TUGAS AKHIR – KI141502

PEMBUATAN APLIKASI METADATA GENERATOR UNTUK KOLEKSI PENINGGALAN WARISAN BUDAYA

WIMBA AGRA WICESA
NRP. 5112 100 102

Dosen Pembimbing 1
Sarwosri, S.Kom., M.T.

Dosen Pembimbing 2
Nurul Fajrin Ariyani, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
2017

TUGAS AKHIR – KI141502

PEMBUATAN APLIKASI METADATA GENERATOR UNTUK KOLEKSI PENINGGALAN WARISAN BUDAYA

WIMBA AGRA WICESA
NRP. 5112 100 102

Dosen Pembimbing 1
Sarwosri, S.Kom., M.T.

Dosen Pembimbing 2
Nurul Fajrin Ariyani, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
2017

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

DEVELOPMENT OF METADA GENERATOR APPLICATION FOR CULTURAL HERITAGE COLLECTION

WIMBA AGRA WICESA
NRP. 5112 100 102

Supervisor 1
Sarwosri, S.Kom., M.T.

Supervisor 2
Nurul Fajrin Ariyani, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya
2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PEMBUATAN APLIKASI METADATA GENERATOR UNTUK KOLEKSI PENINGGALAN WARISAN BUDAYA

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
WIMBA AGRA WICESA
NRP: 5112 100 102

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Sarwosri, S.Kom., M.T. .....
   NIP: 19760809 200112 2 001 (pembimbing 1)

2. Nurul Fajrin Ariyani, S.Kom., M.Sc. .....
   NIP. 19860722 201504 2 003 (pembimbing 2)

SURABAYA
JANUARI, 2017

PEMBUATAN APLIKASI METADATA GENERATOR UNTUK KOLEKSI PENINGGALAN WARISAN BUDAYA

Nama	**: Wimba Agra Wicesa**
NRP	**: 5112100102**
Jurusan	**: Teknik Informatika**
	Fakultas Teknologi Informasi ITS
Dosen Pembimbing I	**: Sarwosri, S.Kom., M.T.**
Dosen Pembimbing II	**: Nurul Fajrin Ariyani, S.Kom., M.Sc.**

ABSTRAK

Warisan budaya merupakan suatu aset penting yang digunakan sebagai sumber informasi dalam mempelajari ilmu sejarah. Mengelola data warisan budaya menjadi suatu hal yang harus diperhatikan guna menjaga keutuhan data warisan budaya di masa depan. Menciptakan sebuah metadata warisan budaya merupakan salah satu langkah yang dapat diambil untuk menjaga nilai dari sebuah artefak. Dengan menggunakan konsep metadata, informasi dari setiap objek warisan budaya tersebut menjadi mudah untuk dibaca, dikelola, maupun dicari kembali meskipun telah tersimpan lama. Selain itu dengan menggunakan konsep metadata, informasi tentang warisan budaya dapat digunakan oleh banyak sistem.

Metadata warisan budaya merupakan metadata yang cukup besar. Sehingga untuk membangun metadata warisan budaya dibutuhkan waktu yang cukup lama. Selain itu kesalahan (human error) juga dapat menghambat proses pembangunan metadata warisan budaya. Proses pembangkitan metadata warisan budaya melalui Aplikasi Metadata Generator menjadi lebih cepat dan mudah karena dilakukan secara otomatis oleh sistem. Aplikasi ini juga dapat menekan human error sehingga proses pembangkitan menjadi lebih efisien.

Kata kunci: CIDOC-CRM, Metadata, Warisan Budaya.

(Halaman ini sengaja dikosongkan)

DEVELOPMENT OF METADA GENERATOR APPLICATION FOR CULTURAL HERITAGE COLLECTION

Name	**: Wimba Agra Wicesa**
NRP	**: 5112100102**
Department	**: Department of Informatics Faculty of Information Technology ITS**
Supervisor I	**: Sarwosri, S.Kom., M.T.**
Supervisor II	**: Nurul Fajrin Ariyani, S.Kom., M.Sc.**

ABSTRACT

Cultural heritage is an important asset that is used as a source of information in studying the history of science. Managing cultural heritage data into a matter that must be considered in order to maintain the data integrity of the cultural heritage in the future. Creates a metadata cultural heritage is one of the steps that can be taken to preserve the value of an artifact. By using the concept of metadata, information from any Object of cultural heritage has become easier to read, manage, and retrievable although it has been stored for long time. In addition, by using the concept of metadata, information about the cultural heritage can be used by many systems.

However, cultural heritage metadata is large enough. So as to build metadata cultural heritage takes quite a long time. Moreover, human error can also inhibit the development process of cultural heritage metadata. Cultural heritage metadata generation process with the Metadata Generator Applications can be completed quickly and easily because it is done automatically by the system. This application can also suppress the human error so that the generation process becomes more efficient.

Keywords: CIDOC-CRM, Cultural Heritage, Metadata.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

Pembuatan Aplikasi Metadata Generator Untuk Koleksi Peninggalan Warisan Budaya

Melalui lembar ini, penulis hanya ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Allah Subhaanahu Wa Ta'ala atas segala nikmat dan rahmat yang telah diberikan kepada hamba-Nya ini.
2. Nabi Muhammad ﷺ atas tuntunan dan bimbingan-Nya.
3. Ayah, Ibu dan keluarga penulis yang tiada henti-hentinya mencurahkan kasih sayang, perhatian dan doa kepada penulis selama ini.
4. Ibu Nurul Fajrin Ariyani selaku dosen pembimbing yang telah memberikan nasihat, arahan, dan bantuan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
5. Ibu Sarwosri selaku dosen pembimbing yang telah memberikan bimbingan, motivasi, nasihat dan meluangkan waktu untuk membantu pengerjaan Tugas Akhir ini.
6. Bapak Imam Kuswardayan selaku dosen wali penulis yang telah memberikan perhatian dan motivasi kepada penulis selama menjadi mahasiswa di lingkungan Teknik Informatika ITS.
7. Bapak dan Ibu dosen Teknik Informatika ITS yang telah membina dan memberikan ilmu kepada penulis selama menempuh studi di Teknik Informatika ITS.
8. Keluarga penulis di Surabaya, Yeyen, Aal, Syaiba, Riski, Ramandha, Adit, Esqy, Kurnia, M. Riski, Deni, Rahandi, Ferdik dan Diki, yang terus mengalirkan dukungan dan selalu bersedia untuk berbagi suka duka.
9. Sahabat-sahabat penulis, M. Rizky Alamsyah dan Wahyu Apria Dharma yang senantiasa memberikan motivasi, semangat serta doa.

10. Teman seperjuangan penulis, Hanafi, Kamali, Madis, Alief, dan Naufal yang selalu bersedia untuk membagikan ide-ide mengenai Tugas Akhir ini.
11. Keluarga TC 2012 yang terus memberikan dukungan dan memberikan semangat selama kuliah.
12. Seluruh teman yang telah membantu, memberikan dukungan kepada penulis untuk menyelesaikan Tugas Akhir ini.
13. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu persatu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Januari 2017

Wimba Agra Wicesa

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Bab ini akan menjelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan, metodologi dan sistematika penulisan yang digunakan dalam pembuatan buku tugas akhir ini.

1.1. Latar Belakang

Warisan budaya merupakan suatu aset penting yang digunakan sebagai sumber informasi dalam mempelajari ilmu sejarah. Mengelola data warisan budaya menjadi suatu hal yang harus diperhatikan guna menjaga keutuhan data warisan budaya di masa depan. Warisan budaya dapat disimpan dalam bentuk objek 3D maupun objek 2D berupa dokumentasi foto atau gambar. Namun objek tersebut akan menjadi kurang bernilai jika tidak disertai dengan informasi yang mendukung, seperti nama, lokasi penemuan, perkiraan umur, siapa pembuatnya dan bahan apa yang digunakan. Informasi dari setiap objek warisan budaya tersebut harus terlampir, disusun sedemikian rupa sehingga mudah untuk dibaca, dikelola, maupun dicari kembali meskipun telah tersimpan lama. Struktur data seperti inilah yang disebut dengan metadata.

