal viewer* yang menerima *input* data ontologi, RDF, dan *templates*. Model berupa Java *classes* yang menggunakan *library* Jena untuk membungkus data (ontologi dan data *instances*) yang dapat berasal dari banyak berkas dan juga basis data. *View* menggunakan *velocity template engine* untuk menampilkan halaman portal. *Controller* berupa Java *servlets* dengan sejumlah *built-in actions* (Aprilia, 2008). Struktur portalCore ditunjukkan pada Gambar 3.5.



Gambar 3.5 Struktur portalCore (Reynolds, 2004)

Model objects yang terdapat pada *portal viewer* terdiri dari *facet*, *facet state*, *datasource*, *filter*, dan *resource wrappers*. *Facet* merupakan kelompok atribut yang digunakan untuk pencarian informasi dalam portal. *Facet state* merupakan nilai atribut yang digunakan sebagai *filter* untuk menampilkan hasil pencarian. *Datasource* merupakan definisi sumber data dan juga konfigurasi dari portal yang dibuat. Dengan adanya *resource wrappers* maka memungkinkan *templates* untuk menampilkan data-data yang terdapat dalam *datasource* (Reynolds, 2004). Dalam proyek ini, untuk memodifikasi *templates* digunakan juga Scite *editor*. Gambar 3.6 menampilkan hubungan antara *model objects* tersebut.



Gambar 3.6 Hubungan Antar *Model Objects*

Akses informasi pada portalCore dapat dilakukan dengan dua cara, yaitu *browse* melalui *facet* dan juga *text search*. Sebelumnya, perlu diketahui bahwa untuk satu tipe objek dapat terdiri dari sekumpulan *facet*. Misalnya, ingin dilakukan pencarian data kucing bernama ‘Jerry’. Objek yang ada adalah *animal*, yang memiliki

tiga macam *facet*, yaitu *facet alphabetic* (berdasarkan alfabet dari nama), jenis kelamin, dan jenis kucing. Pencarian data mengenai 'Jerry' dapat dilakukan dengan memilih objek *animal* lalu memilih melalui *facet* mana data tersebut ingin dicari. Selanjutnya, akan ditampilkan daftar *instance* dari hasil penelusuran berdasarkan *facet* tersebut. Data 'Jerry' dapat dilihat dengan memilih *instance* 'Jerry' dari hasil pencarian tersebut.

Pencarian dengan *text search* menggunakan Lucene *text search engine* yang mengindeks setiap *resource* RDF berdasarkan properti yang dimilikinya (Reynolds, 2004). Sebagai contoh, pencarian kucing bernama 'Jerry' dengan menggunakan *text search* cukup memasukkan nama 'Jerry' pada *box text search*. Perlu diketahui bahwa hasil yang ditampilkan dapat berupa data 'Jerry' saja dan tidak menampilkan keterhubungan 'Jerry' dengan data lainnya. Hal ini bergantung pada apakah kata 'Jerry' digunakan sebagai *id* dari *instance* tersebut atau tidak. Jika kata 'Jerry' digunakan sebagai *id* dari *instance* tersebut, maka hasil yang akan ditampilkan pada *text search* juga akan menampilkan keterhubungan 'Jerry' dengan data lainnya. *Query* pada portalCore masih menggunakan RDQL (*RDF Query Language*), *predecessor* SPARQL yang sudah direkomendasikan oleh W3C. *Query* RDF sudah diabstraksi dalam portalCore sehingga tidak perlu membuat *query* secara langsung untuk dapat memperoleh data (Aprilia, 2008).

Selain *portal viewer* yang telah dibahas sebelumnya, terdapat juga komponen lain, yaitu *aggregator*. *Aggregator* berfungsi sebagai *service* untuk *scan* ataupun sebagai *harvester* data RDF secara periodik agar tetap ter-*update* (Aprilia, 2008). Pengembangan portal dalam proyek ini belum menggunakan komponen ini karena pengembangan portal yang masih berupa prototipe. Berdasarkan penjelasan di atas, penggunaan portalCore dapat membantu dalam mengembangkan sebuah portal berbasis *semantic web* tanpa harus memulainya dari awal sehingga dapat mempercepat proses pengembangan portal. Di samping itu, komponen-komponen yang terdapat dalam portalCore dianggap sudah dapat memenuhi kebutuhan portal yang dikembangkan.

3.4 Survei

Pada subbab 1.2 telah disebutkan bahwa domain permasalahan pada proyek ini adalah bagian Humas Protokol Pemerintah Kota Depok. Oleh karena itu, untuk memperoleh data-data yang terdapat pada bagian tersebut, pelaksana proyek melakukan survei ke bagian Humas Protokol Pemerintah Kota Depok. Untuk dapat melakukan survei, pelaksana proyek mengajukan surat perizinan survei terlebih dahulu. Permohonan perizinan tersebut memakan waktu yang cukup lama sehingga survei baru dapat dilakukan pada awal bulan Maret.

Survei dilakukan dua kali melalui wawancara dengan Kepala Subbagian Humas. Hasil survei pertama, pelaksana proyek memperoleh data-data kepegawaian dan infrastruktur yang digunakan pada bagian Humas Protokol. Data-data kegiatan Humas Protokol baru diperoleh setelah survei kedua. Di samping itu, pelaksana proyek mengajukan konsep awal ontologi yang dapat diterapkan pada bagian tersebut yang kemudian disetujui oleh Kepala Bagian Humas dengan sedikit perubahan. Keseluruhan data-data hasil survei tersebut dapat dilihat pada Lampiran A – Data Hasil Survei.

3.5 Pengembangan Ontologi

Pada subbab 2.2.3 telah dijelaskan mengenai langkah-langkah dasar dalam pengembangan ontologi. Secara umum, pengembangan ontologi pada proyek ini mengikuti langkah-langkah tersebut. Pada bagian ini akan dijelaskan mengenai langkah-langkah pengembangan ontologi yang dilakukan pada proyek ini.

3.5.1 *Ontology Scope*

Terdapat beberapa hal utama yang dilakukan pelaksana proyek dalam tahap ini, yaitu mengidentifikasi pengguna, kebutuhannya, tujuan pengembangan ontologi, dan juga kegunaan dari ontologi yang dikembangkan. Hal pertama yang dilakukan adalah mengidentifikasi pengguna ontologi. Sebelumnya, telah disebutkan bahwa kategori *e-government* yang diterapkan pada proyek ini adalah G2E dan G2G. Oleh karena itu,

pengguna dari ontologi ini adalah internal pemerintah, dalam kasus ini Bagian Humas Protokol Pemerintah Kota Depok.

Berikutnya dilakukan identifikasi kebutuhan ontologi. Ontologi yang dikerjakan harus dapat menghubungkan data-data yang tersebar dalam bagian Humas Protokol Pemerintah Kota Depok. Oleh karena itu, ontologi ini dapat mempermudah pencarian data yang dilakukan. Perlu diketahui bahwa data-data yang terdapat dalam ontologi ini, antara lain data kepegawaian, data kegiatan, data infrastruktur, data tugas dan peranan, dan data bagian Humas Protokol. Ontologi yang dikembangkan akan menghubungkan data-data tersebut.

Tujuan pengembangan ontologi yang dikerjakan adalah untuk mempermudah pencarian data-data yang tersebar tersebut. Ontologi yang dikembangkan akan memberikan hubungan semantik antar data sehingga dengan ontologi tersebut dapat diketahui hubungan antar data.

Ontologi yang dikembangkan berguna untuk mengatasi perbedaan pemahaman terhadap hubungan antar data. Di samping itu, hasil pencarian terhadap suatu data tidak hanya menampilkan data yang dicari saja melainkan juga menampilkan hubungan data tersebut dengan data yang lain. Hal ini dapat mempercepat kinerja dari seorang pegawai pemerintahan yang berujung pada peningkatan kinerja dari pemerintahan tersebut.

3.5.2 *Ontology Capture*

Setelah mengetahui cakupan dari ontologi yang dikembangkan, diketahui bahwa *domain* dari ontologi ini adalah bagian Humas Protokol Pemerintah Kota Depok. Terdapat lima konsep utama dalam ontologi yang dikembangkan, antara lain:

- **Bagian Humas Protokol**

Bagian Humas Protokol memiliki beberapa subbagian, yaitu Humas dan Protokol (Pemerintah Kota Depok, 2008). Masing-masing subbagian ini memiliki perbedaan dalam kegiatan, tugas dan peranannya.

- **Tugas dan Peranan**

Konsep ini mencakup tugas dan peranan dari masing-masing subbagian (Pemerintah Kota Depok, 2008).

- **Infrastruktur**

Masing-masing subbagian memiliki infrastruktur yang digunakan untuk menunjang kegiatannya masing-masing.

- **Pegawai**

Terdapat dua jenis pegawai yang terdapat pada bagian Humas Protokol, yaitu pegawai negeri sipil dan non pegawai negeri sipil. Di samping itu, terdapat juga beberapa jabatan dalam bagian ini, yaitu Kepala Bagian, Kepala Subbagian, dan Pelaksana.

- **Kegiatan**

Kegiatan yang terdapat pada Humas Protokol dibedakan menjadi dua jenis, yaitu kegiatan internal dan kegiatan eksternal. Kegiatan internal merupakan kegiatan yang dilakukan di dalam Pemerintah Kota Depok, sedangkan kegiatan eksternal dilakukan di luar Pemerintah Kota Depok.

3.5.3 *Ontology Encoding*

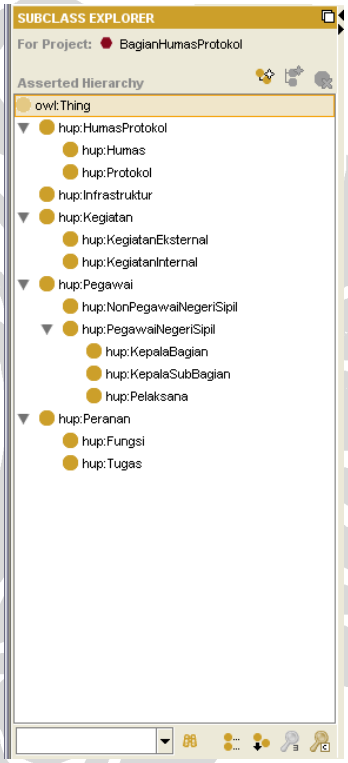
Setelah diketahui konsep utama dari ontologi yang dikembangkan, selanjutnya dilakukan *coding ontology* dengan menggunakan *tool* Protégé 3.3.1. Alasan pemilihan *tool* telah dijelaskan pada bagian 3.3. Ontologi yang dibuat disimpan dalam format berkas .owl yang kemudian menjadi salah satu *input* portal. Penjelasan lebih lanjut mengenai *input* portal dapat dilihat pada subbab 3.6.

Sebelum membuat kelas-kelas yang terlibat dalam ontologi, dilakukan pendefinisian *namespace* dan *prefix*-nya. *Namespace* yang digunakan dalam ontologi ini adalah `http://localhost:8080/portalHumasProtokol/ontology#`, sedangkan

prefix dari *namespace* tersebut adalah HUP. Berikut ini dijelaskan *coding ontology* yang dilakukan.

3.5.3.1 Pendefinisian Kelas

Untuk mengetahui kelas-kelas apa saja yang dibuat, dapat dilihat pada Gambar 3.7. Penjelasan dari kelas-kelas tersebut dapat dilihat pada Tabel 3.1 dan 3.2. Lima kelas pertama yang terdapat pada tabel tersebut merupakan kelas utama dari ontologi yang dibuat.



Gambar 3.7 Ontologi Humas Protokol *Classes*

Tabel 3.1 Definisi Kelas

Kelas	Definisi
hup:HumasProtokol	Kelas ini merepresentasikan konsep Bagian Humas dan Protokol. Kelas ini memiliki dua subkelas, yaitu hup:Humas dan hup:Protokol.
hup:Infrastruktur	Kelas ini merepresentasikan konsep infrastruktur. Kelas ini tidak memiliki subkelas.

Tabel 3.2 Definisi Kelas (Lanjutan)

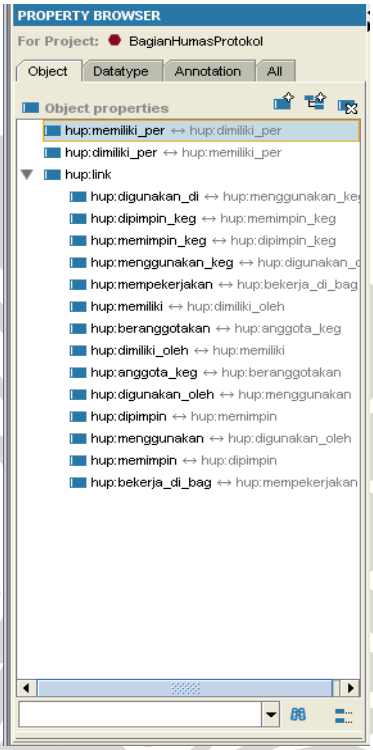
Kelas	Definisi
hup:Kegiatan	Kelas ini merepresentasikan konsep kegiatan. Kelas ini memiliki dua subkelas, yaitu hup:KegiatanInternal dan hup:KegiatanEksternal.
hup:Pegawai	Kelas ini merepresentasikan konsep Pegawai. Kelas ini memiliki dua subkelas, yaitu hup:NonPegawaiNegeriSipil dan hup:PegawaiNegeriSipil.
hup:Peranan	Kelas ini merepresentasikan konsep Tugas dan Peranan. Kelas ini memiliki dua subkelas, yaitu hup:Fungsi dan hup:Tugas.
hup:Humas	Merepresentasikan konsep subbagian Humas
hup:Protokol	Merepresentasikan konsep subbagian Protokol
hup:KegiatanInternal	Merepresentasikan konsep kegiatan internal
hup:KegiatanEksternal	Merepresentasikan konsep kegiatan eksternal
hup:NonPegawaiNegeriSipil	Merepresentasikan konsep non pegawai negeri sipil
hup:PegawaiNegeriSipil	Merepresentasikan konsep pegawai negeri sipil. Subkelas ini memiliki tiga subkelas lagi, yaitu hup:KepalaBagian, hup:KepalaSubBagian, dan hup:Pelaksana
hup:Fungsi	Merepresentasikan konsep fungsi bagian
hup:Tugas	Merepresentasikan konsep fungsi tugas
hup:KepalaBagian	Merepresentasikan konsep Kepala Bagian
hup:KepalaSubBagian	Merepresentasikan konsep Kepala Sub Bagian
hup:Pelaksana	Merepresentasikan konsep Pelaksana

3.5.3.2 Pendefinisian Properti

Terdapat dua jenis properti yang didefinisikan, yaitu *object properties* dan *datatype properties*. Berikut ini akan dijelaskan pendefinisian dari kedua properti tersebut.

a. *Object Properties*

Object properties yang dibuat akan menghubungkan suatu *instance* dengan *instance* yang lain. Pada Gambar 3.8 dapat dilihat *object property* apa saja yang terdapat pada ontologi yang dibuat. Penjelasan dari masing-masing *object property* dapat dilihat pada Tabel 3.3 dan 3.4.



Gambar 3.8 *Object Properties*

Tabel 3.3 Definisi *Object Properties*

Object Properties	**Definisi**
hup:memiliki_per	Menghubungkan hup:HumasProtokol dengan hup:Peranan. *Object properties* ini memiliki *inverse* hup:dimiliki_per.
hup:link	Merupakan *abstract link* yang digunakan untuk membantu visualisasi data ke dalam graf di dalam portal.