Metadata adalah data tentang data atau informasi tentang informasi. Metadata adalah bentuk *database* modern yang saat ini sudah banyak digunakan dalam bidang teknologi informasi. Dengan menggunakan sistem metadata informasi yang terkandung dapat digunakan oleh banyak sistem. Ada berbagai macam *framework* yang dapat digunakan untuk membangun metadata, salah satunya adalah RDF (*Resource Description Framework*). RDF adalah sebuah *framework* yang berasal dari teknologi *Extensible Markup Language* (XML), dan sedang dikembangkan di bawah naungan *World Wide Web Consortium* (W3C). RDF disusun menggunakan kode XML. Bahasa XML yang digunakan oleh RDF disebut RDF/XML [1].

Sebuah struktur metadata dengan RDF mengacu pada sebuah *file* RDFS (*Resource Description Framework Schema*). *File* ini berperan sebagai acuan (*rule*) dalam pembangunan metadata RDF. CIDOC (*ICOM's International Committee for Documentation*) adalah salah satu RDFS yang tersedia. CIDOC dibangun pada tahun 2006 untuk menciptakan sebuah acuan tentang entitas dan *property* yang digunakan dalam metadata warisan budaya [1].

Saat ini untuk dapat menerapkan struktur metadata yang kompleks pada data warisan budaya merupakan hal yang rumit bagi sebagian orang. Karena dibutuhkan keahlian khusus yang harus dimiliki. Selain itu untuk membangun sebuah metadata yang besar, seperti data warisan budaya, dibutuhkan waktu yang lama apabila hal tersebut dilakukan secara manual. Tidak bisa dipungkiri kesalahan (*human error*) dalam proses pembuatan kode juga menjadi kendala yang sering terjadi dalam pembangunan metadata.

Untuk mengatasi masalah tersebut maka dalam tugas akhir ini akan dibangun sebuah aplikasi berbasis *dekstop* yang dapat digunakan sebagai generator dalam membangun metadata untuk warisan budaya. Aplikasi ini dapat mengubah data dalam bentuk tabel (Excel) menjadi metadata dalam bentuk RDF/XML dengan mengacu pada RDF *Schema* CIDOC. Hal ini tentunya akan mengurangi kendala dalam pembangunan metadata warisan budaya, seperti *human error*. Selain itu dengan menggunakan aplikasi ini proses penyusunan kode metadata akan menjadi lebih cepat karena dilakukan secara otomatis oleh sistem.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana membuat sebuah sistem yang dapat membantu proses pembangkitan metadata koleksi warisan budaya?

2. Bagaimana sistem dapat menciptakan sebuah metadata koleksi warisan budaya yang seragam?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Aplikasi ini berbasis *dekstop* dengan bahasa pemrograman Java.
2. Dataset yang digunakan adalah data artefak Museum Indonesia.
3. RDF *Schema* yang digunakan adalah CIDOC-CRM versi 5.0.4.
4. Pengguna telah memahami struktur RDF *Schema* yang digunakan.
5. Aplikasi hanya dapat membaca Entitas dan *object property*.

1.4. Tujuan

Tujuan yang ingin dicapai dalam pengerjaan tugas akhir ini antara lain:

1. Membuat aplikasi yang dapat men-*generate* metadata koleksi warisan budaya yang sesuai dengan RDF *Schema* CIDOC.
2. Membuat aplikasi yang dapat men-*generate* metadata koleksi warisan budaya yang valid sesuai dengan struktur metadata RDF/XML.

1.5. Metodologi

Ada beberapa tahap dalam proses pengerjaan tugas akhir ini, yaitu sebagai berikut:

1. Studi Literatur

   Tahap ini akan mempelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu metadata, format metadata RDF/XML dan RDF *Schema* CIDOC. Bahan untuk studi literatur diambil dari sumber referensi tertulis, seperti buku dan internet.

2. Analisis dan Desain Perangkat Lunak

   Tujuan utama pembuatan aplikasi ini adalah untuk membantu proses pembangunan metadata warisan budaya. Metadata yang dihasilkan berbentuk RDF/XML yang sesuai dengan RDF *Schema* CIDOC. Pada tahap ini ditentukan spesifikasi kebutuhan dari aplikasi yang akan dibangun. Selain itu juga dirancang penggunaan dari komponen penyusun aplikasi seperti nama kelas, nama fungsi, algoritma, dan *framework* yang digunakan.

3. Implementasi

   Pada tahap ini desain aplikasi yang telah dibuat pada tahap sebelumnya akan diimplementasikan ke dalam kode program menggunakan bahasa pemrograman Java dengan bantuan *framework* Jena, *framework* khusus yang digunakan dalam proses pembuatan metadata. *Framework* ini digunakan untuk proses pembacaan Entitas dan *Object Property*. Aplikasi dibangun menggunakan *Integrated Development Environment* (IDE) Netbeans.

4. Uji coba dan Evaluasi

   Pengujian dilakukan dengan men-*generate* dataset artefak Museum Indonesia yang tersedia dalam bentuk tabel Excel. Hasil keluaran sistem yang berupa kumpulan *file* RDF/XML akan diuji kebenarannya menggunakan kakas bantu Protege. Mencoba menegtikan kata

5. Penyusunan buku tugas akhir

   Tahap ini merupakan tahap penyusunan laporan berupa buku tugas akhir sebagai dokumentasi pelaksanaan tugas akhir, yang mencakup seluruh teori, implementasi, serta hasil pengujian yang telah dikerjakan.

1.6. Sistematika Penulisan

Sistematika penulisan dibuat dengan tujuan untuk mendapatkan gambaran umum dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut terhadap penelitian ini. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Metodologi Pemecahan Masalah

Bab ini membahas mengenai metode yang digunakan untuk memecahkan masalah yang dipaparkan pada rumusan permasalahan.

Bab IV Analisis dan Perancangan Sisten

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada perangkat lunak.

Bab V Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak dan implementasi fitur-fitur penunjang.

Bab VI Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan (*usability*) dari perangkat lunak dan pengujian fungsionalitas yang dibuat dengan memperhatikan keluaran yang dihasilkan serta evaluasi terhadap fitur-fitur perangkat lunak.

Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Bagian ini akan membahas dasar teori dan literatur yang menjadi dasar pengerjaan tugas akhir ini. Literatur diambil dari berbagai sumber tertulis seperti buku dan internet.

2.1. Warisan Budaya

Warisan budaya adalah bentuk ekspresi dari pola hidup kelompok masyarakat yang diwariskan secara turun temurun dari genenrasi ke generasi berikutnya, seperti cara berpakaian, praktek ritual, tempat, objek, aliran seni dan norma yang berlaku. Warisan budaya dapat diekspresikan dalam bentuk sesuatu yang berwujud maupun tidak berwujud [2].

Warisan budaya adalah benda atau atribut tak berbenda yang merupakan jati diri suatu masyarakat atau kaum yang diwariskan dari generasi-generasi sebelumnya, yang dilestarikan untuk generasi-generasi yang akan datang. Warisan budaya dapat berupa benda, seperti monumen, artefak, dan kawasan, atau tak benda, seperti tradisi, bahasa, dan ritual [3].

Warisan budaya merupakan suatu aset penting yang digunakan sebagai sumber informasi dalam mempelajari ilmu sejarah. Mengelola data warisan budaya menjadi suatu hal yang harus diperhatikan guna menjaga keutuhan data warisan budaya di masa depan. Data warisan budaya dapat disimpan dalam bentuk objek 3D maupun objek 2D berupa dokumentasi foto atau gambar.

Usaha untuk melestarikan warisan budaya disebut konservasi, misalnya dengan perlindungan, dokumentasi, pemulihan, dan mengumpulkan di museum. Salah satu organisasi yang mempromosikan pelestarian warisan budaya dalah UNESCO [4].

2.2. Metadata

Metadata adalah informasi terstruktur yang mendeskripsikan, menjelaskan, menemukan, atau setidaknya menjadikan suatu informasi mudah untuk ditemukan kembali, digunakan, atau dikelola. Metadata sering disebut sebagai data tentang data atau informasi tentang informasi [5].

Metadata adalah informasi tambahan yang menyertai dan mendeskripsikan tentang sebuah data tertentu. Misalnya, sebuah gambar memiliki metadata yang menginformasikan seberapa besar ukuran *file* gambar, kedalaman warnanya, resolusinya, kapan dibuat, dan sebagainya. Contoh lain, metadata sebuah dokumen teks berisi informasi tentang seberapa panjang dokumen tersebut, siapa yang membuat, kapan ditulis, dan ringkasan isinya. Adapun metadata pada halaman *website* adalah bagian yang dituliskan pada *tag meta* di bagian *header* halaman *web*, misalnya deskripsi singkat tentang *web*site dan *keyword*-nya [6].