Tabel 3.4 Definisi *Object Properties* (Lanjutan)

Object Properties	Definisi
hup:memiliki	Menghubungkan hup:HumasProtokol dengan hup:Kegiatan. *Object properties* ini memiliki *inverse* hup:dimiliki_oleh
hup:bekerja_di_bag	Menghubungkan hup:Pegawai dengan hup:HumasProtokol. *Object properties* ini memiliki *inverse* hup:mempekerjakan.
hup:beranggotakan	Menghubungkan hup:Kegiatan dengan hup:Pegawai. *Object properties* ini memiliki *inverse* hup:anggota_keg.
hup:memimpin_keg	Menghubungkan hup:KepalaSubBagian dengan hup:Kegiatan. *Object properties* ini memiliki *inverse* hup:dipimpin_keg.
hup:digunakan_di	Menghubungkan hup:Infrastruktur dengan hup:Kegiatan. *Object properties* ini memiliki *inverse* hup:menggunakan_keg.
hup:menggunakan	Menghubungkan hup:Pegawai dengan hup:Infrastruktur. *Object properties* ini memiliki *inverse* hup:digunakan_oleh.
hup:memimpin	Menghubungkan hup:KepalaBagian dengan hup:HumasProtokol. *Object properties* ini memiliki *inverse* hup:dipimpin.

b. *Datatype Properties*

Datatype properties merupakan atribut yang dimiliki oleh suatu *instance*. *Datatype properties* yang dibuat dapat dilihat pada Gambar 3.9. Penjelasan dari masing-masing *datatype property* dapat dilihat pada Tabel 3.5 dan 3.6.

Gambar 3.9 *Datatype Properties*

Tabel 3.5 Definisi *Datatype Properties*

Datatype Properties	**Definisi**
hup:jenis_kelamin	Atribut ini dimiliki oleh hup:Pegawai. Merupakan jenis kelamin dari pegawai.
hup:namaBagian	Atribut ini dimiliki oleh hup:HumasProtokol. Merupakan nama bagian di Humas Protokol.
hup:jenis_kegiatan	Atribut ini dimiliki oleh hup:Kegiatan. Merupakan jenis dari kegiatan yang ada, yaitu harian, mingguan, dan bulanan.
hup:tingkat_ijazah	Atribut ini dimiliki oleh hup:Pegawai. Merupakan tingkat pendidikan dari masing-masing pegawai.
hup:golongan	Atribut ini dimiliki oleh hup:PegawaiNegeriSipil. Merupakan golongan yang dimiliki oleh Pegawai Negeri Sipil.

Tabel 3.6 Definisi *Datatype Properties* (Lanjutan)

Datatype Properties	Definisi
hup:deskripsi	Atribut ini dimiliki oleh hup:Kegiatan. Merupakan deskripsi dari suatu kegiatan.
hup:tahun_lulus	Atribut ini dimiliki oleh hup:Pegawai. Merupakan tahun kelulusan seorang pegawai dari tingkat pendidikan yang dimilikinya.
hup:tempat	Atribut ini dimiliki oleh hup:Kegiatan. Merupakan tempat kegiatan diselenggarakan.
hup:almamater	Atribut ini dimiliki oleh hup:Pegawai. Merupakan almamater yang dimiliki oleh seorang pegawai.
hup:nama_barang	Atribut ini dimiliki oleh hup:Infrastruktur. Merupakan nama barang dari suatu infrastruktur.
hup:nama	Atribut ini dimiliki oleh hup:Pegawai. Merupakan nama dari pegawai.
hup:nip	Atribut ini dimiliki oleh hup:Pegawai. Merupakan nomor induk pokok dari pegawai.
hup:agama	Atribut ini dimiliki oleh hup:Pegawai.
hup:catatan_mutasi	Atribut ini dimiliki oleh hup:Pegawai. Merupakan catatan kepindahan dari seorang pegawai.
hup:deskripsi_per	Atribut ini dimiliki oleh hup:Peranan. Merupakan deskripsi dari suatu peranan.
hup:tanggal_lahir	Atribut ini dimiliki oleh hup:Pegawai. Merupakan tanggal lahir dari pegawai.
hup:jumlah	Atribut ini dimiliki oleh hup:Infrastruktur. Merupakan jumlah barang dari infrastruktur tersebut.
hup:tempat_lahir	Atribut ini dimiliki oleh hup:Pegawai. Merupakan tempat lahir dari pegawai.

3.5.4 *Ontology Integration, Evaluation,* dan *Documentation*

Proyek ini tidak me-*reuse* ontologi ataupun menggunakan ontologi lainnya sehingga tidak dilakukan tahap integrasi. Untuk tahap evaluasi, dilakukan pada saat

pengujian portal. Penjelasan mengenai evaluasi portal dapat dilihat pada Bab IV mengenai hasil dan pembahasan. Untuk tahap dokumentasi ontologi, tidak dilakukan pendokumentasian ontologi. Penjelasan mengenai ontologi yang dibuat sudah terdapat pada penjelasan mengenai *ontology encoding*.

3.6 Input Portal

Portal berbasis *semantic web* yang menggunakan portalCore membutuhkan data dan *rules* untuk dapat berjalan. Data pada *portal* dibagi menjadi dua, yaitu ontologi dan *instances data. Rules* dibutuhkan untuk proses *inference* agar memperoleh data atau relasi baru. Data dan *rules* inilah yang menjadi input portal.

3.6.1 Persiapan Data

Pada bagian ini, akan dijelaskan tentang persiapan data, yaitu ontologi dan *instance data*, yang dibutuhkan sebagai *input* portal. Ontologi pada penelitian ini, dirancang sendiri dan diberi nama ontologi HUP (berasal dari Humas dan Protokol). Ontologi HUP ini dirancang berdasarkan survei yang dilakukan pada Bagian Humas dan Protokol Pemerintah Kota Depok. Sama halnya ontologi, *instances data* pada *portal* ini juga berasal dari survei yang dilakukan pada Bagian Humas dan Protokol Pemerintah Kota Depok.

3.6.1.1 Ontologi HUP

Seperti yang telah dijelaskan pada Subbab 3.5, ontologi HUP memiliki lima kelas utama, yaitu HumasProtokol, Pegawai, Kegiatan, Infrastruktur, dan Peranan, 11 subkelas, 9 *object properties*, serta 18 *datatype properties.* Rincian mengenai kelas, *object* properties, dan *datatype properties* juga telah dibahas pada Subbab 3.5.

Ontologi HUP dibuat menggunakan *tool* Protégé 3.3.1 dan disimpan dalam bahasa OWL. Berikut merupakan potongan *code* dari ontologi HUP yang ditulis dalam bahasa OWL. Kode ontologi HUP seutuhnya dapat dilihat pada Lampiran B – Kode Ontologi.

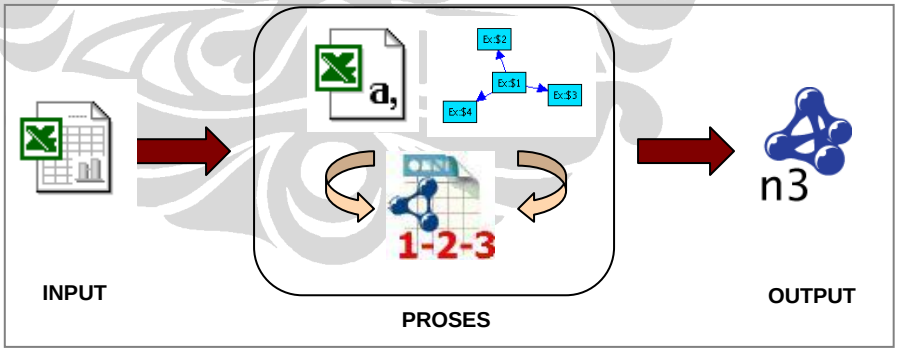
```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://localhost:8080/portalHumasProtokol/ontology-01#"
    xmlns:hup="http://localhost:8080/portalHumasProtokol/ontology#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://localhost:8080/portalHumasProtokol/ontology-01">
  <owl:Ontology rdf:about="">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></owl:versionInfo>
  </owl:Ontology>
  <owl:Class rdf:about="http://localhost:8080/portalHumasProtokol/ontology#Pelaksana">
    <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost:8080/portalHumasProtokol/ontology#KepalaSubBagian"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost:8080/portalHumasProtokol/ontology#KepalaBagian"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class
rdf:about="http://localhost:8080/portalHumasProtokol/ontology#PegawaiNegeriSipil"/>
    </rdfs:subClassOf>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Pelaksana</rdfs:label>
  </owl:Class>
  .
  .
  .
  .
  <owl:ObjectProperty
rdf:about="http://localhost:8080/portalHumasProtokol/ontology#dipimpin">
  <rdfs:subPropertyOf
rdf:resource="http://localhost:8080/portalHumasProtokol/ontology#link"/>
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <owl:inverseOf>
      <owl:ObjectProperty
rdf:about="http://localhost:8080/portalHumasProtokol/ontology#memimpin"/>
  </owl:inverseOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

3.6.1.2 *Instances Data*

Data lain yang dibutuhkan oleh portalCore adalah *instances data*. Data tersebut menggunakan format RDF dengan sintaks N3. *Instances data* yang perlu disiapkan sesuai dengan lima kelas utama dari ontologi HUP, yaitu HumasProtokol, Pegawai, Kegiatan, Infrastruktur, dan Peranan. Dengan kata lain terdapat lima berkas *instances data* yang berformat N3. Sumber dari *instances data* berasal dari hasil survei yang dilakukan pada Bagian Humas dan Protokol Pemerintah Kota Depok.

Instances data disiapkan dengan menggunakan *tool converter* RDF123 yang memroses berkas *spreadsheet* menjadi format RDF. Berikut merupakan proses yang dilakukan:

a) Menyiapkan berkas .xls menjadi .csv (*comma separated values*). Berkas .csv menjadi input dari RDF123.
b) Membuat *map graph* yang digunakan untuk memetakan data .csv tersebut ke dalam bentuk .n3.
c) Memilih *prefix* yang akan digunakan. *Prefix* yang terdapat dalam graf harus dibuat terlebih dahulu.
d) Memilih menu untuk menampilkan hasil pengubahan data menjadi bentuk .n3.



Gambar 3.10 Transformasi Data (Aprilia, 2008)

Langkah pertama adalah menyiapkan input data. Berkas *spreadsheet* yang didapat dari survei adalah satu buah berkas .xls yang berisi data pegawai untuk input kelas Pegawai. Contoh data pegawai tersebut dapat dilihat pada Tabel 3.7.

Tabel 3.7 Contoh Data Pegawai Bagian Humas dan Protokol Pemerintah Kota Depok

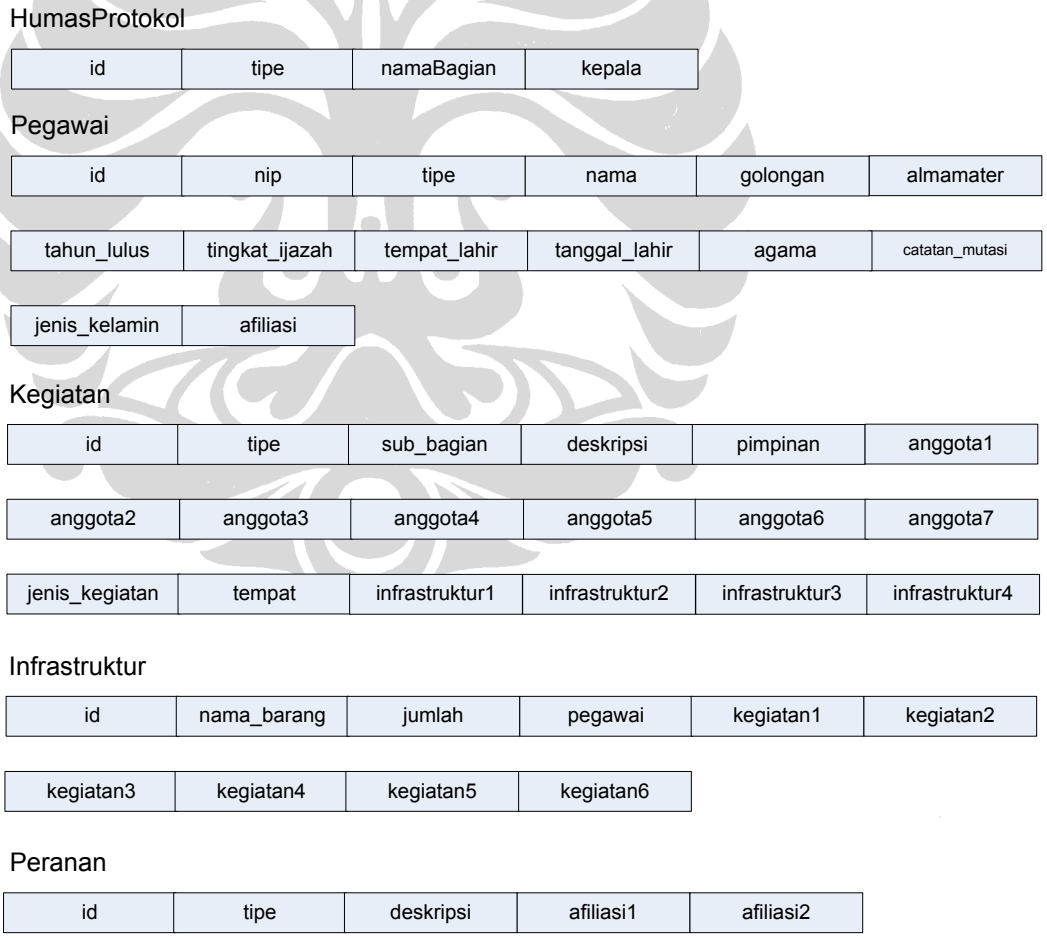
NO	NAMA	NIP	PANGKAT		JABATAN		
			GOL	TMT.GOL	NAMA	TMT	Sub Bagian
1	Eko Herwiyanto, AP	010 249 559	III/d	01-10-2005	Kabag Humas dan Protokol	31-12-2008	
2	Muhammad Fahmi, ST. M.Si	480 126 069	III/c	01-10-2008	Kasubag Humas	31-12-2008	
3	Dani Hamdani, S.STP	010 264 839	III/b	01-10-2008	Kasubag Protokol	31-12-2008	
4	Rusmini	010 175 817	III/b	01-04-2007	Pelaksana	-	Humas
5	Minar Rosdiana	400 033 131	III/b	01-04-2006	Pelaksana	-	Humas
6	Dra. Hartikah	480 123 288	III/b	01-04-2007	Pelaksana	-	Humas
7	Ahmad, S.STP	010 267 122	III/a	01-12-2001	Pelaksana	-	Protokol
8	Fathir Fajar Sidiq, S.STP	010 268 932	III/a	01-12-2002	Pelaksana	-	Protokol
9	R. Tranto Dwi Hardjono, SH	480 148 203	III/a	01-04-2006	Pelaksana	-	Protokol
10	Nasrullah	480 133 066	II/c	01-01-2005	Pelaksana	-	Humas
11	Shillawati	480 145 936	II/c	01-04-2006	Pelaksana	-	Humas
12	Erwin Narto	480 119 637	II/b	01-12-2001	Pelaksana	-	Protokol
13	Retno Sustyaningsih	480 123 296	II/b	01-12-2002	Pelaksana	-	Protokol
14	Djoko Wahyudi	160 033 051	II/a	01-04-1998	Pelaksana	-	Humas
15	Riswati	010 163 253	II/a	01-04-1999	Pelaksana	-	Humas
16	Erwan Mindaya	480 126 143	II/a	01-12-2003	Pelaksana	-	Humas

Dua kelas lainnya, yaitu Kegiatan dan Infrastruktur didapat dari hasil wawancara. Dua kelas terakhir, yaitu HumasProtokol dan Peranan didapat dari

Peraturan Sekretaris Walikota Depok. Untuk dapat diproses dengan *tool* RDF123, data kelima kelas tersebut harus dalam format .csv. Masing-masing kelas dipisah datanya dalam satu berkas .csv sehingga terdapat lima berkas .csv. Berikut merupakan contoh berkas .csv untuk kelas HumasProtokol.

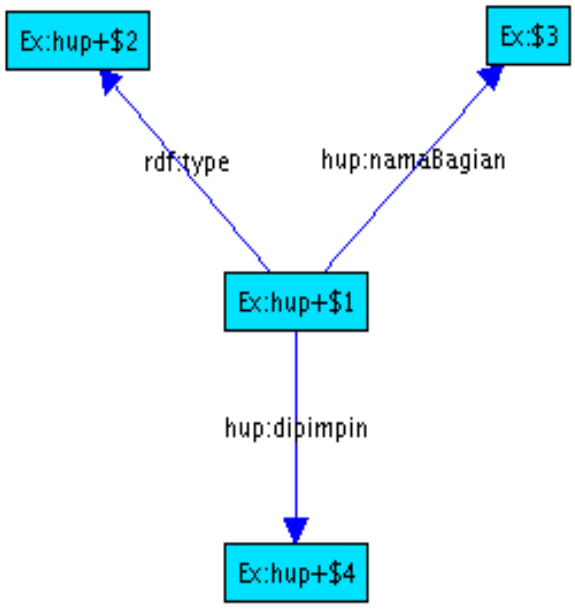
```
id,tipe,namaBagian,kepala
HUP,HumasProtokol,Humas Protokol, Eko_Herwiyanto_AP
HUM,Humas,Humas,fahmi
PRO,Protokol,Protokol, Dani_Hamdani_S_STP
```

Baris pertama atau *header* tabel pada berkas .csv adalah *properties* dari setiap kelas. *Header* tabel setiap kelas dapat dilihat pada Gambar 3.11. Setiap satu baris berikutnya merupakan satu *instance* dari kelas yang bersangkutan.



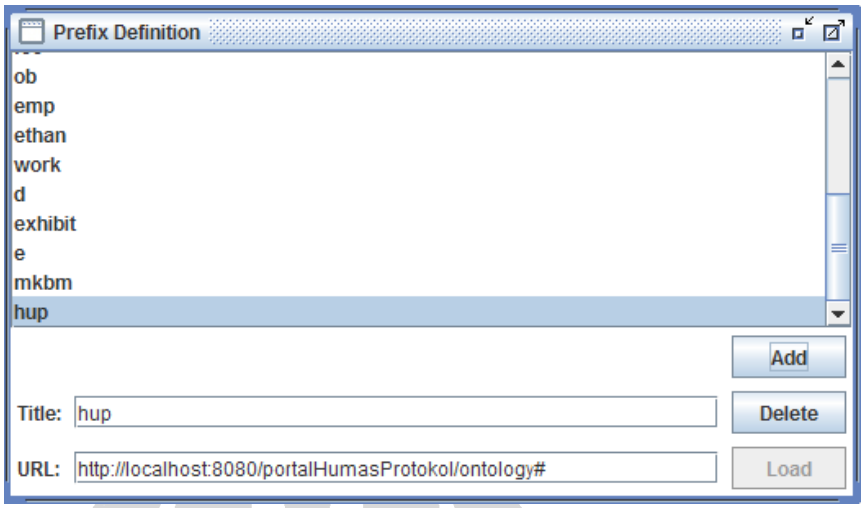
Gambar 3.11 *Header* Tabel Berkas .csv

Langkah kedua adalah membuat *map graph*, suatu *template* dalam bentuk *graph* RDF dengan format yang telah didefinisikan oleh RDF123. Format yang dimaksud seperti penggunaan *namespace* Ex:, $n, *logical expression* @If(B;E;E) dan @IsEmpty(E) (Aprilia, 2008). *Map graph* disimpan dengan tipe berkas .xgmml dan dapat di-*convert* juga menjadi format RDF. Tiap satu berkas .csv dibuat *map graph* masing-masing sehingga terdapat lima berkas .xgmml. Gambar 3.12 merupakan salah satu *map graph* untuk kelas HumasProtokol.



Gambar 3.12 *Map Graph* HumasProtokol

Langkah ketiga adalah memilih *prefix* yang akan digunakan pada berkas .n3. *Prefix* yang digunakan adalah *prefix* hup, rdf, dan rdf123. *Prefix* rdf dan rdf123 telah ada pada RDF123, sedangkan *prefix* hup belum ada. Untuk itu, harus ditambahkan terlebih dahulu *prefix* hup pada *window Prefix Definition* seperti pada Gambar 3.13. Pada bagian *Title* dimasukkan nilai "hup" dan pada bagian URL dimasukkan nilai `http://localhost:8080/portalHumasProtokol/ontology#` .



Gambar 3.13 *Prefix Definition* pada RDF123

Langkah terakhir adalah melakukan pemetaan berkas .csv dengan berkas .xgmml yang bersesuaian dan menyimpan *output* dari RDF123 dalam bentuk .n3. Contoh berkas .n3 pada kelas HumasProtokol adalah sebagai berikut.

```
# Base: http://mybase
@prefix :        <#> .
@prefix hup:     <http://localhost:8080/portalHumasProtokol/ontology#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdf123:  <http://rdf123.umbc.edu/ns/> .

hup:HUP
      a       hup:HumasProtokol ;
      hup:dipimpin hup: Eko_Herwiyanto_AP ;
      hup:namaBagian "Humas Protokol" .

hup:HUM
      a       hup:Humas ;
      hup:dipimpin hup:fahmi ;
      hup:namaBagian "Humas" .

hup:PRO
      a       hup:Protokol ;
      hup:dipimpin hup: Dani_Hamdani_S_STP ;
      hup:namaBagian "Protokol" .
```

Berkas .csv, *map graph*, dan .n3 dapat dilihat pada Lampiran C – *Instances Data*. Ringkasan proses persiapan *instances data* sebagai *input* portalCore dengan RDF123 adalah sebagai berikut:

1 berkas .xls → 5 berkas .csv + 5 berkas .xgmml → 5 berkas .n3

Total jumlah *instances* yang dibuat untuk *input* data portalCore adalah sebagai berikut:

- HumasProtokol = 3
- Pegawai = 23
- Kegiatan = 9
- Infrastruktur = 10
- Peranan = 16

3.6.2 Pendefinisian *Rules*

Untuk mendapatkan data baru dari data yang sudah ada, dibuat *rules* untuk melakukan proses *inference*. *Rules* ini dijadikan sebagai *input portal* bersama dengan ontologi dan *instances* data, disimpan dalam berkas terpisah (.rules). Proses *inference* pada *portal* menggunakan GenericRuleReasoner Jena sehingga struktur dan sintaks *rules* mengikuti *rule engine* tersebut (Aprilia, 2008).

Terdapat beberapa RDFS *closure rules* yang telah didefinisikan pada portalCore. RDFS *closure rules* adalah:

```
[rdfs2:  (?x ?p ?y), (?p rdfs:domain ?c) -> (?x rdf:type ?c)]
[rdfs3:  (?x ?p ?y), (?p rdfs:range ?c) -> (?y rdf:type ?c)]
[rdfs7:  (?a rdf:type rdfs:Class) -> (?a rdfs:subClassOf ?a)]
[rdfs8:  (?a rdfs:subClassOf ?b), (?b rdfs:subClassOf ?c) -> (?a rdfs:subClassOf ?c)]
[rdfs9:  (?x rdfs:subClassOf ?y), (?a rdf:type ?x) -> (?a rdf:type ?y)]
```

Rule rdfs2 menyatakan bahwa jika x memiliki relasi p dengan y dan p adalah *domain* dari c, maka x adalah *instance* dari c.

Rule rdfs3 menyatakan bahwa jika x memiliki relasi p dengan y dan p adalah *range* dari c, maka y adalah *instance* dari c.

Rule rdfs7 menyatakan suatu *class* adalah *subclass* dari dirinya sendiri.

Rule rdfs8 menyatakan jika a *subclass* b, dan b *subclass* c, maka a juga subclass c.

Rule rdfs9 menyatakan jika x *subclass* y, a *instance* x, maka a juga *instance* y.

Selain RDFS *closure rules* yang telah didefinisikan pada portalCore, dibutuhkan juga *rules* lain untuk melakukan *inferences* data. Terdapat sepuluh *rules* yang didefinisikan, yaitu:

a) Untuk membantu visualisasi portal.

```
(?A rdf:type hup:Kegiatan), (?A hup:deskripsi ?B) -> (?A rdfs:label ?B) .
(?A rdf:type hup:Pegawai), (?A hup:nama ?B) -> (?A rdfs:label ?B) .
(?A rdf:type hup:Infrastruktur), (?A hup:nama_barang ?B) -> (?A rdfs:label ?B)
   .
(?A rdf:type hup:HumasProtokol),  (?A hup:namaBagian ?B) -> (?A rdfs:label ?B)
   .
(?A hup:dimiliki_tug ?B), (?A hup:deskripsi_per ?C) -> (?B hup:memiliki_tug
   ?A), (?A rdfs:label ?C) .
(?A hup:dimiliki_fung ?B), (?A hup:deskripsi_per ?C) -> (?B hup:memiliki_fung
   ?A), (?A rdfs:label ?C) .
```

b) Untuk mengetahui siapa anggota dari masing-masing kegiatan.

```
(?A hup:beranggotakan ?B) -> (?B hup:anggota_keg ?A) .
```

Jika kegiatan A beranggotakan pegawai B, maka pegawai B adalah anggota dari kegiatan A (*inverse*).

c) Untuk mengetahui siapa anggota dari masing-masing bagian.

```
(?A hup:bekerja_di_bag ?B) -> (?B hup:mempekerjakan ?A) .
(?A hup:bekerja_di_bag hup:HUM) -> (?A hup:bekerja_di_bag hup:HUP) .
(?A hup:bekerja_di_bag hup:PRO) -> (?A hup:bekerja_di_bag hup:HUP) .
```

Jika pegawai A bekerja di bagian B, maka bagian B mempekerjakan pegawai A (*inverse*).

Jika pegawai A bekerja di Subbagian Humas, maka A juga bekerja di Bagian Humas dan Protokol. *Rule* ini digunakan untuk mengetahui pegawai yang bekerja di Subbagian Humas.

Jika pegawai A bekerja di Subbagian Protokol, maka A juga bekerja di Bagian Humas dan Protokol. *Rule* ini digunakan untuk mengetahui pegawai yang bekerja di Subbagian Protokol.

d) Untuk mengetahui siapa pimpinan masing-masing bagian.

```
?A hup:memimpin ?B) -> (?B hup:dipimpin ?A) .
(?A hup:dipimpin ?B) -> (?B hup:memimpin ?A) .
```

Jika pegawai A memimpin bagian B, maka bagian B dipimpin oleh pegawai A (*inverse*).

Jika bagian A dipimpin oleh pegawai B, maka pegawai B memimpin bagian A (*inverse*).

e) Untuk mengetahui kegiatan apa yang dimiliki oleh masing-masing bagian.

```
(?A hup:dimiliki_oleh ?B) -> (?B hup:memiliki ?A) .
(?A hup:dimiliki_oleh hup:HUM) -> (?A hup:dimiliki_oleh hup:HUP) .
(?A hup:dimiliki_oleh hup:PRO) -> (?A hup:dimiliki_oleh hup:HUP) .
```

Jika kegiatan A dimiliki oleh bagian B, maka bagian B memiliki kegiatan A (*inverse*).

Jika kegiatan A dimiliki oleh Subbagian Humas, maka kegiatan A juga dimiliki oleh Bagian Humas dan Protokol. *Rule* ini digunakan untuk mengetahui kegiatan yang dimiliki oleh Subbagian Humas.

Jika kegiatan A dimiliki oleh Subbagian Protokol, maka kegiatan A juga dimiliki oleh Bagian Humas dan Protokol. *Rule* ini digunakan untuk mengetahui kegiatan yang dimiliki oleh Subbagian Protokol.

f) Untuk mengetahui tugas dari masing-masing bagian.

```
(?A hup:dimiliki_tug ?B) -> (?B hup:memiliki_tug ?A) .
(?A hup:dimiliki_tug hup:HUM) -> (?A hup:dimiliki_tug hup:HUP) .
(?A hup:dimiliki_tug hup:PRO) -> (?A hup:dimiliki_tug hup:HUP) .
```

Jika tugas A dimiliki oleh bagian B, maka bagian B memiliki tugas A (*inverse*).

Jika tugas A dimiliki oleh Subbagian Humas, maka tugas A juga dimiliki oleh Bagian Humas Protokol. *Rule* ini digunakan untuk mengetahui tugas yang dimiliki oleh Subbagian Humas.

Jika tugas A dimiliki oleh Subbagian Protokol, maka tugas A juga dimiliki oleh Bagian Humas Protokol. *Rule* ini digunakan untuk mengetahui tugas yang dimiliki oleh Subbagian Protokol.

g) Untuk mengetahui fungsi dari masing-masing bagian.

```
(?A hup:dimiliki_fung ?B) -> (?B hup:memiliki_fung ?A) .
(?A hup:dimiliki_fung hup:HUM) -> (?A hup:dimiliki_fung hup:HUP) .
(?A hup:dimiliki_fung hup:PRO) -> (?A hup:dimiliki_fung hup:HUP) .
```

Jika fungsi A dimiliki oleh bagian B, maka bagian B memiliki fungsi A (*inverse*).

Jika fungsi A dimiliki oleh Subbagian Humas, maka fungsi A juga dimiliki oleh Bagian Humas Protokol. *Rule* ini digunakan untuk mengetahui fungsi yang dimiliki oleh Subbagian Humas.

Jika fungsi A dimiliki oleh Subbagian Protokol, maka fungsi A juga dimiliki oleh Bagian Humas Protokol. *Rule* ini digunakan untuk mengetahui fungsi yang dimiliki oleh Subbagian Protokol.

h) Untuk mengetahui siapa pimpinan kegiatan.

```
(?A hup:dipimpin_keg ?B) -> (?B hup:memimpin_keg ?A) .
(?A hup:memimpin_keg ?B) -> (?A hup:anggota_keg ?B) .
```

Jika kegiatan A dipimpin oleh pegawai B, maka pegawai B memimpin kegiatan A (*inverse*).

Jika pegawai A memimpin kegiatan B, maka A juga merupakan anggota kegiatan B.

i) Untuk mengetahui infrastruktur yang digunakan pada suatu kegiatan.

```
(?A hup:menggunakan_keg ?B) -> (?B hup:digunakan_di ?A) .
```

Jika kegiatan A menggunakan infrastruktur B, maka infrastruktur B digunakan pada kegiatan A (*inverse*).

j) Untuk mengetahui siapa yang bertanggung jawab pada suatu infrastruktur.

```
(?A hup:digunakan_oleh ?B) -> (?B hup:menggunakan ?A) .
```

Jika infrastruktur A digunakan oleh pegawai B, maka pegawai B menggunakan infrastruktur A (*inverse*).

3.7 Konfigurasi Portal

Untuk dapat digunakan, portalCore harus dikonfigurasi terlebih dahulu sesuai dengan kebutuhan. Proses konfigurasi portalCore dilakukan melalui satu berkas RDF (…/WEB-INF/config/sources.n3) yang mendefinisikan *basic properties*, *facets*, dan *templates* dari *datasources*. Format penulisan pada berkas RDF tersebut menggunakan *prefix* pcv: dan *namespace* `http://jena.hpl.hp.com/2003/04/portal-config-vocab#`. Konfigurasi portalCore dilakukan berdasarkan dokumentasi dari portalCore (SWAD-E Portal Customization, n.d.).

3.7.1 *Basic Properties Datasource*

Konfigurasi ini menentukan *datasources* yang digunakan oleh portalCore. *Datasources* tersebut termasuk ontologi dan *instances data* yang digunakan, *rule* yang dipakai untuk proses data, *facet* dan *templates* yang digunakan dalam tampilan portal, serta *stylesheet* dan beberapa *property* lain yang dipakai untuk konfigurasi *portal*.

Instance datasource yang didefinisikan pada berkas sources.n3 memiliki *type* pcv:DataSources. *Property* yang perlu didefinisikan adalah *encoding* dan *order number*. *Encoding* adalah *string* yang akan menjadi identitas dari *datasource* pada *http request* dari portal. *Order number* merupakan bilangan bulat yang menentukan urutan tampilan dari *datasources*. Makin kecil *order number*, *datasources* makin diletakkan pada urutan teratas.

Instance datasource yang didefinisikan adalah Pegawai, Kegiatan, dan Semua Objek. *Instance* Pegawai dan Kegiatan menggunakan *property* pcv:filterOnType untuk memfilter data yang ditampilkan hanya dari kelas Pegawai dan Kegiatan. Untuk *instance* Semua Objek, tidak menggunakan *property* pcv:filterOnType karena menampilkan seluruh kelas. *Instance data*, ontologi, *rule*, dan *style* yang digunakan oleh portalCore juga didefinisikan di berkas sources.n3 ini. Berikut merupakan potongan kode sources.n3 yang mendefinisikan objek Pegawai.