Metadata dalam konteks warisan budaya berarti informasi yang menjelaskan tentang sebuah objek peninggalan warisan budaya. Misal terdapat objek cermin yang ditemukan sebagai warisan budaya, maka metadata dari cermin tersebut adalah nama, lokasi penemuan, tahun ditemukan, terbuat dari apa, dan lokasi penyimpanan saat ini. Informasi tersebut disusun sedemikian rupa sehingga mudah untuk dibaca, dikelola, maupun dicari kembali meskipun telah tersimpan lama.

Metadata berisi berbagai informasi dan dapat dipecah menjadi tiga jenis utama: metadata deskriptif, metadata struktural, dan metadata administrative [6].

1. Metadata deskriptif berisi informasi atau kata kunci yang menjelaskan *resource* dan membantu menemukan *resource* dalam suatu sistem. Sebuah katalog buku adalah contoh dari metadata deskriptif. Katalog buku umumnya berisi informasi seperti nomor buku, tema, judul, dan penulis yang dapat digunakan untuk mencari buku-buku tertentu di perpustakaan. Metadata deskriptif juga biasa digunakan untuk membuat halaman *web*. Metadata

deskriptif pada halaman *web* akan berisi informasi seperti kata kunci, deskripsi, set karakter, bahasa, dan bagaimana Anda ingin jaringan mesin pencari mengindeks halaman *web*.

2. Metadata struktural menggambarkan struktur obyek yang terdiri dari tabel dan kolom dalam *database*, daftar lagu pada CD, atau halaman dalam sebuah buku. Metadata strukrutal membantu pengguna mengeksplorasi *resource* menggunakan indeks dan bahkan menggambarkan bagaimana beberapa objek berhubungan dengan yang lain, seperti urutan *file*. Metadata struktural dapat mengandung informasi tambahan termasuk batas-batas logis, posisi dalam memori, dan menjelaskan bagaimana dan mengapa sebuah objek digunakan.
3. Metadata administratif membantu pemakai mengelola *resource* dan menyediakan informasi seperti jenis *file*, tanggal pembuatan, kontrol kualitas, dan frekuensi penggunaan. Jenis metadata ini digunakan untuk merekam informasi jangka pendek dan jangka panjang tentang kumpulan data. Sebagai contoh, metadata administratif tentang monitor komputer akan berisi ukuran monitor, model, resolusi, warna, cahaya, dan kebutuhan daya listrik.

Dalam *semantic web* metadata ditulis dalam kode RDF/XML. Dengan menggunakan format XML yang merupakan dasar pembentukan metadata (RDF), maka metadata dapat didistribusikan atau di simpan dalam domain yang beragam. Data yang tersimpan berupa BibTex *file* yang dikonversi ke format RDF, selanjutnya dengan menggunakan kakas bantu Sesame, metadata di *load* pada *file* indeks, SeRQL selanjutnya melakukan *query* terhadap metadata yang telah dimuat untuk ditampilkan di *browser* [7].

2.3. ontologi

ontologi merupakan suatu teori tentang makna dari suatu objek, serta hubungan objek tersebut yang mungkin terjadi pada suatu domain pengetahuan. *Ontologi* juga merupakan sebuah uraian formal yang menjelaskan tentang sebuah konsep dalam sebuah domain tertentu (*Classes*, terkadang disebut konsep), *properties* dari masing-masing konsep menjelaskan bermacam-macam fitur dan atribut dari sebuah konsep (*Slots*, terkadang disebut *rules* atau *properties*). Sebuah ontologi bersama dengan seperangkat *instances* (menyatakan objek pada suatu domain) dari class membentuk sebuah *knowledge base*.

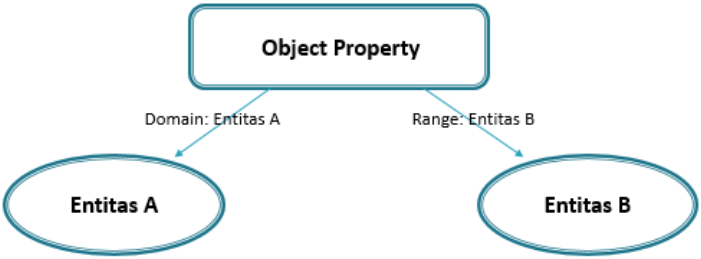
Secara umum ontologi digunakan pada *Artificial Intelligence* (AI) dan persentasi pengetahuan. segala bidang ilmu yang ada di dunia, dapat menggunakan metode ontologi untuk dapat berhubungan dan saling berkomunikasi dalam hal pertukaran informasi antara sistem-sistem yang berbeda.

ontologi terdiri dari beberapa komponen penyusun yang digunakan untuk mendeskripsikan sebuah pengetahuan terhadap objek. Komponen tersebut adalah *Classes, Instance,* dan *Relation*. Penjelasen dari setiap komponen tersebut diuraikan sebagai berikut.

Classes merupakan definisi dari tipe suatu objek yang memiliki karakteristik dan atribut yang sama. *Classes* dapat mendefinisikan objek yang berwujud seperti manusia, mobil, dan artefak warisan budaya. *Classes* juga dapat mendefinisikan suatu objek yang tak berwujud seperti, konsep, suatu keadaan dan waktu. *Classes* dalam Tugas Akhir ini selanjutnya akan disebut sebagai Entitas. Entitas ini selanjutnya akan digunakan sebagai dasar *instance* dari sebuah objek warisan budaya. Dalam konsep *Classes* juga terdapat *subclass* dan *superclass*. *Subclass* merupakan bentuk definisi yang lebih khusus dari sebuah *superclass.* Setiap *subclass* pasti akan mewarisi sifat dan atribut dari leluhurnya. Namun *subclass* juga dapat memiliki atribut sendiri yang tidak dimiliki oleh *superclass*-nya. Contoh dari *subclass* adalah kelas Pria yang merupakan *subclass* dari kelas Manusia.

Instance atau yang disebut *Individual* adalah bentuk realisasi dari kelas. Misalkan terdapat kelas Manusia dan terdapat objek bernama Alex yang memenuhui kriteria kelas manusia, maka objek Alex tersebut dapat dijadikan sebagai bentuk *Instance* dari Manusia.

Relation adalah suatu penghubung antara dua objek dalam konsep ontologi. Terdapat dua jenis *relation* dalam ontologi yaitu *Data property* dan *object property*. *Data property* merupakan relasi yang menghubungkan sebuah entitas dengan objek literal (objek literal merupakan sebuah nilai, dapat berupa nominal atau kata, untuk mendeskripsikan informasi dari sebuah entitas). Sedangkan object *property* adalah sebuah relasi yang menghubungkan dua buah entitas yang telah didefinisakan sebelumnya. *Object property* memiliki atribut domain dan *range* yang memiliki nilai sebuah entitas. Contoh hubungan Entitas dan *Object Property* dapat dilihat pada Gambar 2.1 berikut.



Gambar 2.1 Hubungan Entitas dan *Object Property*

2.4. RDF/XML

Resource Description Framework (RDF) adalah standar *World Wide Web* Consortium (W3C) yang didesain untuk pemodelan metadata. Penulisan *file* RDF berbasiskan format *file* XML sehingga sering dikenal dengan format RDF/XML [8]. Kode RDF ditulis dengan menggunakan pola *triplet*. Pola *triplet* berarti kode RDF disusun seperti sebuah kalimat, yang terdiri dari tiga

komponen utama, Subjek, Predikat, dan Objek. Subjek dalam RDF berisi sebuah kelas (*Entity*). Predikat berisi sebuah keterangan yang menjelaskan kelas subjek (*Data Property*). Dan objek dapat berisi sebuah kelas maupun sebuah nilai. Contoh definisi class dalam format RDF:

```
<rdfs:Class rdf:about=http://localhost#Person>
      <rdfs:isDefinedBy
rdf:resource=http://localhost#/>
      <rdfs:label>Person</rdfs:label>
</rdfs:Class>
```

Tiap satu *tag* dalam RDF tersebut merepresentasikan satu kode *triplet*. Bisa dilihat bahwa terdapat beberapa data *triplet* yaitu `<rdfs:Class rdf:about=http://localhost#Person> </rdfs:Class>` yang mendefinisikan bahwa *Person* merupakan *Class*. Yang kedua adalah `<rdfs:isDefinedBy rdf:resource=http://localhost#/>` yang mendefiniskan bahwa `http://localhost#Person` didefiniskan oleh `http://localhost#` yang artinya *Person* berada pada salah satu alamat lokasi `http://localhost#`. Apabila ada data lain, misalnya ditambahkan kelas *Female* yang didefinisikan pada tempat yang sama yaitu di `http://localhost#` maka dapat dikatakan bahwa *Person* dan *Female* berada pada lokasi yang sama. Dan *tag* yang terakhir adalah `<rdfs:label>Person</rdfs:label>` yang mendefinisikan bahwa label dari *class* adalah *Person*. Jadi *class* dengan definisi nama *Person* sama dengan *class* dengan definisi `<http://localhost#Person>`.