```
[]  rdf:type pcv:DataSource ;
    rdfs:label "Pegawai" ;
    pcv:encoding "peg";
    pcv:order "2"^^xsd:integer ;
    dc:description "Prototipe Portal Humas Protokol" ;
    pcv:sourceURL <portal://data/humasprotokol.n3> ;
    pcv:sourceURL <portal://data/infrastruktur.n3> ;
    pcv:sourceURL <portal://data/kegiatan.n3> ;
    pcv:sourceURL <portal://data/pegawai.n3> ;
    pcv:sourceURL <portal://data/peranan.n3> ;

    pcv:ontologySourceURL <portal://data/BagianHumasProtokol.owl> ;

    pcv:closureRulesURL   <portal://data/portalHumasProtokol.rules> ;

    pcv:baseRelationProperty hup:link;
    pcv:filterOnType hup:Pegawai;

    pcv:styleSheet "site.css" ;
    . . .
```

3.7.2 Pendefinisian *Facets*

Facets berfungsi untuk proses navigasi dari data. Pendefinisian *facet* menggunakan *property* pcv:facet. *Instance* dari *facet* tersebut didefinisikan kembali dengan *property* pcv:Facet. *Property* pcv:linkProp digunakan untuk menampilkan objek yang dicari berdasarkan *datatype* atau *object property*. Berikut merupakan potongan kode pendefinisian *facet* untuk Pegawai.

```
pcv:facet hup:namaFacet ;
pcv:facet hup:tipeFacet ;
pcv:facet hup:bagianFacet ;
pcv:facet hup:kegiatanFacet ;
. . .

hup:kegiatanFacet a pcv:Facet;
        rdfs:label "Kegiatan" ;
        pcv:linkProp hup:anggota_keg;
        pcv:order "4"^^xsd:integer;
.
. . .
```

Facet hup:kegiatanFacet akan menampilkan kegiatan-kegiatan yang diikuti oleh Pegawai. Contoh tampilan dari *facet* hup:kegiatanFacet dapat dilihat pada Gambar 3.14.

Gambar 3.14 *Facet* Kegiatan untuk *Object* Pegawai

Facet memiliki tiga tipe, yaitu *flat*, *alpharange*, dan *hierarchical*. Contoh *facet* hup:kegiatanFacet merupakan *facet* bertipe *flat*, yang menampilkan *instance* yang didapatkan. *Alpharange facet* menampilkan huruf pertama dari *instance* yang didapatkan. Contohnya digunakan untuk *facet* hup:namaFacet (Nama) yang menampilkan huruf pertama dari nama orang. *Hiearchical facet* akan menampilkan

classes dan hierarkinya seperti contoh *facet* hup:tipeFacet (Jenis Pegawai) yang menampilkan jenis Pegawai. Gambar 3.15 memperlihatkan contoh *alpharange* dan *hierarchical facet*.

Gambar 3.15 *Alpharange* dan *Hierarchical Facet*

Seperti yang telah disebutkan pada Subbab 3.7.1 bahwa terdapat tiga objek yang didefinisikan, yaitu Pegawai, Kegiatan, dan Semua Objek. Objek Pegawai memiliki empat *facets*, objek Kegiatan memiliki tiga *facets*, dan Semua Objek memiliki satu *facet*. Daftar *facet* masing-masing objek dapat dilihat pada Tabel 3.8 dan 3.9.

Tabel 3.8 Daftar *Facets*

Object	*Facets*	Tipe *Facet*	Deskripsi
Pegawai	Nama	*Alpharange*	Menampilkan nama-nama Pegawai
	Jenis	*Hierarchical*	Menampilkan jenis Pegawai (Pegawai Negeri Sipil atau Non Pegawai Negeri Sipil)

Tabel 3.9 Daftar *Facets* (Lanjutan)

Object	*Facets*	Tipe *Facet*	Deskripsi
Pegawai	Bagian	*Flat*	Menampilkan bagian-bagian yang ada pada Humas Protokol
	Kegiatan	*Flat*	Menampilkan kegiatan-kegiatan yang diikuti oleh Pegawai
Kegiatan	Nama	*Alpharange*	Menampilkan nama-nama Kegiatan
	Jenis	*Hierarchical*	Menampilkan jenis Kegiatan (Kegiatan Internal atau Kegiatan Eksternal)
	Bagian	*Flat*	Menampilkan bagian yang memiliki Kegiatan
Semua Objek	Semua Objek	*Hierarchical*	Menampilkan keseluruhan kelas

3.7.3 Pendefinisian *Templates*

Pendefinisian *template* juga dilakukan pada berkas sources.n3. *Template* digunakan sebagai tempat untuk menampilkan data dari portal. Jumlah *template* yang didefinisikan sesuai dengan jumlah kelas yang akan ditampilkan. Dalam hal ini, terdapat lima *template* untuk lima kelas, yaitu HumasProtokol, Pegawai, Kegiatan, Infrastruktur, dan Peranan. Mengenai *template* yang dibuat dengan VTL (*Velocity Template Language*) akan dibahas pada bagian berikutnya. Berikut potongan kode dari pendefinisian *templates* untuk semua kelas.

```
pcv:template [a pcv:Template;
    pcv:templateContext "page" ;
    pcv:templatePath      <portal://templates/pagePelaksana.vm> ;
pcv:templateClass     hup:Pegawai;
];
pcv:template [a pcv:Template;
    pcv:templateContext "page" ;
    pcv:templatePath      <portal://templates/pageHumasProtokol.vm> ;
pcv:templateClass     hup:HumasProtokol;
];
```

```
pcv:template [a pcv:Template;
    pcv:templateContext "page" ;
    pcv:templatePath      <portal://templates/pageKegiatan.vm> ;
pcv:templateClass     hup:Kegiatan;
];
pcv:template [a pcv:Template;
    pcv:templateContext "page" ;
    pcv:templatePath      <portal://templates/pageInfrastruktur.vm> ;
pcv:templateClass     hup:Infrastruktur;
];
pcv:template [a pcv:Template;
    pcv:templateContext "page" ;
    pcv:templatePath      <portal://templates/pagePeranan.vm> ;
pcv:templateClass     hup:Peranan;
];
```

3.8 Tampilan Portal

Tampilan portal atau visualisasi *template* pada portalCore berhubungan langsung dengan *velocity template* karena *template* pada portalCore menggunakan *Velocity Template Language* (VTL). *Velocity* merupakan *template engine* yang dirancang khusus dalam penerapan model MVC yang berbasis Java. Bahasa yang digunakan adalah *scripting language*, seperti JSP atau PHP. Tidak seperti JSP, *velocity* tidak mengizinkan Java *code* yang *embedded* pada suatu *template page.* Jadi, pemisahan antara Java *code* dan *HTML template code* benar-benar dibedakan pada *velocity*.

Ada tiga kategori *page* yang menjadi tampilan *portal*, yaitu *browse page*, *results page,* dan *resources page*. Kategori *page* berdasarkan urutan tampilannya dapat dilihat pada Gambar 3.16. *Browse page* merupakan tampilan dengan *facets* untuk suatu tipe objek. *Results page* merupakan tampilan yang berisi hasil *browse* maupun *search*. Sedangkan *resource page* merupakan tampilan yang memberikan deskripsi suatu *resource. Template* yang didefinisikan pada tahap ini merupakan *resource page* (Aprilia, 2008).

Gambar 3.16 Jenis Halaman Portal (Aprilia, 2008)

Seperti yang telah disebutkan pada bagian pendefinisian *templates*, ada lima berkas *templates* yang dibuat untuk menampilkan data dari lima kelas. Selain itu, modifikasi *template* juga dilakukan pada *header page* dan *stylesheet* untuk mengatur *look-and-feel. Template* yang dibuat bersifat dinamis, artinya isi *page* sesuai dengan *resource* yang diambil. *Value* dari *resource* dapat dijadikan *link* menuju *page* lain. Berikut contoh kode untuk menampilkan *resource* "anggota kegiatan" berdasarkan *property* "beranggotakan" pada halaman *template* "Kegiatan".

```
1   #if ($resource.hasProperty("hup:beranggotakan"))
2   <tr><th valign="top" nowrap>Anggota</th>
3   <td valign="top">
4   <ul>
5   #set($count=0)
6       #foreach ($p in $resource.findProperties("hup:beranggotakan"))
7             #foreach ($v in $p.values)
8                 #if ($count > 0 )
9                 #end
10                <li> $v.render("leaf", $request)</li>
11                #set($count = $count+1)
12            #end
13      #end
14  #end
15  </ul>
16  </td> </tr>
```

Baris 1 berupa *statement* yang maksudnya adalah menambahkan satu *row* pada tabel jika *resource* "Kegiatan" mempunyai *property* "beranggotakan". Baris 2, 3, 4, 15 dan 16 adalah kode HTML untuk membuat baris dan kolom baru, dengan *header* baris bertuliskan "Anggota". Baris 6–14 merupakan proses *recursive* untuk mendapatkan *value* dari *resource* yang dihubungkan oleh *property* "beranggotakan"

lalu membuat *link* menuju halaman *resource* tersebut. Gambar 3.17 merupakan contoh tampilan dari salah satu *instance* Kegiatan, yaitu "Keprotokolan Pemkot Depok".

Tabel 3.10 dan 3.11 merupakan ringkasan *property* yang ditampilkan di tiap *template page* yang dibuat. *Datatype property* berisi data statis yang diambil dari data RDF, sedangkan *object property* biasanya merupakan data dinamis yang dapat menuju *page* lain atau didapatkan dari proses *inference.* Tampilan masing-masing *page* ini dapat dilihat pada Lampiran D – Tampilan Portal.

Gambar 3.17 Tampilan Halaman Kegiatan

Tabel 3.10 Daftar *Template* dan *Property*

No	*Template File*	Kelas	*Property* yang ditampilkan
1.	pageHumasProtokol.vm	HumasProtokol	*Datatype Property* Nama Bagian

Tabel 3.11 Daftar *Template* dan *Property* (Lanjutan)

No	*Template File*	Kelas	*Property* yang ditampilkan
1.	pageHumasProtokol.vm	HumasProtokol	*Object Property* Kepala (hup:dipimpin) Daftar Kegiatan (hup:memiliki) Tugas (hup:memiliki_tug) Fungsi (hup:memiliki_fung) Daftar Pegawai (hup:mempekerjakan)
2.	pagePelaksana.vm	Pegawai	*Datatype Property* Nama Pegawai, Jenis Pegawai, Agama, Golongan, Jenis Kelamin, NIP, Almamater, Tahun Lulus, Tingkat Ijazah, Tempat Lahir, Tanggal Lahir, Catatan Mutasi
			Object Property Afiliasi (hup:bekerja_di_bag) Sub Bagian yang dikepalai (hup:memimpin) Kegiatan yang dipimpin (hup:memimpin_keg) Kegiatan (hup:anggota_keg)
3.	pageKegiatan.vm	Kegiatan	*Datatype Property* Deskripsi, Tipe, Jenis Kegiatan, Tempat
			Object Property Afiliasi (hup:dimiliki_oleh) Pimpinan (hup:dipimpin_keg) Anggota (hup:beranggotakan) Menggunakan Infrastruktur (hup:menggunakan_keg)
4.	pageInfrastruktur.vm	Infrastruktur	*Datatype Property* Nama Barang, Jumlah
			Object Property Kegiatan (hup:digunakan_di) Penanggung Jawab (hup:digunakan_oleh)
5.	pagePeranan.vm	Peranan	*Datatype Property* Fungsi, Tugas, Deskripsi
			Object Property Afiliasi-Fungsi (hup:dimiliki_fung) Afiliasi-Tugas (hup:dimiliki_tug)

3.9 Tambahan Fitur Portal

Portal dikembangkan dengan memasukkan tiga fitur tambahan utama, yaitu:

- *Add* Data (penambahan data).
- *Update* Data (pengubahan data).
- *Delete* Data (penghapusan data).

Data yang dapat ditambahkan, diubah, serta dihapus tersebut terbagi menjadi tiga jenis data, yaitu:

- Data pegawai.
- Data kegiatan.
- Data infrastruktur.

Dengan demikian, terdapat tiga menu tambahan, dengan masing-masing menu tambahan memiliki tiga sub menu. Seluruh fungsi tambahan dibuat dengan menggunakan *Java Server Pages* (JSP). Selain itu, dalam sebagian fungsi tambahan juga ditanamkan *JavaScript* didalamnya. Ide dasar dari ketiga fitur tambahan adalah menggunakan *input form* yang dihubungkan dengan beberapa *file* N3 pada portal. Berikut ini perubahan yang dapat terjadi pada proses penambahan, pengubahan, dan penghapusan data:

- Setiap penambahan data akan berdampak terhadap bertambahnya satu blok data (dilakukannya *append*) pada *file* N3 yang bersangkutan. Penambahan data tersebut dapat juga berdampak pada berubahnya atribut dari blok pada *file* N3 lainnya yang berkaitan dengan data tersebut.

  Sebagai contoh, penambahan data pegawai akan berpengaruh pada bertambahnya satu blok data mengenai pegawai tersebut pada *file* "pegawai.n3". *File* N3 lainnya yang dapat mengalami perubahan atribut data adalah "humasprotokol.n3", "infrastruktur.n3" dan "kegiatan.n3".

  Berikut ini contoh blok data pegawai yang dapat ditambahkan.

```
hup:Akhmad_Mubarok
      a       hup:KepalaSubBagian ;
      hup:agama "Islam" ;
      hup:almamater "14" ;
      hup:bekerja_di_bag hup:PRO ;
      hup:golongan "III/d" ;
      hup:jenis_kelamin "L" ;
      hup:nama "Akhmad Mubarok" ;
      hup:nip "120 500 096" ;
      hup:tahun_lulus "2005" ;
      hup:tanggal_lahir "25/05/1987" ;
      hup:tempat_lahir "Jakarta" ;
      hup:tingkat_ijazah "SLTA" .
```

- Setiap pengubahan data akan berdampak terhadap berubahnya atribut yang mengalami perubahan pada *file* N3 yang bersangkutan. Pengubahan data tersebut dapat juga berdampak pada berubahnya atribut dari blok pada *file* N3 lainnya yang berkaitan dengan data tersebut.

  Sebagai contoh, pengubahan data kegiatan akan berpengaruh pada berubahnya blok data kegiatan terkait pada *file* "kegiatan.n3". *File* N3 lainnya yang dapat mengalami perubahan atribut data adalah "infrastruktur.n3".

  Berikut ini contoh blok data kegiatan yang dapat diubah.

```
hup:Pengelolaan_Media_Internet
      a       hup:KegiatanInternal ;
      hup:beranggotakan hup:Akhmad_Mubarok ;
      hup:deskripsi "Pengelolaan Media Internet" ;
      hup:dimiliki_oleh hup:PRO ;
      hup:dipimpin_keg hup:Akhmad_Mubarok ;
      hup:jenis_kegiatan "bulanan" ;
      hup:menggunakan_keg hup:Komputer ;
      hup:tempat "Kantor Protokol" .
```

- Setiap penghapusan data akan berdampak terhadap dihapusnya blok terkait pada *file* N3 yang bersangkutan. Penghapusan data tersebut dapat juga berdampak pada berubahnya atribut dari blok pada *file* N3 lainnya yang berkaitan dengan data tersebut.

  Sebagai contoh, penghapusan data infrastruktur akan berpengaruh pada dihapusnya blok data infrastruktur terkait pada *file* "infrastruktur.n3". *File* N3

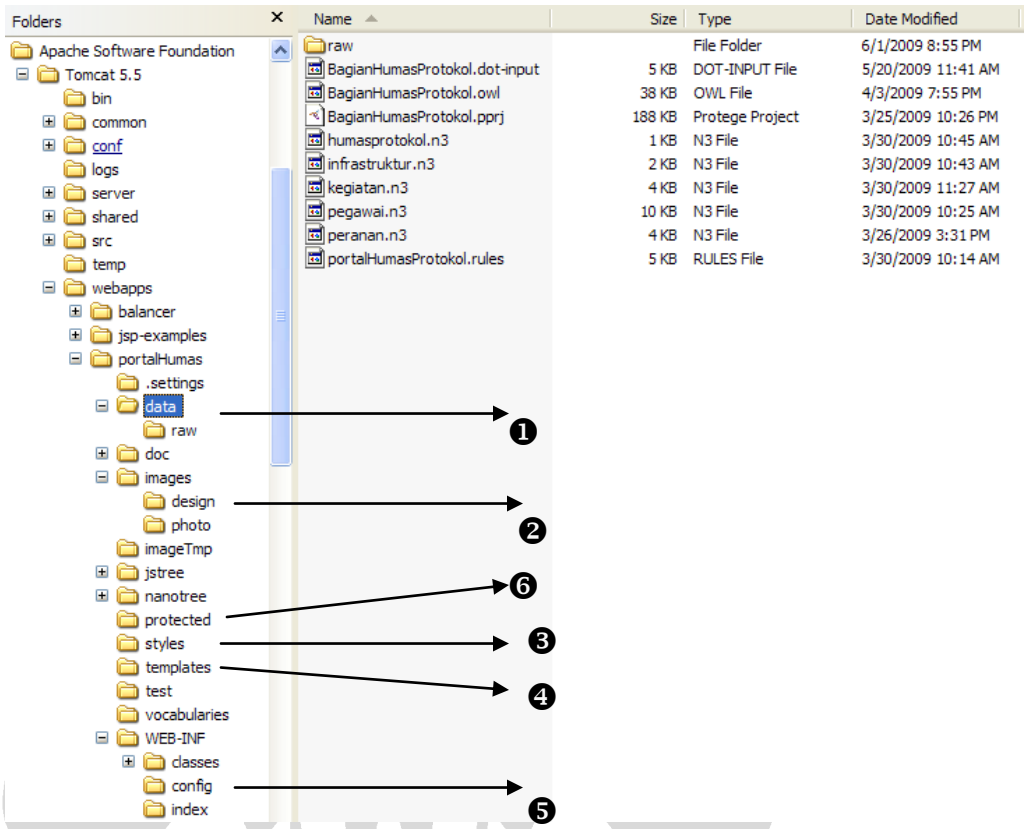
lainnya yang dapat mengalami perubahan atribut data adalah “humasprotokol.n3”, “infrastruktur.n3” dan “kegiatan.n3”.

Berikut ini contoh blok data infrastruktur yang dapat dihapus.