RDF diperuntukkan pada situasi dimana informasi akan diproses lebih lanjut oleh sebuah aplikasi, dan bukan hanya ditampilkan ke pengguna saja. RDF menawarkan sebuah *framework* untuk merepresentasikan informasi sekaligus untuk menjaga agar informasi tidak kehilangan maksud yang dikandungnya ketika dipertukarkan antara aplikasi yang berbeda [9].

2.5. *Resource Description Framework Schema* (RDFS)

Resource Description Framework Schema (RDFS), atau yang biasa dikenal dengan RDF *Schema*, adalah bentuk pengembangan dari *file* RDF. RDF *Schema* merupakan sebuah *file* yang menjelaskan mekanisme untuk menyusun sebuah RDF. Di dalam RDF *Schema* dijelaskan nama dan karakteristik dari setiap kelas (Entitas) dan relasi yang menggambarkan hubungan dari setiap kelas tersebut (*Data Property*). RDF *Schema* disusun dengan menggunakan kode RDF [10].

RDF *Schema* digunakan sebagai patokan (*rule*) dalam penyusunan *file* RDF. Dengan mengacu pada sebuah RDF *Schema* maka dapat diketahui hubungan dari tiap kelas yang terdefinisi melalui karakteristik *Data Property* yang digunakan. Dimana di RDF *Schema* dijelaskan domain dan range dari setiap Data *Property*.

Aplikasi metadata generator ini dibangun dengan menggunakan RDF *Schema* CIDOC yang merupakan RDF *Schema* khusus untuk membangun metadata warisan budaya.

2.6. CIDOC CRM

CIDOC (*ICOM's International Committee for Documentation*) adalah salah satu RDFS yang tersedia. CIDOC dibangun pada tahun 2006 untuk menciptakan sebuah acuan tentang *entity* dan *property* yang digunakan dalam penyusunan metadata warisan budaya [1]. CIDOC *Conceptual Reference Model* (CRM) menyediakan definisi dan struktur yang formal untuk menggambarkan kelas implisit maupun eksplisit dan hubungan yang digunakan dalam dokumentasi data warisan budaya [11].

CIDOC CRM disusun untuk menciptakan pemahaman bersama tentang informasi warisan budaya dengan menyediakan kerangka kerja *semantic*. Hal ini dimaksudkan agar para pakar metadata memiliki satu acuan yang sama dalam mengembangkan

metadata warisan budaya sehingga dapat dihasilkan suatu format metadata yang seragam [10].

Di dalam CIDOC CRM terdapat 90 entitas untuk mendefinisikan objek warisan budaya. Baik entitas yang mendefinisikan objek yang nampak, seperti E19_Physical_Thing dan E18_Physical_*Object*, maupun objek yang tidak nampak seperti E2_Temporal_Entity dan E4_Period. Selain itu juga terdapat 149 *data property* yang mendeskripsikan relasi antar entitas melalui hubungan domain – *range* yang telah ditentukan.

2.7. *Framework* Jena

Jena merupakan sebuah *framework* khusus yang digunakan dalam bidang *semantic web*, terutama dalam hal membangun sebuah metadata RDF. *Framework* Jena merupakan *Framewok* yang dilengkapi dengan API untuk mengambil dan menulis data ke grafik RDF. Grafik ini diwakili sebagai "model" abstrak. Sebuah model dapat bersumber dengan data dari *file*, *database*, URL atau kombinasi dari semuanya. Sebuah model juga dapat dilihat melalui SPARQL dan diperbarui melalui SPARUL [12].

Framework Jena menyediakan sebuah API yang digunakan untuk membaca dan menulis *file* RDF yang terdapat dalam *package* Apache Jena RIOT (RDF I/O *technology*) [13]. *Package* RIOT ini dapat mengelola beberapa bentuk format *file* RDF seperti:

- Turtle
- N-Triples
- RDF/XML
- JSON-LD
- RDF/JSON
- TriG
- NQuads
- TriX

BAB III
METODOLOGI PEMECAHAN MASALAH

Pada bab ini dijelaskan mengenai langkah-langkah yang dilakukan untuk menyusun skema metadata warisan budaya. Mulai dari metode yang dilakuakn untuk menyiapkan dataset, menentukan RDF *Schema*, memilih *entity* dan *data property* yang digunakan, sampai proses penulisan metadata yang dilakukan oleh sistem metadata generator.

3.1. Analisis Data

Untuk menyusun sebuah metadata dari sebuah objek warisan budaya dibutuhkan informasi terkait objek tersebut seperti nama, dimensi, lokasi penemuan, dan tahun penemuan. Tugas akhir ini menggunakan informasi dari Artefak Museum Indonesia yang digunakan sebagai dataset. Proses pembangunan metadata dilakukan dengan berdasar pada RDF *Schema* CIDOC yang merupakan *schema* khusus untuk metadata warisan budaya. Selain itu dalam proses pembuatan metadata juga ditulis sebuah RDF *Schema* baru yang merupakan *file* konfigurasi dari pengguna.

3.1.1. Analisis Data Artefak Monumen Nasional

Data dari Monumen Nasional berupa *file* Excel. Dalam *file* tersebut ada beberapa atribut antara lain: No.Urut, No.InvB, No_FotoGab, Nama_Benda, Tempat_Asal_Des_Kec, Des_Gabungan1, Kondisi, Lokasi_Gedung1, Lokasi Ruang1, Lokasi Lemari_Laci1, Lokasi_Lain1, Tempat_Asal_Kab, Tempat_Asal_Prop, Tempat Pembuatan, Tempat Perolehan, Tahun Pembuatan. Penamaan atribut pada dataset tersebut kurang proporsional untuk digunakan sebagai dataset Tugas Akhir ini. Terdapat pengulangan data dan beberapa atribut yang menampung lebih dari satu informasi. Sehingga perlu dilakukan perubahan pada atribut untuk disesuaikan dengan sistem yang dibangun. Berikut beberapa perubahan yang dilakukan untuk menyesuaikan dataset.

- Menghapus No.InvB dan No_FotoGab dan menjadikan No Urut sebagai identitas unik dari setiap artefak.
- Menghilangkan atribut Des_Gabungan1, yang merupakan deskripsi gabungan dari setiap artefak. Pada atribut ini dideskripsikan beberap informasi mengenai artefak terkait dalam satu kolom. Sedangkan terdapat beberapa atribut yang telah dideskripsikan pada kolom atribut lain sehingga terjadi pengulangan data. Selain itu ada beberapa informasi mengenai dimensi artefak seperti panjang, lebar, tinggi, dan tebal yang perlu dideskripsikan secara terpisah. Sehingga ditambahkan beberapa atribut baru untuk menampung informasi dimensi artefak seperti: Ukuran panjang, ukuran lebar, ukuran tinggi, ukuran tebal, dan ukuran dimensi.
- Mendeskripsikan data kosong dengan karakter kosong. Data kosong pada dataset ini dideskripsikan dengan karakter ‘ - ’ dan karakter kosong. Dalam Tugas Akhir ini data kosong didefiniskan2 dengan karakter kosong.
- Memperbaiki nama atribut menjadi lebih formal.

Pada akhir proses penyesuaian didapatkan 19 nama atribut yang digunakan dalam proses pembangkitan metadata. Penjelasan dari setiap atribut disajikan pada Tabel 3.1 berikut.