```
hup:Komputer
      hup:digunakan_di hup:Pengelolaan_Media_Internet ;
      hup:digunakan_oleh hup:Akhmad_Mubarok ;
      hup:jumlah "99" ;
      hup:nama_barang "Komputer" .
```

3.10 Struktur Direktori

Bagian ini akan memperlihatkan struktur direktori *portal* dan lokasi berkas yang dimodifikasi dalam penelitian ini. Gambar 3.18 menunjukkan struktur direktori *portal*. Pada penelitian ini, terdapat enam direktori yang berhubungan dengan berkas yang diproses pada setiap tahapan dalam pengembangan *portal*. Enam direktori tersebut yaitu: ❶ \data , ❷ \images , ❸ \styles , ❹ \templates , ❺ \config ❻ \protected, . Direktori ❶ merupakan lokasi berkas yang dikerjakan pada tahapan input *portal*, yaitu berkas ontologi (BagianHumasProtokol.owl), lima berkas *instances data* (*.n3), dan berkas *rules* (portalHumasProtokol.rules). Berkas .csv dan *map graph* yang digunakan dalam proses *generate* data RDF disimpan pada \data\raw. Pada tahap konfigurasi dilakukan modifikasi berkas sources.n3 yang berada di direktori ❺. Bagian yang berhubungan dengan tampilan berada di direktori ❷, ❸ dan ❹ yang berisi berkas *image*, *site*.css, dan *template* (*.vm). Pada direktori ❻ ditambahkan berkas .jsp untuk fitur *add* data, *update* data, dan *delete* data.



Gambar 3.18 Struktur Direktori

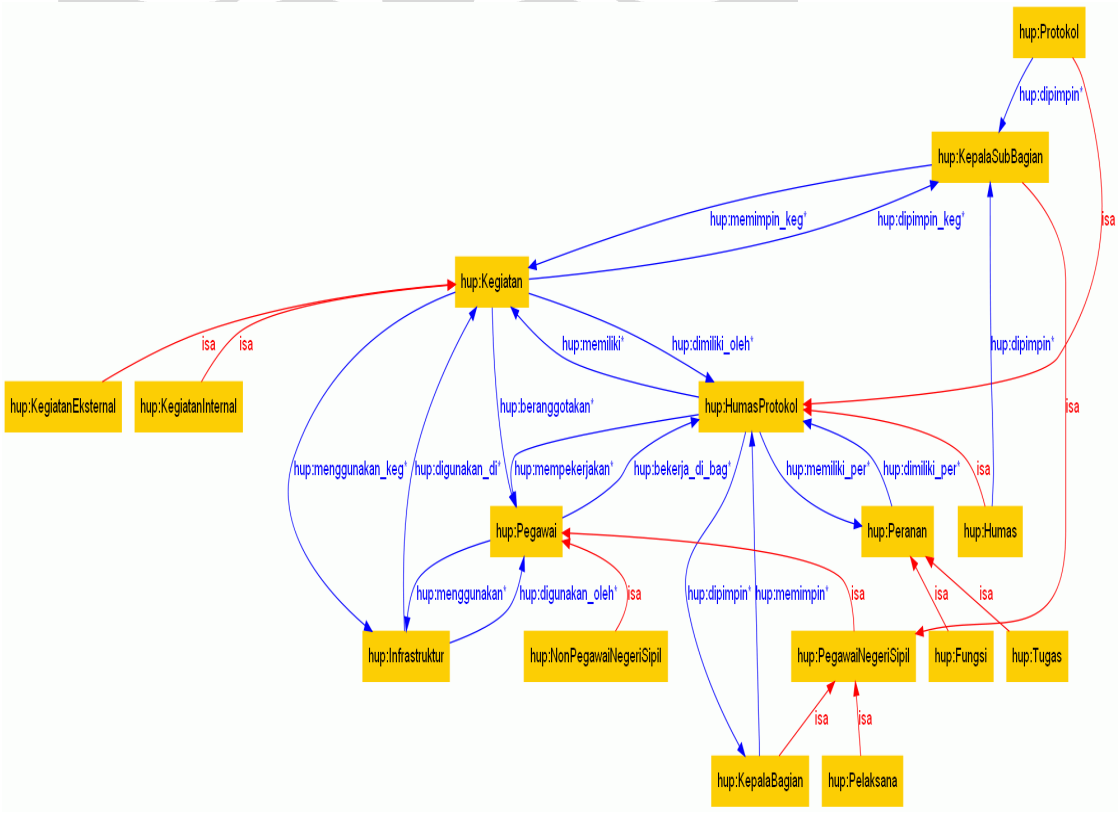
BAB 4

HASIL DAN PEMBAHASAN

Bab ini berisikan tentang penjelasan mengenai portal yang dihasilkan. Hal-hal yang dibahas dalam bab ini mencakup penjelasan mengenai ontologi, fungsionalitas portal, contoh skenario penggunaan portal, dan evaluasi sistem.

4.1 Ontologi HUP

Ontologi HUP memiliki lima kelas utama, yaitu HumasProtokol, Pegawai, Kegiatan, Infrastruktur, dan Peranan, 11 subkelas, 9 *object properties*, serta 18 *datatype properties.* Rincian mengenai kelas, *object properties*, dan *datatype properties* telah dibahas pada Subbab 3.5. Gambar 4.1 merupakan konsep dari ontologi HUP.



Gambar 4.1 Ontologi Humas Protokol

4.2 *Semantic Portal*

Proyek ini menghasilkan prototipe *semantic portal* dengan menggunakan ontologi HUP yang telah dijelaskan sebelumnya. Pada bagian berikut, akan dijelaskan fungsionalitas dari *semantic portal* yang dihasilkan, contoh skenario penggunaan dari portal tersebut, dan juga evaluasi dari prototipe *semantic portal* tersebut.

4.2.1 Fungsionalitas

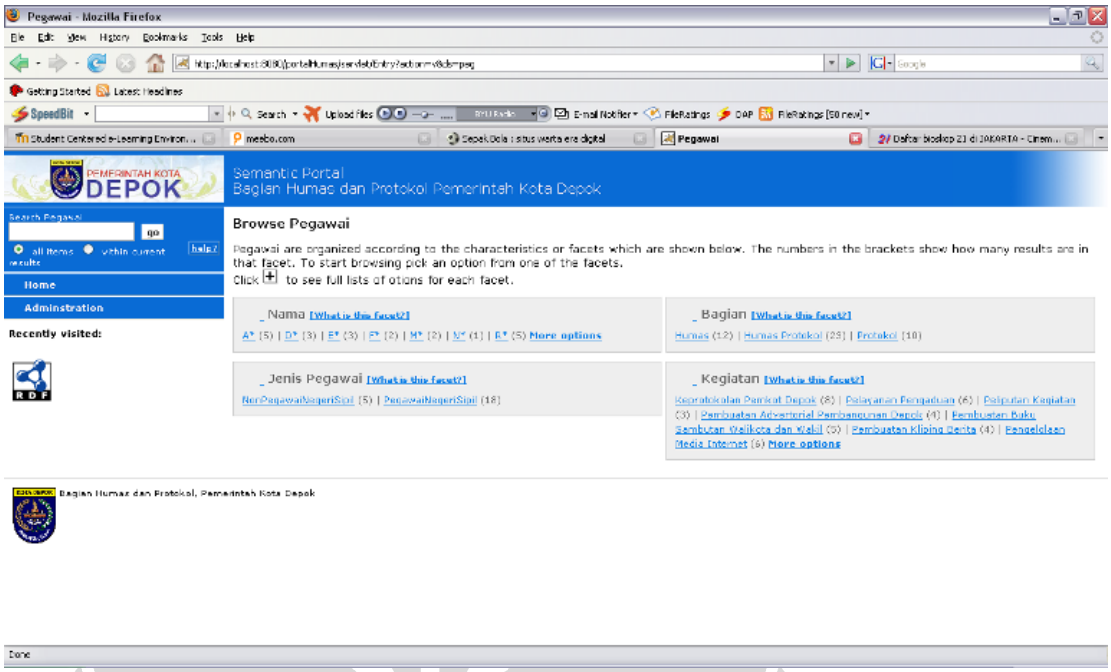
Pada umumnya, fungsi-fungsi yang terdapat pada prototipe yang dihasilkan sudah terdapat dalam portalCore. Meskipun begitu, terdapat tiga penambahan fungsi yang dilakukan dalam proyek ini. Tabel 4.1 menampilkan fungsi-fungsi yang terdapat dalam prototipe yang dikembangkan. Penjelasan dari masing-masing fungsi akan dijelaskan pada bagian berikut.

Tabel 4.1 Fungsi-fungsi Prototipe *Semantic Portal*

***Built-in* portalCore**	**Dikembangkan Pelaksana Proyek**
Faceted Browse	*Add Data*
Text Search	*Update Data*
Refined Search	*Delete Data*
Tree Search	
Visualize Link	

4.2.1.1 *Faceted Browse*

Browse merupakan fungsi utama yang terdapat dalam sebuah portal. Pencarian yang dilakukan dengan menelusuri objek dari berbagai dimensi disebut dengan *faceted browse*. Pada portal yang dikembangkan, terdapat dua objek utama yang dapat di-*browse* melalui *facet*, yaitu Pegawai dan Kegiatan. Untuk objek yang lain dapat ditelusuri melalui fungsi Semua Objek. Pada Gambar 4.2 dapat dilihat contoh dari *faceted browse* untuk objek Pegawai. Pada objek tersebut terdapat empat *facet*, yaitu Nama, Bagian, Jenis Pegawai, dan Kegiatan.

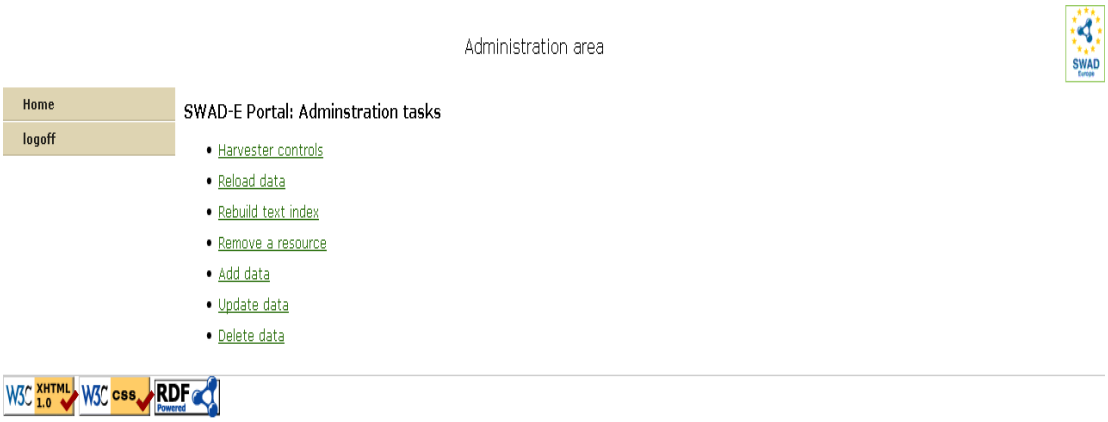


Gambar 4.2 *Browse* Pegawai

4.2.1.2 *Text Search*

Pencarian suatu objek juga dapat dilakukan melalui *text search*. Pengguna dapat melakukan pencarian dengan mengetikkan kata kunci objek yang dicari pada *box text search* yang terdapat pada *sidebar* portal. Fungsi ini akan memberikan hasil yang berbeda-beda jika digunakan pada objek yang berbeda. Misal, ingin dilakukan pencarian pegawai yang memiliki jenis pegawai negeri sipil pada objek pegawai. Pengguna mengetikkan "PegawaiNegeriSipil" pada *box text search*. Hasil pencarian berupa daftar pegawai yang merupakan pegawai negeri sipil. Jika pencarian dilakukan pada objek Semua Objek, maka hasil yang diberikan merupakan semua data yang berhubungan dengan objek yang dicari tersebut. Di samping itu, kata kunci yang digunakan dalam *text search* mempengaruhi hasil pencarian. Jika kata kunci yang digunakan merupakan *id* dari objek, maka hasil yang didapatkan menunjukkan adanya keterhubungan antara objek yang dicari dengan data lainnya. Jika tidak, maka hasil pencarian akan menampilkan data-data sesuai dengan kata kunci yang diberikan tanpa adanya keterhubungan antar data (layaknya *text search* biasa).

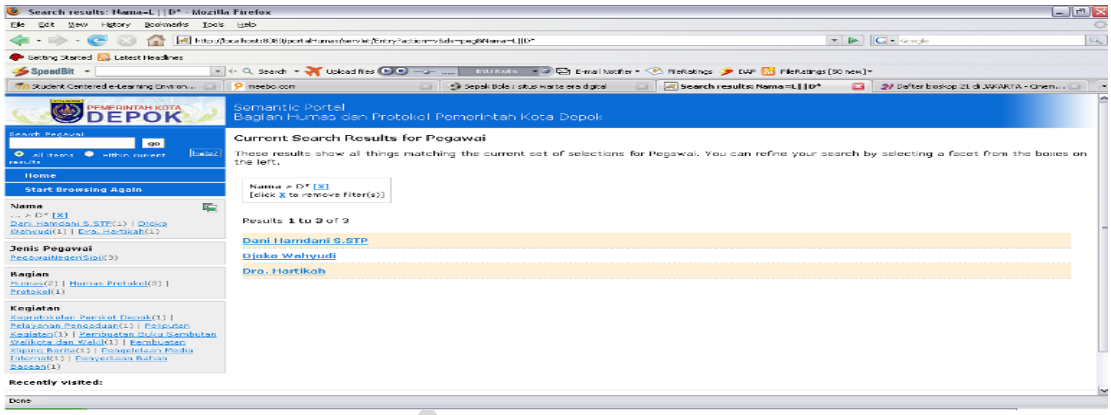
Text search dapat dilakukan jika data-data sudah diindeks di dalam portal. Untuk dapat memberikan indeks pada data-data tersebut dilakukan dengan cara mengklik menu "*Rebuild Text Index*" yang terdapat pada menu *administration*. Jika data-data belum diindeks, pencarian dengan *text search* tidak akan menghasilkan apapun. Gambar 4.3 merupakan tampilan pilihan menu *administration*.



Gambar 4.3 Menu *Administration*

4.2.1.3 *Refined Search*

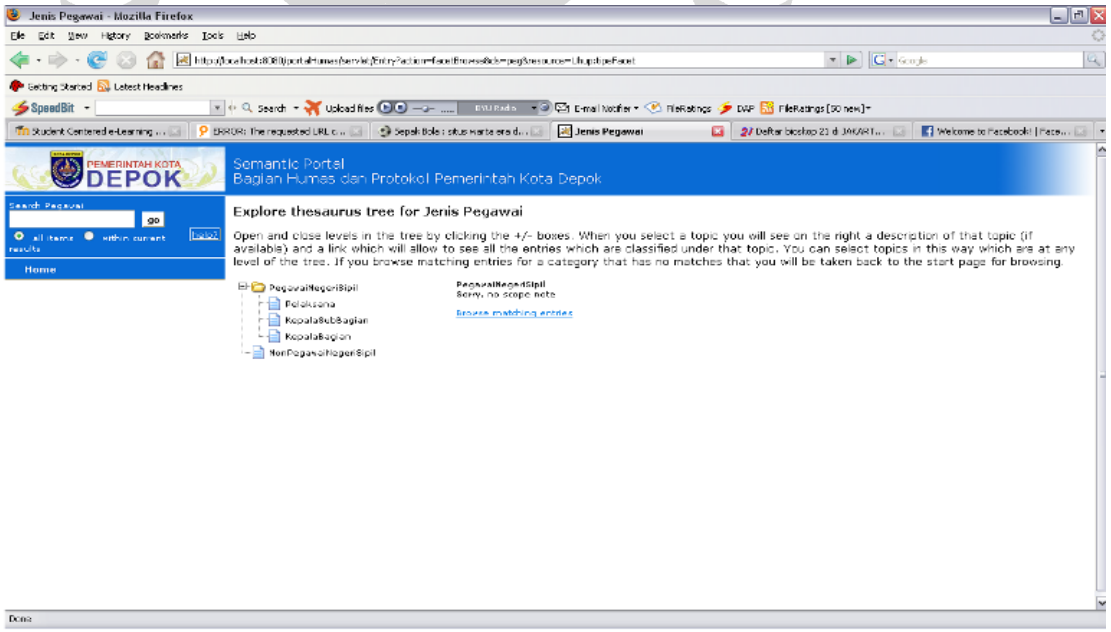
Fungsi lain dari prototipe yang dikembangkan adalah *refined search*. Fungsi ini berguna untuk menambah *filter* pencarian yang sesuai sehingga pencarian dapat dilakukan dengan lebih cepat. Misal, ingin dilakukan pencarian pegawai yang terlibat dalam kegiatan Pelayanan Pengaduan. Terdapat tiga nama yang diberikan setelah mengklik huruf 'D' pada *facet* Nama. Selanjutnya, untuk mengetahui siapa pegawai yang terlibat dalam kegiatan tersebut, dapat dilakukan dengan cara mengklik kegiatan Pelayanan Pengaduan pada *filter* Kegiatan. Filter pencarian tersebut terdapat pada *sidebar* portal seperti yang terlihat pada Gambar 4.4.