Tabel 3.1 Deskripsi Atribut Dataset

No	Nama Atribut	Penjelasan
1	No Urut	Berisi nomor urut penemuan artefak. Pada Tugas Akhir ini dijadikan sebagai identitas unik dari masing-masing artefak.
2	Nama Benda	Nama artefak
3	Tempat Asal Des/Kec	Desa atau kecamatan tempat ditemukannya artefak
4	Tempat Asal Kab	Kabupaten tempat ditemukannya artefak

5	Tempat Asal Prop	Proponsi tempat ditemukannya artefak
6	Lokasi Gedung	Lokasi gedung tempat penyimpanan artefak saat ini
7	Lokasi Ruang	Lokasi ruang tempat penyimpanan artefak saat ini
8	Lokasi Lemari/Laci	Lokasi lemari/laci tempat penyimpanan artefak saat ini
9	Lokasi Lain	Lokasi lain tempat penyimpanan artefak saat ini
10	Kondisi	Kondisi artefak
11	Bahan	Bahan yang digunakan untuk membuat artefak
12	Ukuran Panjang	Ukuran panjang artefak
13	Ukuran Lebar	Ukuran lebar artefak
14	Ukuran Tinggi	Ukuran tinggi artefak
15	Ukuran Tebal	Ukuran tebal artefak
16	Ukuran Diameter	Ukuran diameter artefak
17	Tempat Perolehan	Tempat perolehan artefak
18	Tempat Pembuatan	Perkiraan tempat pembuatan artefak
19	Tahun Pembuatan	Perkiraan tahun pembuatan artefak

Contoh tabel *dataset* warisan budaya yang digunakan dalam Tugas Akhir dapat dilihat pada Lampiran.

3.1.2. Analisis RDF *Schema* CIDOC-CRM

RDF *Schema* CIDOC *Conceptual Reference Model* (CRM) merupakan ontologi resmi yang digunakan sebagai skema untuk membangun metadata warisan budaya. Skema ini mendeskripsikan 81 Entitas dan 262 *Object Property* terkait dengan objek warisan budaya. Karena ontologi tersebut merupakan ontologi resmi maka tidak perlu dilakukan perubahan terhadapnya. Sistem akan membaca data keseluruhan untuk selanjutnya disimpan dalam variabel larik Entitas dan larik *Object Property*.

Contoh deskripsi entitas dan *object property* dari skema CIDOC dapat dilihat pada Gambar 3.1 dan Gambar 3.2 berikut.

```
<rdfs:Class rdf:about="E2_Temporal_Entity">
    <rdfs:label xml:lang="fr">Entité temporelle</rdfs:label>
    <rdfs:label xml:lang="en">Temporal Entity</rdfs:label>
    <rdfs:label xml:lang="ru">Временная Сущность</rdfs:label>
    <rdfs:label xml:lang="el">Εγχρονη  Οντότητα</rdfs:label>
    <rdfs:label xml:lang="de">Geschehendes</rdfs:label>
    <rdfs:label xml:lang="pt">Entidade Temporal</rdfs:label>
    <rdfs:comment>This class comprises all phenomena, such as
    the instances of E4 Periods, E5 Events and states, which
    happen over a limited extent in time.
    In some contexts, these are also called perdurants. This
    class is disjoint from E77 Persistent Item. This is an
    abstract class and has no direct instances. E2 Temporal
    Entity is specialized into E4 Period, which applies to a
    particular geographic area (defined with a greater or lesser
    degree of precision), and E3 Condition State, which applies
    to instances of E18 Physical Thing.
</rdfs:comment>
    <rdfs:subClassOf rdf:resource="E1_CRM_Entity"/>
</rdfs:Class>
```

Gambar 3.1 Contoh Entitas CIDOC-CRM

```
<rdf:Property rdf:about="P2_has_type">
    <rdfs:label xml:lang="de">hat den Typus</rdfs:label>
    <rdfs:label xml:lang="en">has type</rdfs:label>
    <rdfs:label xml:lang="el">έχει τύπο</rdfs:label>
    <rdfs:label xml:lang="fr">est de type</rdfs:label>
    <rdfs:label xml:lang="ru">имеет тип</rdfs:label>
    <rdfs:label xml:lang="pt">é do tipo</rdfs:label>
<rdfs:comment>This property allows sub typing of CRM entities -
a form of specialisation - through the use of a terminological
hierarchy, or thesaurus.
The CRM is intended to focus on the high-level entities and
relationships needed to describe data structures. Consequently,
it does not specialise entities any further than is required
for this immediate purpose. However, entities in the isA
hierarchy of the CRM may by specialised into any number of sub
entities, which can be defined in the E55 Type hierarchy. E51
Contact Point, for example, may be specialised into "e-mail
address", "telephone number", "post office box", "URL" etc.
none of which figures explicitly in the CRM hierarchy. Sub
typing obviously requires consistency between the meaning of
the terms assigned and the more general intent of the CRM
entity in question.
</rdfs:comment>
    <rdfs:domain rdf:resource="E1_CRM_Entity"/>
    <rdfs:range rdf:resource="E55_Type"/>
</rdf:Property>
```

Gambar 3.2 Contoh *Object Property* CIDOC-CRM

Penjelasan atribut dari Gambar 3.1 dan Gambar 3.2 disajikan dalam Tabel 3.2 berikut.

Tabel 3.2 Penjelasan Atribut

No	Atribut	Penjelasan
1	rdf:about	Menjelaskan nama entitas atau *object property*
2	rdfs:label	Menjelaskan nama *object* melalui berbagai bahasa (deutch, eglish, el, fr, ru, pt)
3	rdfs:comment	Berisi penjelasan singkat dari entitas dan *object property*
4	rdfs:domain	Berisi domain dari sebuah *object property*
5	rdfs:range	Berisi *range* dari sebuah *object property*

3.2. Proses Pemilihan Entitas dan *Object Property*

Aplikasi yang dibangun bertujuan untuk memudahkan pengguna dalam men-*generate* metadata warisan budaya sesuai dengan kebutuhan masing-masing. Sehingga entitas dan *object property* yang digunakan untuk membangun metadata dipilih sesuai dengan kebutuhan pengguna.

3.2.1. Pemilihan Entitas

Pada tahap sebelumnya sistem telah menyimpan setiap entitas dalam sebuah *array* Entitas. Selanjutnya pengguna akan menentukan entitas yang cocok untuk setiap atribut pada dataset Excel Artefak Museum Nasional. Setiap atribut dapat dipetakan ke dalam entitas yang sama atau berbeda satu sama lain tergantung dari kebutuhan pengguna. Hasil pemetaan atribut dengan entitas selanjutnya disimpan dalam *file* Mapping.txt yang secara otomatis dimunculkan oleh sistem. *File* tersebut dapat digunakan untuk membangun metadata warisan budaya di waktu selanjutnya.

Setiap entitas yang terdapat pada RDF *Schema* CIDOC (kecuali E1_CRM Entity) merupakan bentuk penurunan dari entitas yang lebih besar. Konsep tersebut didefinisikan dalam atribut *subClassOf*. Pengguna juga dapat menentukan tingkat kedalaman entitas yang dipilihnya. Misalkan pengguna menentukan tingkat kedalaman entitas sebesar dua, maka entitas yang terbaca oleh sistem adalah entitas terpilih dan dua tingkat entitas di atasnya yang ditentukan dari atribut *subClassOf*.

3.2.2. Pemilihan *Object Property*

Pada tahap sebelumnya sistem telah menyimpan setiap entitas dalam sebuah larik *object Property*. Sistem akan menampilkan *object property* yang terkait dengan entitas terpilih. Terkait disini berarti *object property* mengandung entitas terpilih pada domain, *range*, atau keduanya.

Misalkan pengguna memetakan sebuah atribut dengan entitas E53_Place, maka sistem akan menampilkan semua *object property* yang domain dan/atau *range*-nya adalah E53_Place. Dalam contoh kasus ini hasil yang dimunculkan akan seperti Tabel 3.3 berikut.

Tabel 3.3 Contoh *Object Property* yang Memiliki Hubungan dengan Entitas E53_Place

No	*Object Property*	Domain	Range
1	P122_borders_with	E53_Place	E53_Place
2	P88_consist_of	E53_Place	E53_Place
3	P89i_contains	E53_Place	E53_Place
4	P89_falls_within	E53_Place	E53_Place
5	P55_has_current_location	E19_Physical_*Object*	E53_Place
6	P26_moved_to	E9_Move	E53_Place
7	P27_moved_from	E9_Move	E53_Place
8	P7_took_place_at	E4_Period	E53_Place
9	P121_overlaps_with	E53_Place	E53_Place
10	P59_has_section	E18_Physical_Thing	E53_Place

Pengguna akan menentukan *object property* apa saja yang digunakan untuk membangun metadata warisan budaya sesuai dengan kebutuhan.