Gambar 4.4 *Search Results*

4.2.1.4 *Tree Search*

Pencarian lain yang dapat dilakukan adalah pencarian melalui *tree* yang menampilkan hierarki objek yang dicari. *Tree* ini diperoleh dari *facet* yang memiliki tipe *hierarchical*. Pencarian dilakukan dengan mengklik tulisan [*What is this facet*?] yang terdapat pada *facet box*. Setelah itu, hierarki yang muncul dapat diklik dan pencarian dapat dilakukan dengan mengklik *link* "*Browse Matching Entries*". Gambar 4.5 merupakan contoh dari *tree search* yang terdapat dalam prototipe portal yang dihasilkan.



Gambar 4.5 *Tree Search*

4.2.1.5 *Visualize Links*

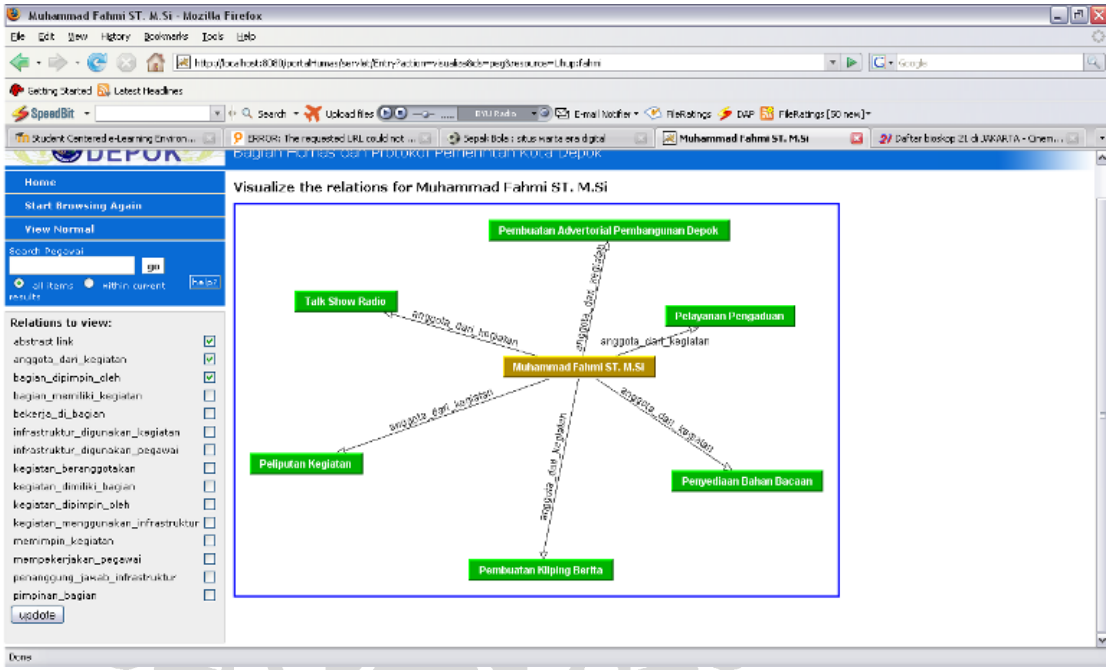
Fungsi lain yang terdapat dalam prototipe portal yang dihasilkan adalah *visualize link*s. Fungsi ini akan menampilkan hubungan antar data dalam bentuk graf. Pada portal ini terdapat empat belas *link* yang dapat divisualisasikan seperti yang terdapat pada Tabel 4.2 dan 4.3. Visualisasi hubungan antar data dapat dilakukan dengan mengklik menu "*Visualize Link*s" pada menu *sidebar* setelah pencarian dilakukan. Setelah itu, pengguna dapat memilih relasi apa saja yang ingin ditampilkan yang terdapat pada bagian "*Relations to view*". Untuk mendapatkan hasil visualisasi, pengguna kemudian mengklik "*update*" pada bagian tersebut. Sebagai contoh hasil visualisasi dapat dilihat pada Gambar 4.6.

Tabel 4.2 Relasi yang Dapat Divisualisasikan

Relasi	Penjelasan Singkat
anggota_dari_kegiatan	Menghubungkan antara pegawai dan kegiatan
bagian_dipimpin_oleh	Menghubungkan antara bagian dan pegawai
bagian_memiliki_kegiatan	Menghubungkan antara bagian dan kegiatan
bekerja_di_bagian	Menghubungkan antara pegawai dan bagian
infrastruktur_digunakan_kegiatan	Menghubungkan antara infrastruktur dan kegiatan
infrastruktur_digunakan_pegawai	Menghubungkan antara infrastruktur dan pegawai
kegiatan_beranggotakan	Menghubungkan antara kegiatan dan pegawai
kegiatan_dimiliki_bagian	Menghubungkan antara kegiatan dan bagian
kegiatan_dipimpin_oleh	Menghubungkan antara kegiatan dan pegawai
kegiatan_menggunakan_infrastruktur	Menghubungkan antara kegiatan dan infrastruktur
memimpin_kegiatan	Menghubungkan antara pegawai dan kegiatan
mempekerjakan_pegawai	Menghubungkan antara bagian dan pegawai

Tabel 4.3 Relasi yang Dapat Divisualisasikan (Lanjutan)

Relasi	Penjelasan Singkat
penanggung_jawab_infrastruktur	Menghubungkan antara pegawai dan infrastruktur
pimpinan_bagian	Menghubungkan antara pegawai dan bagian

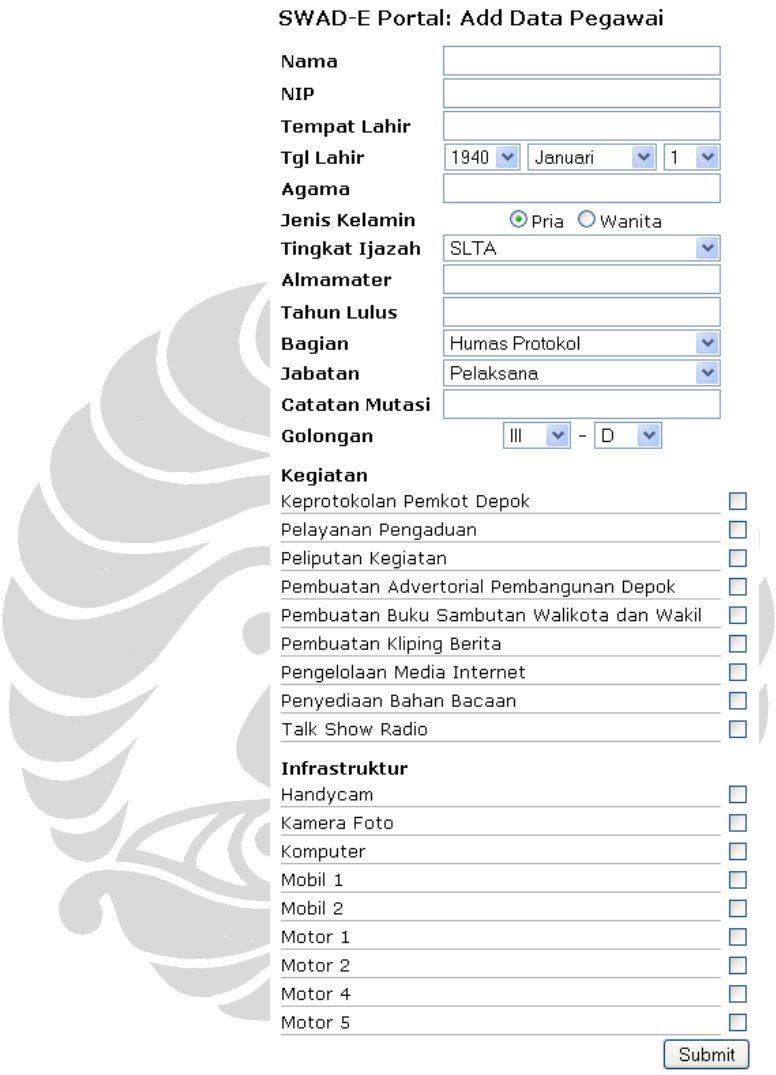


Gambar 4.6 *Visualize Links*

4.2.1.6 *Add Data*

Pada fungsi *add data* terdapat 3 jenis data yang dapat ditambahkan, yaitu data pegawai, data kegiatan, dan data infrastruktur. Untuk menggunakan fungsi ini, pengguna harus *login* terlebih dahulu sebagai *administrator*. Setelah itu, pengguna memilih pilihan *add data*, lalu memilih data apa yang akan ditambahkan. Setelah itu, akan muncul *form* yang harus diisi oleh pengguna dengan data yang akan ditambahkan. Setelah *form* diisi dengan benar, data akan secara otomatis ditambahkan ke dalam sistem dan akan muncul tampilan yang menunjukkan bahwa data sudah berhasil ditambahkan. Adapun jika pengguna tidak mengisi *form* dengan

benar, sistem akan menampilkan pesan bahwa proses penambahan data gagal dilakukan, lalu pengguna diminta untuk mengisi ulang *form* tersebut dengan benar. Gambar 4.7 menunjukkan *form* untuk menambah data pegawai.



Gambar 4.7 *Add* Pegawai

4.2.1.7 *Update Data*

Seperti halnya pada fungsi *add data*, pada fungsi *update data* ini juga terdapat 3 jenis data yang dapat di-*update*, yaitu data pegawai, data kegiatan, dan data infrastruktur. Untuk menggunakan fungsi ini, pengguna harus *login* terlebih dahulu sebagai *administrator*. Setelah itu, pengguna memilih pilihan *update* data, lalu

memilih data apa yang akan di-*update*. Setelah itu, sistem akan menampilkan daftar data yang dapat di-*update*, lalu pengguna diminta untuk memilih salah satu dari data tersebut yang ingin di-*update*. Setelah memilih data yang akan di-*update*, akan muncul *form* yang sudah berisi data yang dapat diubah. Setelah *form* diisi dengan benar, data akan secara otomatis ter-*update* ke dalam sistem dan akan muncul tampilan yang menunjukkan bahwa data sudah berhasil di-*update*. Adapun jika data yang diisi ke dalam *form* tidak benar, sistem akan menampilkan pesan bahwa proses *update* data gagal dilakukan, lalu pengguna diminta untuk mengisi ulang form tersebut dengan benar. Gambar 4.8 menunjukkan contoh *form update* pegawai.

Gambar 4.8 *Update* Pegawai

4.2.1.8 *Delete Data*

Fitur "*Delete Data*" mencakup tiga jenis data yang dapat dihapus, yaitu data pegawai, data kegiatan, dan data infrastruktur. Menurut alur fitur ini, pengguna perlu mengikuti urutan tindakan sebagai berikut:

a) *Login* melalui menu *administration* (sebagai *administrator*).
   Sistem akan menampilkan pilihan-pilihan "*Administration tasks*" yang dapat dilakukan.

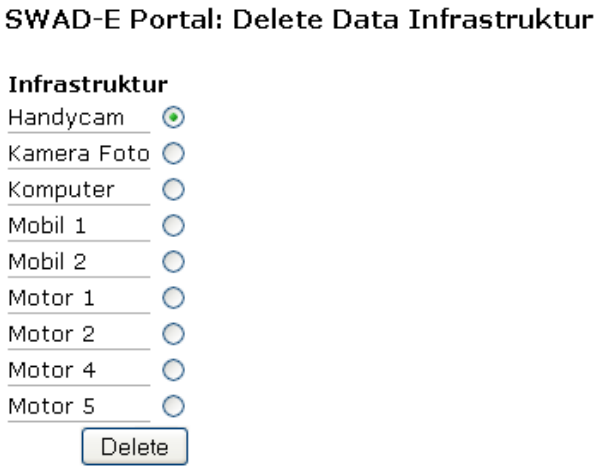
b) Memilih "*Delete data*".
   Sistem akan menampilkan pilihan jenis data yang dapat dihapus, yaitu "*Delete* Pegawai", "*Delete* Kegiatan", atau "*Delete* Infrastruktur".

c) Memilih jenis data yang sesuai.
   Sistem akan menampilkan daftar data yang sesuai dengan jenisnya.

d) Memilih data yang akan dihapus melalui tombol bulat di sebelah kanan data tersebut.

e) Mengklik tombol "*Delete*".
   Sistem akan menampilkan pesan berikut: "Proses penghapusan data berhasil dilakukan". Selanjutnya, di bawah pesan tersebut terdapat tombol "*Continue*", yang apabila ditekan akan mengarahkan pengguna pada halaman "*Browse* Semua Objek".

Setelah penghapusan data dilakukan, maka sistem akan menampilkan data terbaru yang sesuai dengan perubahan yang terjadi pada proses tadi. Gambar 4.9 merupakan contoh tampilan dari fitur "*Delete* Data".