3.3. Pembangunan RDF *Schema Configuration*

Kombinasi dari entitas dan *object property* yang dipilih oleh pengguna merupakan sebuah pengaturan dari proses pembangkitan metadat oleh sistem. Proses ini dapat memakan waktu yang cukup lama karena pengguna harus mengerti mengenai apa yang dipilih. Untuk itu hasil konfigurasi tersebut disimpan dalam sebuah RDF *Schema,* yang selanjutnya disebut sebagai RDF *Schema Configuration,* guna menghemat waktu dalam proses pembangunan metadata selanjutnya. RDF *Schema Configuration* merupakan bentuk penyederhanaan dari RDF *Schema* CIDOC-CRM. RDF *Schema Configuration* hanya mendeskripsikan entitas dan *object property* yang dipilih oleh pengguna. Dengan adanya RDF *Schema Configuration* dapat memudahkan pengguna dalam proses pembangunan metadata di waktu selanjutnya. Pengguna dapat memasukkan RDF *Schema Configuration* sebagai acuan untuk membangun metadata tanpa melalu proses konfigurasi.

3.4. Pembangunan Metadata Warisan Budaya

Metadata warisan budaya dibangun dalam bentuk format RDF/XML. Format tersebut menggunakan *tag* XML untuk mendeskripsikan sebuah objek. Deskripsi objek dalam bentuk RDF ditulis sebagai *individual* yang dinotasikan di dalam *tag* "NamedIndividual".

Bentuk metadata yang dibangun merupakan generalisasi dari entitas terpilih, *object property* terpilih dan dataset artefak Museum Indonesia. Setiap atribut dari artefak didefinisikan sebagai entitas sesuai dengan pemetaan pengguna. Selanjutnya setiap atribut dihubungkan dengan atribut lain sesuai dengan *object property* terpilih dan data pada dataset Excel Museum Indonesia. Proses ini menghasilkan *file* RDF/XML untuk setiap artefaknya. Apabila dalam dataset terdapat 10 artefak, maka sistem akan

menghasilkan 10 *file* RDF/XML. Contoh *file* RDF/XML yang akan dibangun disajikan pada Gambar 3.3 berikut.

```
<owl:NamedIndividual rdf:about="AA">
    <rdf:type rdf:resource="cidoc-crm:BB"/>
    <cidoc-crm:C1 rdf:resource="D1"/>
    <cidoc-crm:C2 rdf:resource="D2"/>
</owl:NamedIndividual>
```

Gambar 3.3 Format RDF/XML

Kode tersebut mendeskripsikan nilai atribut pada sebuah artefak. Penjelasan kode tersebut disajikan dalam Tabel 3.4. Dalam sebuah data artefak pada *dataset* terdapat beberapa nilai atribut sehingga metadata sebuah artefak dalam *file* RDF/XML akan berisi beberapa potongan kode Gambar 3.3 tergantung dari jumlah atribut pada artefak.

Tabel 3.4 Deskripsi Format RDF/XML

No	Kode	Penjelasan
1	AA	Bagian ini akan diisi nilai dari sebuah atribut pada sebuah artefak.
2	BB	Bagian ini akan diisi dengan jenis entitas dari RDF *Schema* CIDOC-CRM yang telah dipetakan dengan atribut terkait.
3	C1, C2	Bagian ini akan diisi dengan *object property* terpilih yang memiliki entitas domain BB
4	D1, D2	Bagian ini akan diisi dengan nilai atribut yang dipetakan terhadap entitas yang sesuai dengan *range* C1/C2

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan tentang analisis permasalahan dan perancangan Tugas Akhir. Analisis permasalahan membahas tentang permasalahan yang diangkat dalam Tugas Akhir ini beserta solusi yang ditawarkan. Selanjutnya dibahas juga tentang perancangan sistem yang dibuat.

4.1. Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan, deskripsi umum sistem, kasus penggunaan sistem dan kebutuhan perangkat lunak.

4.1.1. Cakupan Permasalahan

Indonesia adalah negara yang memiliki warisan budaya beraneka ragam. Warisan budaya merupakan peninggalan yang wajib dijaga keberadaannya. Selama ini warisan budaya yang berupa benda seperti artefak disimpan di dalam museum. Namun seringkali data artefak yang ada masih terdokumentasi secara manual yang membutuhkan waktu yang lama dan sumber daya yang besar. Hal itu terjadi karena belum ada sistem yang dapat membantu proses pembangunan metada warisan budaya secara otomatis.

Dengan adanya permasalahan tersebut, dalam Tugas Akhir ini dibuat sebuah aplikasi berbasis *dekstop* sebagai sarana untuk membantu proses pembangunan metadata warisan budaya secara otomatis. Aplikasi ini dapat men-*generate* metadata dari ribuan data artefak dalam waktu yang relatif singkat. Selain itu melalui aplikasi ini *human error* dalam proses pembangunan metadata dapat ditekan.

4.1.2. Deskripsi Umum Sistem

Aplikasi yang akan dibangun merupakan aplikasi berbasis *dekstop* dan dibangun menggunakan bahasa pemrograman Java. Untuk mendukung proses pembangkitan metadata digunakan *framework* Jena yang merupakan salah satu *framework* khusus untuk pembangunan metadata. Secara garis besar proses pembangkitan metadata pada aplikasi ini ditunjukkan pada Gambar 4.1.

Gambar 4.1 Proses Pembangkitan Metadata

Dari Gambar 4.1 di atas dapat dilihat bahwa dalam men-*generate* metadata dibutuhkan lima proses utama: *Import File, Entity Configuration, Object Property Configuration, Saving, Configuration,* dan *Generating RDF/XML*. Detail dari masing-masing proses akan dijelaskan pada Subbab 4.2

4.1.3. Spesifikasi Kebutuhan Perangkat Lunak

Bab ini menjelaskan kebutuhan perangkat lunak dalam bentuk diagram kasus dan diagram aktivitas. Masing-masing diagram menjelaskan perilaku atau sifat dari sistem yang dibangun.

4.1.3.1. Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan pokok yang harus dipenuhi agar sistem dapat berjalan dengan baik. Daftar kebutuhan fungsional dapat dilihat pada Tabel 4.1.

.

Tabel 4.1 Daftar Kebutuhan Fungsional Perangkat Lunak

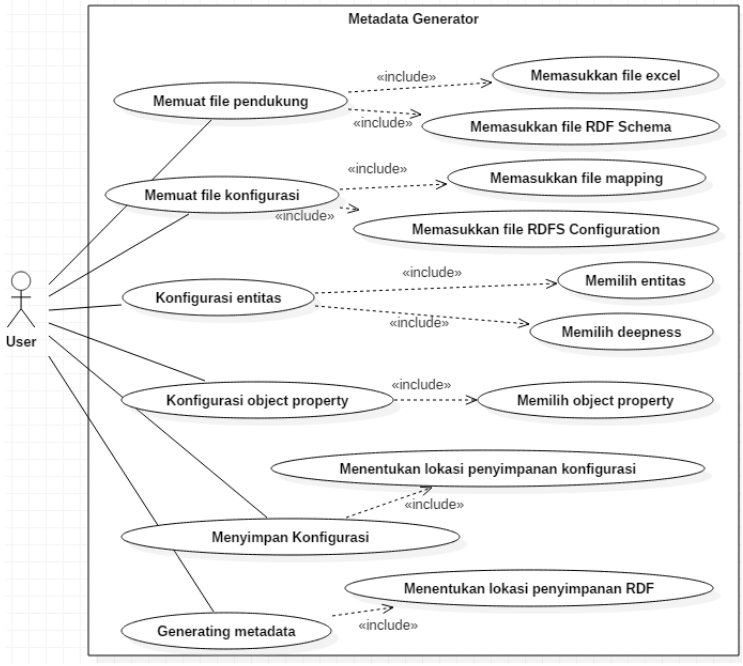
Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
MG-F01	Memasukkan *Dataset*	Memilih dan memasukkan *dataset* artefak yang berbentuk *file* Excel
MG-F02	Memasukkan RDFS CIDOC	Memilih dan memasukkan RDFS CIDOC sebagai acuan konfigurasi
MG-F03	Memuat *File* Konfiigurasi	Memilih dan memasukkan *Mapping file* dan RDFS *Configuration* sebagai acuan pembangkitan metadata
MG-F04	Konfigurasi Entitas	Memilih entitas CIDOC yang akan digunakan dalam proses pembangkitan metadata
MG-F05	Menentukan *Threshold*	Menentukan kedalam entitas
MG-F06	Konfigurasi *Object Property*	Memilih *object property* CIDOC yang akan digunakan dalam proses pembangkitan metadata
MG-F07	Menyimpan Konfigurasi	Menyimpan pengaturan yang telah dilakukan
MG-F08	*Generating* Metadata	Menyimpan *file* RDF/XML yang telah di-*generate* oleh sistem.

4.1.4. Aktor

Aktor merupakan entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas yang dimaksud dapat berupa manusia, sistem, atau perangkat lunak yang lain. Aktor yang berinteraksi dengan Tugas Akhir ini adalah pengguna sistem

yang dapat men-*generate* metadata warisan budaya melalui beberapa tahapan konfigurasi yang telah dijelaskan.

4.1.5. Kasus Penggunaan



Gambar 4.2 Diagram Kasus Penggunaan Sistem

Kasus penggunaan dalam subbab ini akan dijelaskan secara rinci. Kasus penggunaan dijabarkan dalam bentuk spesifikasi kasus penggunaan dan diagram aktivitas. Diagram kasus penggunaan dapat dilihat pada Gambar 4.2 . Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Tabel 4.2.

Tabel 4.2 Daftar Kode Diagram Kasus Penggunaan

Kode Kasus Penggunaan	Nama Kasus Penggunaan
UC01	Memuat *File* Pendukung
UC02	Memuat *File* Konfigurasi
UC03	Konfigurasi Entitas
UC04	Konfigurasi *Object Property*
UC05	Menyimpan Konfigurasi
UC06	*Generate* Metadata

4.1.5.1. Kasus Penggunaan Memuat *File* Pendukung

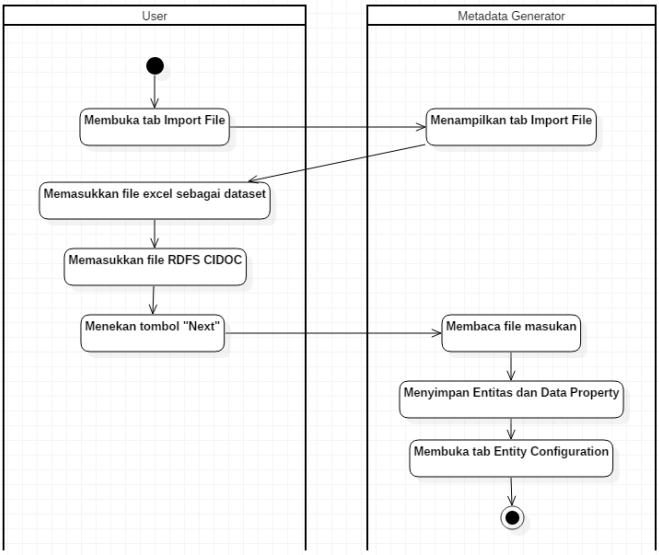
Pada kasus penggunaan memuat *file* pendukung, pengguna memasukkan beberapa *file* pendukung proses pembangunan metadata. *File* tersebut adalah *file dataset* yang berbentuk Excel, dan *file* RDFS CIDOC-CRM.. Penjelasan skenario yang lebih rinci dapat dilihat pada Tabel 4.3.

Tabel 4.3 Spesifikasi Kasus Penggunaan Memuat *File* Pendukung

Nama Kasus Penggunaan	Memuat *File* Pendukung
Nomor	UC01
Aktor	Pengguna
Kondisi Awal	Pengguna telah memiliki *file* pendukung berupa *dataset* dan RDFS CIDOC
Kondisi Akhir	Sistem menyimpan *file* pendukung yang dimasukkan

Alur Normal	1. Pengguna membuka tab *Import File* 2. Sistem membuka tab *Import File* 3. Pengguna memasukkan *dataset* yang berupa *file* Excel 4. Pengguna memasukkan *file* RDFS CIDOC-CRM 5. Pengguna menekan tombol "Next" 6. Sistem membaca entitas dan *data property* pada RDFS CIDOC

Berdasarkan skenario kasus penggunaan pada Tabel 4.3 selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Diagram aktivitas dari kasus penggunaan Memuat *File* Pendukung dapat dilihat pada Tabel 4.3 berikut.



Gambar 4.3 Diagram Aktivitas Memuat *File* Pendukung

4.1.5.2. Kasus Penggunaan Memuat *File* Konfigurasi

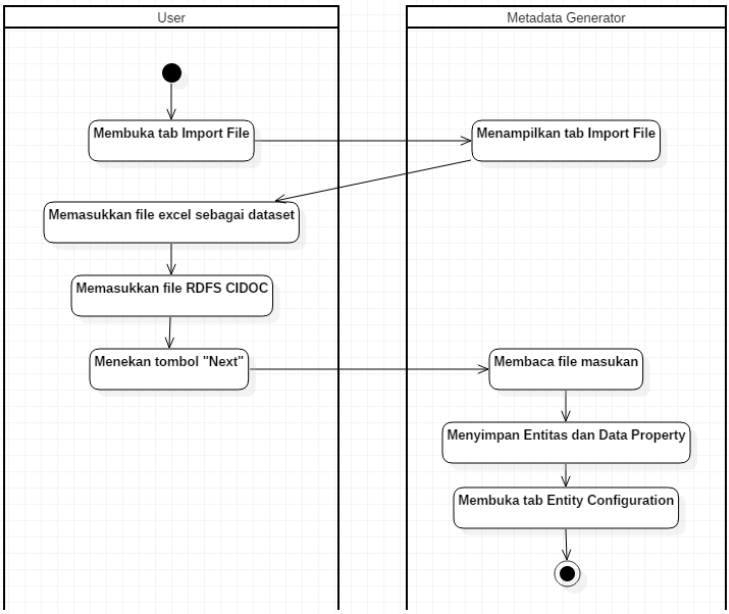
Pada kasus penggunaan memuat *file* konfigurasi, pengguna memasukkan *file* konfigurasi yang berupa *mapping file* dan RDFS *Configuration* yang telah dibuat sebelumnya. *File-file* tersbeut juga merupakan hasil generator dari aplikasi ini. Penjelasan skenario yang lebih rinci dapat dilihat pada Tabel 4.4.

Tabel 4.4 Spesifikasi Kasus Penggunaan Memuat *File* Konfigursai

Nama Kasus Penggunaan	Memuat *File* Konfigurasi
Nomor	UC02
Aktor	Pengguna
Kondisi Awal	Pengguna memiliki *file* konfigurasi yang berupa mapping.txt dan RDFS *Configuration.*
Kondisi Akhir	Sistem menyimipan *file* yang dimasukkan oleh pengguna
Alur Normal	1. Pengguna membuka tab *Import File* 2. Sistem membuka tab *Import File* 3. Pengguna memasukkan *dataset* yang berupa *file* Excel 4. Pengguna mencentang pilihan *load configuration* 5. Pengguna memasukkan *file* RDFS *Configuration* 6. Pengguna menekan tombol “Next” 7. Sistem membaca entitas dan *data property* pada RDFS *Configuration*

Berdasarkan skenario kasus penggunaan pada Tabel 4.4 selanjutnya skenario tersebut digambarkan ke dalam diagram

aktivitas. Diagram aktivitas dari kasus penggunaan Memuat *File* Konfigurasi dapat dilihat pada Gambar 4.4 berikut.



Gambar 4.4 Diagram Aktivitas Memuat *File* Konfigurasi

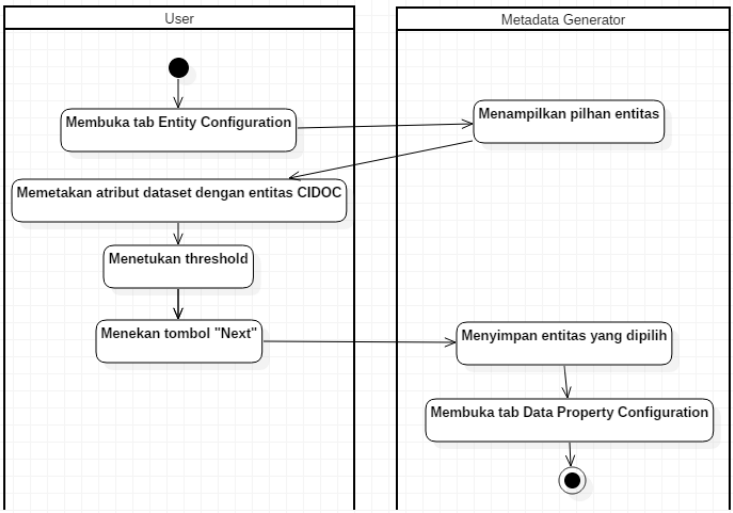
4.1.5.3. Kasus Penggunaan Konfigurasi Entitas

Pada kasus penggunaan konfigurasi entitas, pengguna memilih entitas CIDOC yang akan digunakan dalam proses pembangkitan metadata. Sistem akan menampilkan semua entitas yang terdapat pada RDFS CIDOC. Selanjutnya pengguna memetakan setiap atribut dari dataset dengan entitas CIDOC. Setiap atribut dapat dipetakan dengan entitas yang sama atau berbeda tergantung dari kebutuhan pengguna. Penjelasan skenario yang lebih rinci disajikan pada Tabel 4.5.