Gambar 4.9 *Delete* Infrastruktur

4.2.2 Contoh Skenario Penggunaan

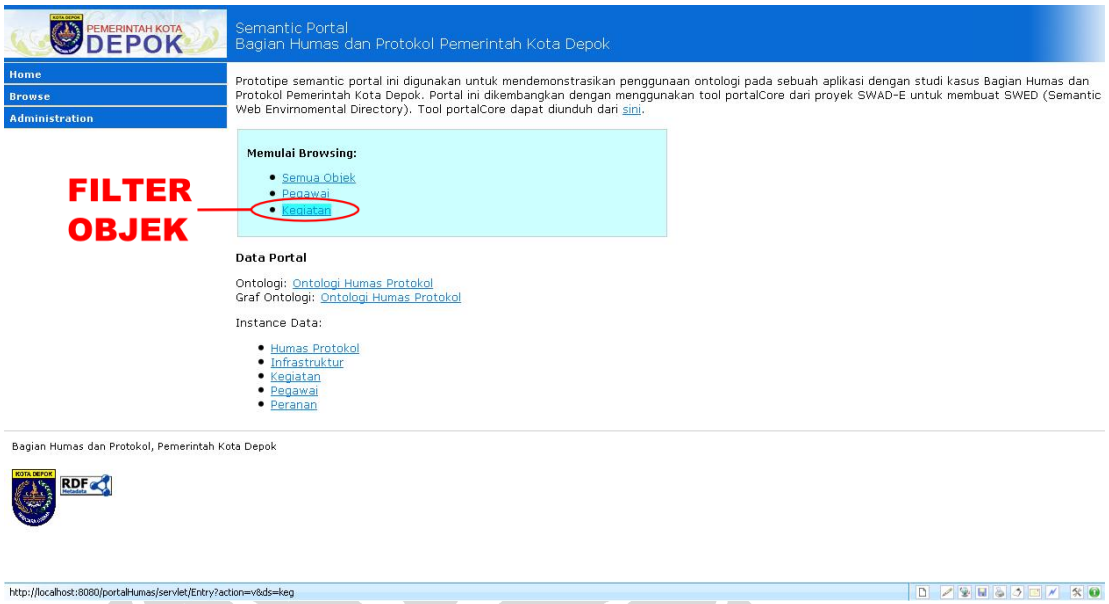
Pada bagian ini akan dijelaskan beberapa contoh skenario penggunaan dari portal yang dihasilkan. Melalui portal tersebut, diharapkan pencarian yang dilakukan menjadi lebih cepat dan akurat.

Skenario 1

Pengguna : Pegawai Pemerintah Kota Depok

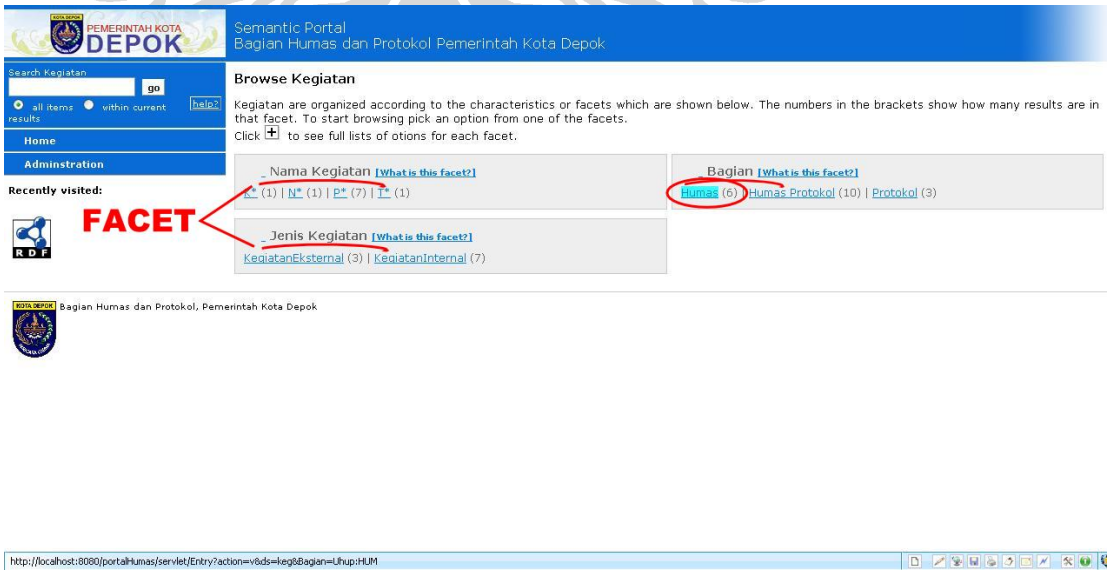
Tujuan : Mencari Penanggung Jawab Kegiatan

Pemerintah Kota Depok memiliki kegiatan yang cukup banyak jumlahnya. Untuk mempermudah pelaksanaan kegiatan-kegiatan tersebut, maka setiap kegiatan akan memiliki seorang penanggung jawab. Penanggung jawab kegiatan ini bertanggung jawab penuh terhadap pelaksanaan kegiatan secara menyeluruh. Jika terjadi sesuatu yang berkaitan dengan suatu kegiatan, maka penanggung jawab inilah yang akan dimintai pertanggungjawabannya. Berikut contoh skenario pencarian yang dilakukan dengan menggunakan portal yang dihasilkan.



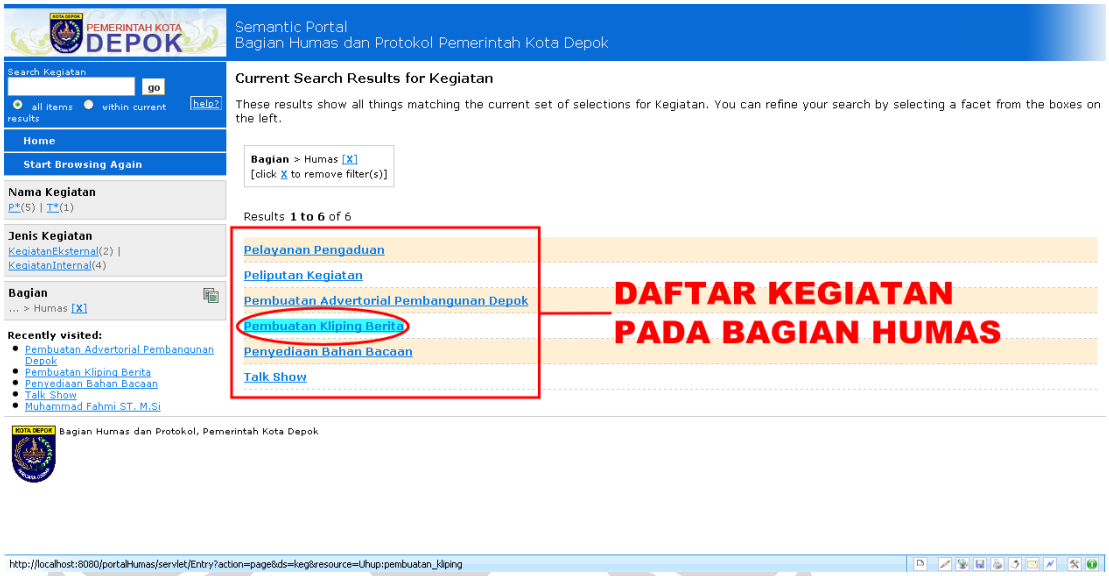
Gambar 4.10 Skenario 1-1

1. Pada halaman depan, pencarian dimulai dengan mengklik *link* "Kegiatan" sebagai *filter* objeknya.
2. Terdapat tiga *facet* yang dapat digunakan untuk mencari suatu kegiatan, yaitu *facet* Nama Kegiatan, Bagian, dan Jenis Kegiatan. Misal, *facet* yang dipilih adalah *facet* Bagian.



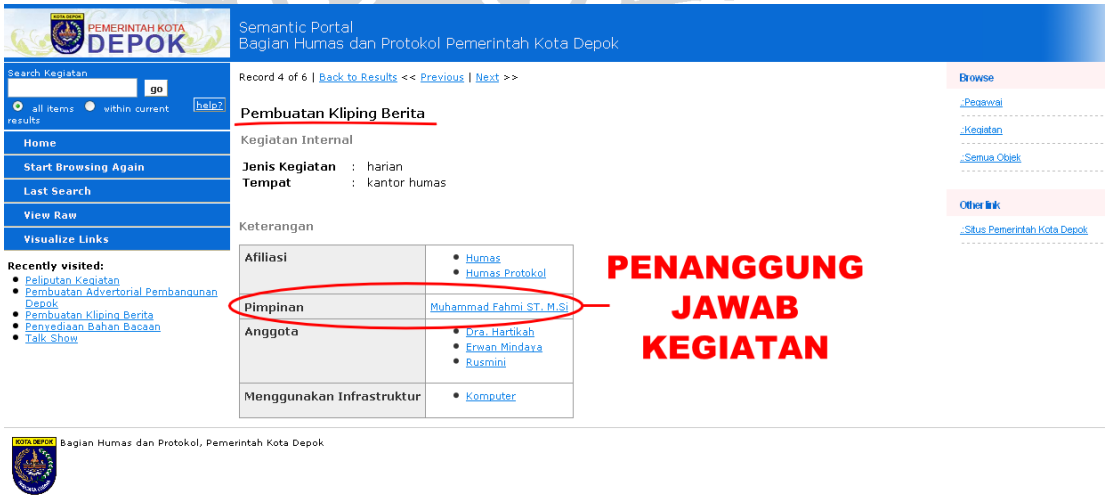
Gambar 4.11 Skenario 1-2 dan 1-3

3. Mengklik salah satu pilihan yang ada pada *facet* Bagian. Misal, mengklik "Humas".



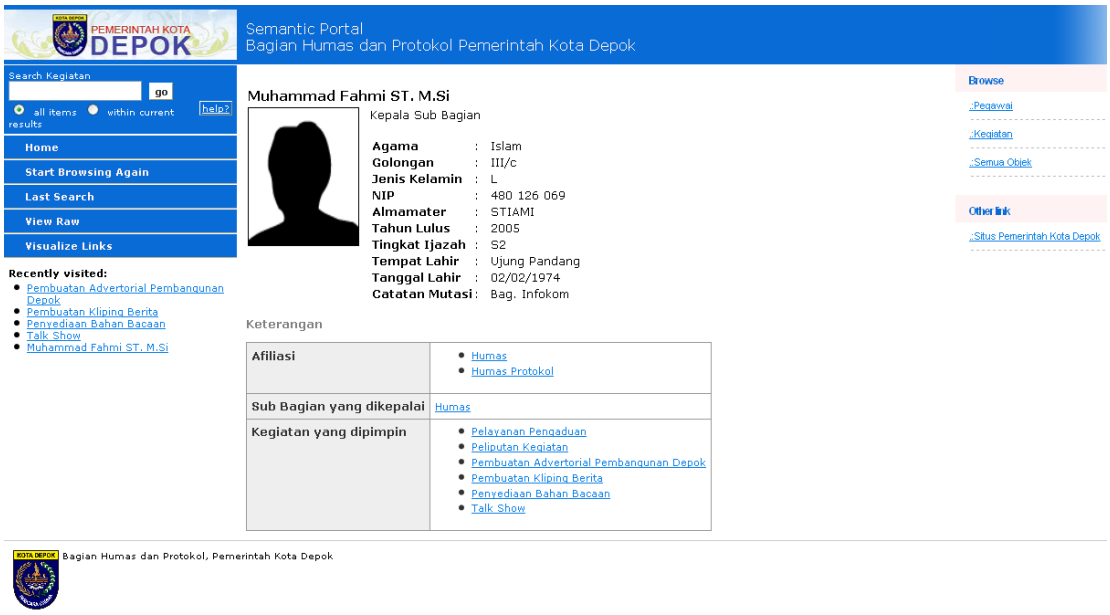
Gambar 4.12 Skenario 1-4

4. Setelah mengklik pilihan tersebut akan muncul daftar kegiatan yang dimiliki bagian tersebut. Pada contoh yang diberikan akan muncul daftar kegiatan yang dimiliki bagian humas.



Gambar 4.13 Skenario 1-5

5. Pilih kegiatan yang dicari. Kemudian akan muncul data-data tentang kegiatan tersebut termasuk data penanggung jawab kegiatan tersebut.



Gambar 4.14 Skenario 1-6

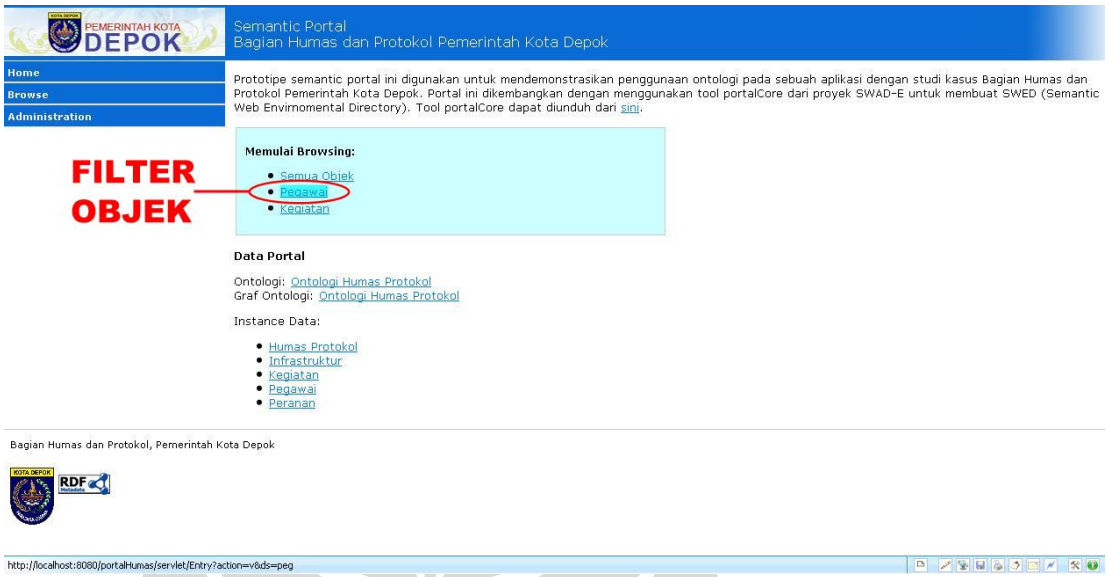
6. Untuk mengetahui data penanggung jawab kegiatan dapat dilakukan dengan mengklik nama penanggung jawab tersebut.

Skenario 2

Pengguna : Kepala Subbagian Humas

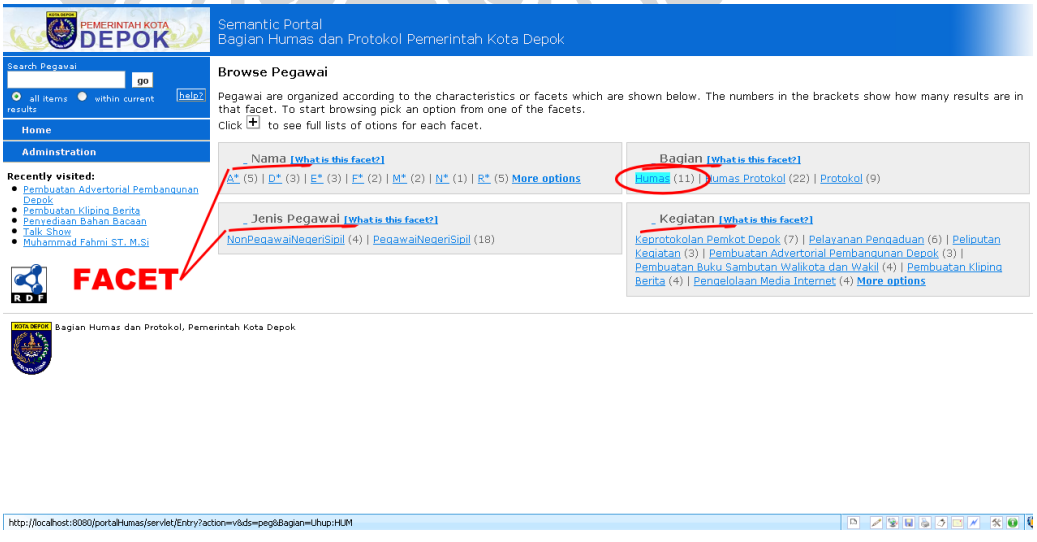
Tujuan : Mencari daftar pegawai pada bagian Humas

Bagian Humas dan Protokol Pemerintah Kota Depok memiliki jumlah pegawai yang cukup banyak. Data-data kepegawaian harus tersimpan dengan tepat sehingga jika diperlukan dapat diperoleh dengan cepat. Seorang kepala subbagian bertanggung jawab penuh terhadap seluruh pegawai dan harus memiliki data-data pegawai yang ada pada subbagian tersebut. Berikut ini contoh skenario pencarian yang dapat dilakukan.



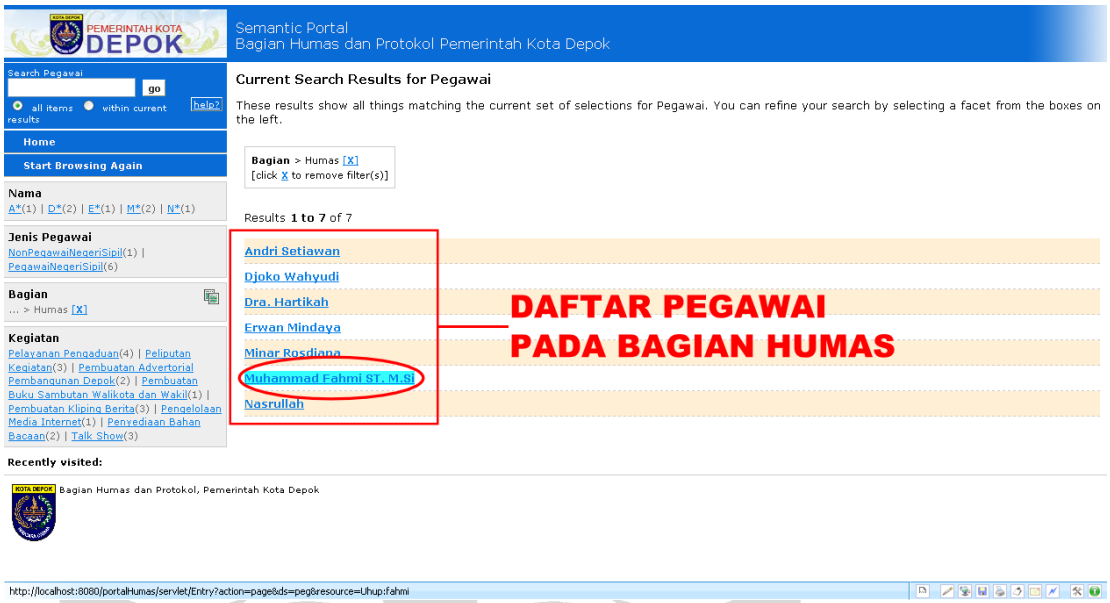
Gambar 4.15 Skenario 2-1

1. Pada halaman depan, pencarian dilakukan dengan mengklik *link* "Pegawai" sebagai *filter* objek.
2. Terdapat empat *facet* yang dapat digunakan untuk mencari data-data pegawai yang dimiliki oleh subbagian humas, yaitu *facet* Nama, Bagian, Jenis Pegawai, dan Kegiatan. Untuk mempermudah, kepala subbagian sebaiknya memilih *facet* Bagian.



Gambar 4.16 Skenario 2-2 dan 2-3

3. Pada *facet* tersebut, terdapat pilihan Humas, Humas Protokol, dan Protokol. Kepala subbagian kemudian mengklik pilihan Humas.



Gambar 4.17 Skenario 2-4

4. Setelah mengklik pilihan tersebut akan muncul daftar pegawai yang dimiliki oleh subbagian humas.

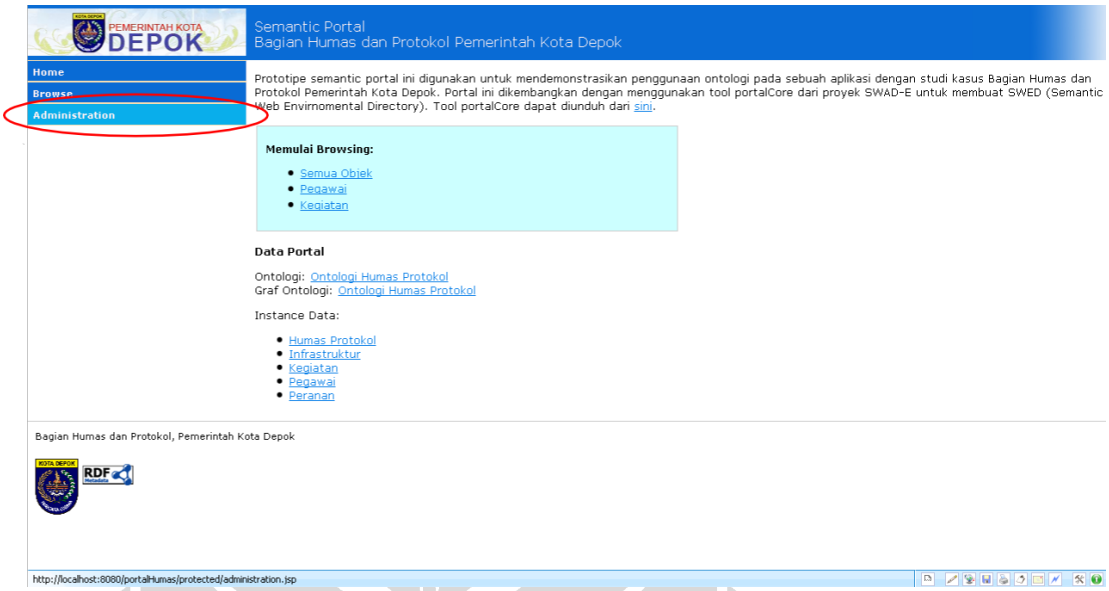
5. Untuk mengetahui rincian data masing-masing pegawai dapat dilakukan dengan mengklik nama-nama yang ada pada daftar tersebut.

Skenario 3

Pengguna : *Administrator* portal

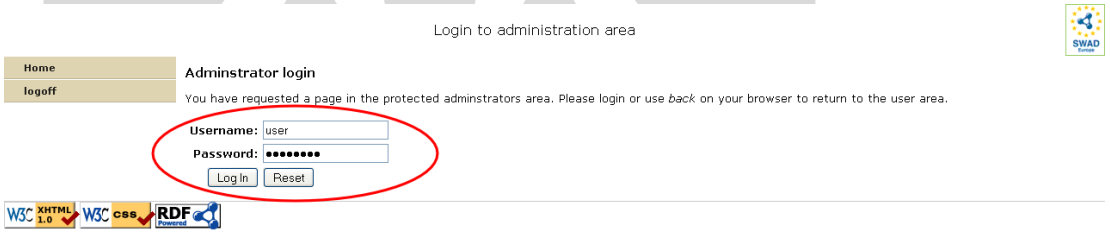
Tujuan : Melakukan penambahan data pegawai

Data-data yang ada dalam portal dapat bertambah seiring berjalannya waktu. Penambahan data ini hanya dapat dilakukan oleh *administrator* portal. Berikut skenario penambahan data tersebut.



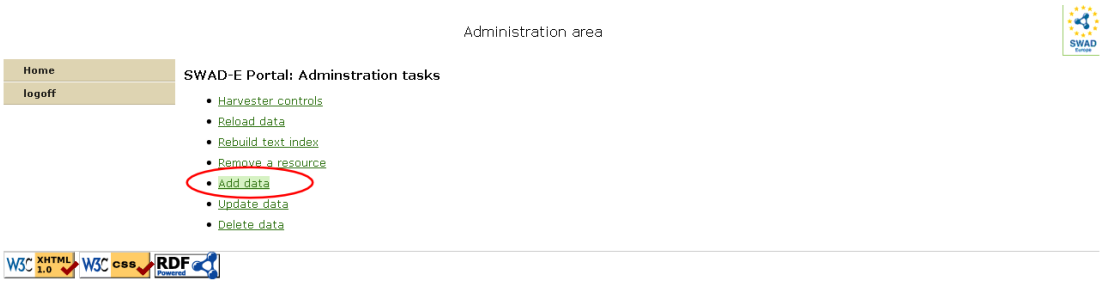
Gambar 4.18 Skenario 3-1-a

1. Pada halaman depan, pengguna memilih menu *administration* dan melakukan *login* dengan *username* dan *password* yang dimiliki.

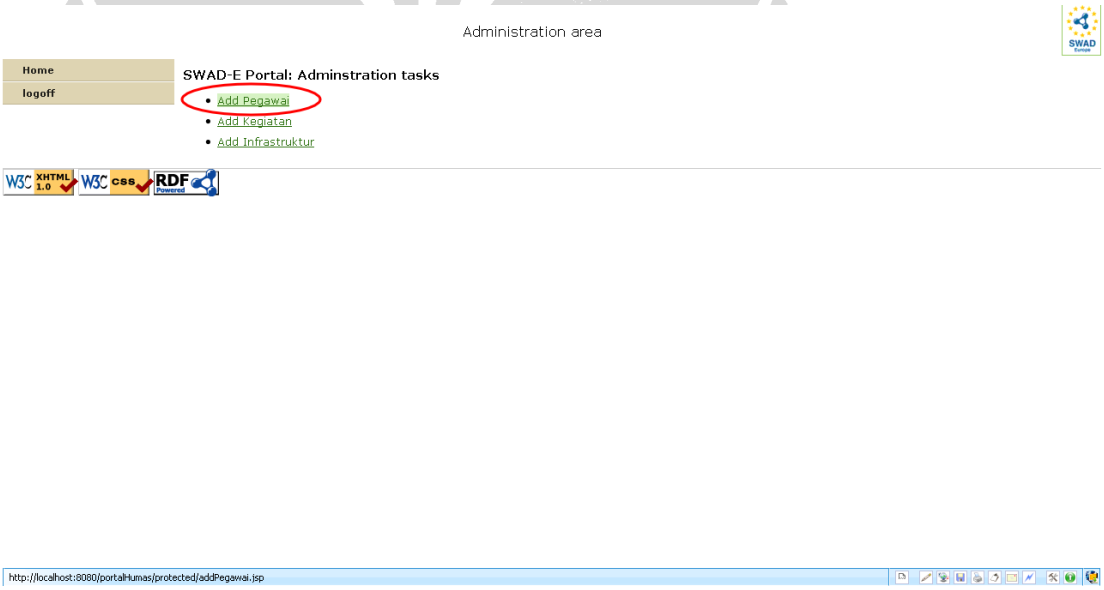


Gambar 4.19 Skenario 3-1-b

2. Setelah itu, memilih pilihan "*Add data*". Setelah mengklik pilihan tersebut, akan muncul tiga pilihan penambahan data, yaitu "*Add* Pegawai", "*Add* Kegiatan", dan "*Add* Infrastruktur".

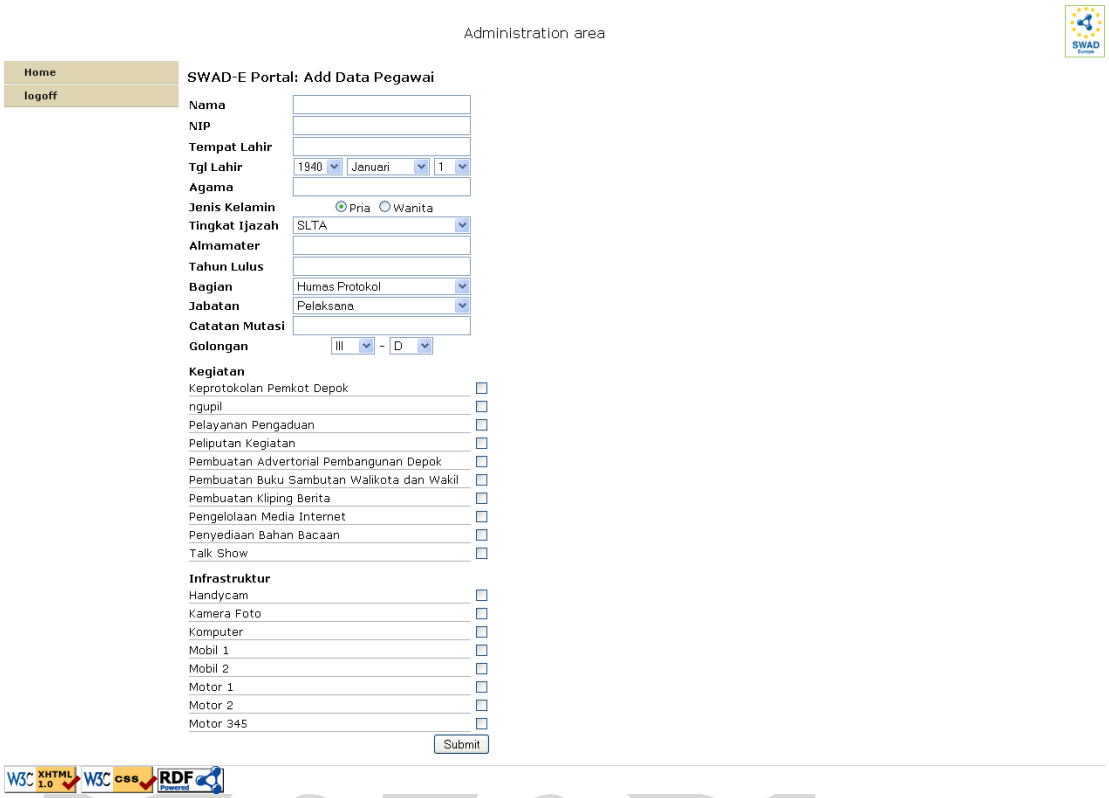


Gambar 4.20 Skenario 3-2-a



Gambar 4.21 Skenario 3-2-b

3. Pilih "*Add* Pegawai". Setelah itu akan muncul *form* penambahan pegawai seperti pada Gambar 4.22. Setelah pengisian *form* kemudian klik tombol "*Submit*". Data dalam portal telah berhasil ditambahkan.



Gambar 4.22 Skenario 3-3

4.2.3 Evaluasi Sistem

Pada bagian ini akan dijelaskan hasil evaluasi dari portal yang dikembangkan. Evaluasi dilakukan oleh pihak ketiga dengan mencoba seluruh fungsi-fungsi yang ada pada portal. Evaluator menyatakan bahwa secara keseluruhan, tujuan umum dari pengembangan sistem ini sudah tercapai. Fitur pencarian yang ada sudah cukup baik dan dapat mempercepat proses pencarian terhadap suatu data. Namun, terdapat juga beberapa kekurangan pada portal yang dikembangkan antara lain:

a) Pada menu "*Visualize Links*", terdapat relasi yang tidak relevan dengan *instance* yang dipilih. Misalnya, dipilih *instance* kegiatan pelayanan pengaduan, maka *link* yang tidak berhubungan dengan kegiatan, seperti misalnya bekerja_di_bagian sebaiknya tidak dapat dipilih.

b) Sistem interaksi untuk menu *add*, *update*, dan *delete data* kurang baik. Jika terjadi kegagalan dalam melakukan penambahan atau pengubahan, peringatan

yang muncul tidak memberitahukan secara spesifik apa yang menyebabkan terjadinya kegagalan.

c) Tidak terdapat pilihan untuk melakukan *multiple delete* sehingga penghapusan data harus dilakukan satu per satu.