Tabel 4.5 Spesifikasi Kasus Penggunaan Konfigurasi Entitas

Nama Kasus Penggunaan	Konfigurasi Entitas
Nomor	UC03
Aktor	Pengguna
Kondisi Awal	Sistem menampilkan semua entitas CIDOC-CRM pada setiap atribut *dataset*
Kondisi Akhir	Sistem menyimpan entitas yang dipilih oleh pengguna
Alur Normal	1. Pengguna membuka tab *Entity Configuration* 2. Sistem menampilkan tab *Entity Configuration* 3. Pengguna memilih entitas CIDOC unutk setiap atribut *dataset* 4. Pengguna menentuka *threshold* 5. Pengguna menekan tombol “Next” 6. Sistem menyimpan entitas yang dipilih 7. Sistem membuka tab *Data Property Configuration*
Alur Alternatif	-

Berdasarkan skenario kasus penggunaan pada Tabel 4.5 selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Diagram aktivitas dari kasus penggunaan Konfigurasi Entitas dapat dilihat pada Gambar 4.5.



Gambar 4.5 Diagram Aktivitas Konfigurasi Entitas

4.1.5.4. Kasus Penggunaan Konfigurasi *Object Property*

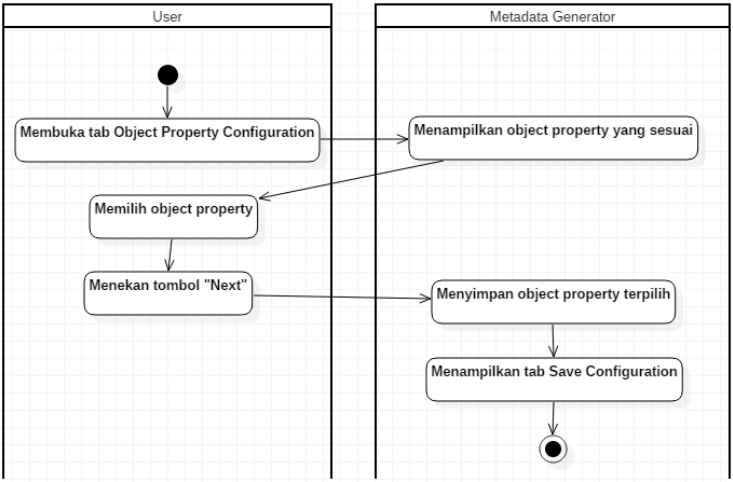
Pada kasus penggunaan Konfigurasi *Object Property*, pengguna harus menentukan *object property* apa yang akan dilibatkan dalam proses pembangkitan metadata. Sistem akan menampilkan daftar *object property* yang sesuai dengan entitas terpilih. Selanjutnya pengguna memilih *object property* sesuai dengan kebutuhan. Penjelasan skenario yang lebih rinci dapat dilihat pada Tabel 4.6.

Tabel 4.6 Spesifikasi Kasus Penggunaan Konfigurasi *Object Property*

Nama Kasus Penggunaan	Konfigurasi *Object Property*
Nomor	UC04
Aktor	Pengguna

Kondisi Awal	Sistem menampilkan daftar *object property* yang sesuai dengan entitas terpilih
Kondisi Akhir	Sistem menyimpan daftar *object property* yang dipilih pengguna
Alur Normal	1. Pengguna membuka tab *Object Property Configuration* 2. Sistem menampilkan tab *Entity Configuration* 3. Sistem menampilkan daftar *object property* yang sesuai dengan entitas terpilih 4. Pengguna memilih *object property* 5. Pengguna menekan tombol "Next" 6. Sistem menyimpan *object property* yang dipilih 7. Sistem menampilkan tab *Save Configuration*

Berdasarkan skenario kasus penggunaan pada Tabel 4.6 selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Diagram aktivitas dari kasus penggunaan Memuat *File* Konfigurasi dapat dilihat pada Gambar 4.6.



Gambar 4.6 Diagram Aktivitas Konfigurasi *Object Property*

4.1.5.5. Kasus Penggunaan Menyimpan Konfigurasi

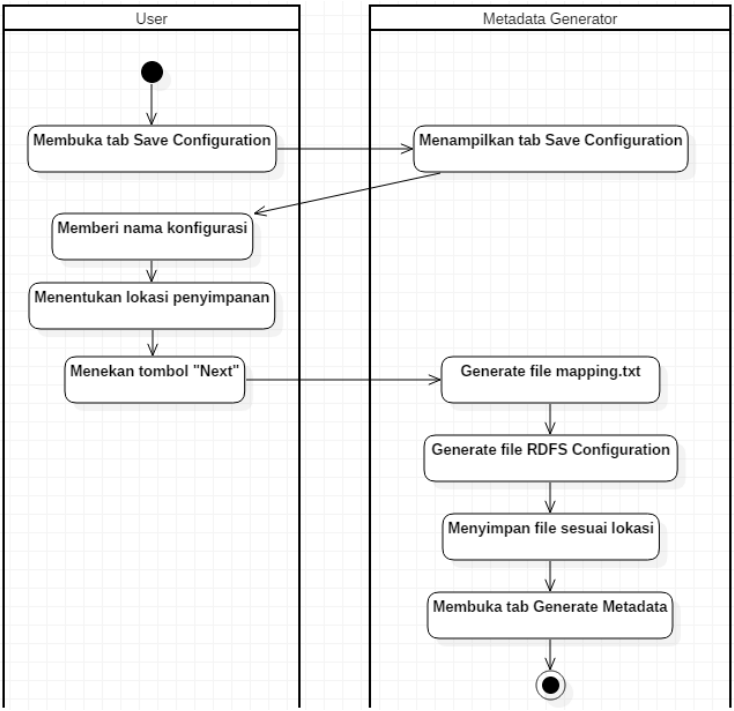
Pada kasus penggunaan Menyimipan Konfigurasi, pengguna akan menyimpan konfigurasi yang telah dilakukan. Konfigurasi tersebut disimpan dalam dua buah *file* yaitu *mapping file* dan RDFS *Configuration. Mapping file* merupakan konfigurasi pemetaan atribut *dataset* dengan entitas CIDOC, sedangkan RDFS *Configuration* berisi deskripsi dari entitas dan *object property* yang telah dipilih oleh pengguna. Untuk menyimpan konfigurasi pengguna diharuskan menentukan lokasi penyimpanan terlebih dahulu. Penjelasan skenario yang lebih rinci dapat dilihat pada Tabel 4.7.

Tabel 4.7 Spesifikasi Kasus Penggunaan Menyimpan Konfigurasi

Nama Kasus Penggunaan	Menyimpan konfigurasi
Nomor	UC05
Aktor	Pengguna

Kondisi Awal	Konfigurasi pengguna tersimpan pada sistem dalam bentuk variabel
Kondisi Akhir	Konfigurasi pengguna tersimpan pada sistem dalam bentuk *file* keluaran
Alur Normal	1. Pengguna membuka tab *Save Configuration* 2. Sistem menampilkan tab *Save Configuration* 3. Pengguna memberi nama konfigurasi 4. Pengguna menentukan lokasi penyimpanan 5. Pengguna menekan tombol “Next” 6. Sistem men-*generate file* mapping.txt dan RDFS *Configuration* 7. Sistem menampilkan tab *Generate* Metadata

Berdasarkan skenario kasus penggunaan pada Tabel 4.7 selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Diagram aktivitas dari kasus penggunaan Memuat *File* Konfigurasi dapat dilihat pada Gambar 4.7.



Gambar 4.7 Diagram Aktivitas Menyimpan Konfigurasi

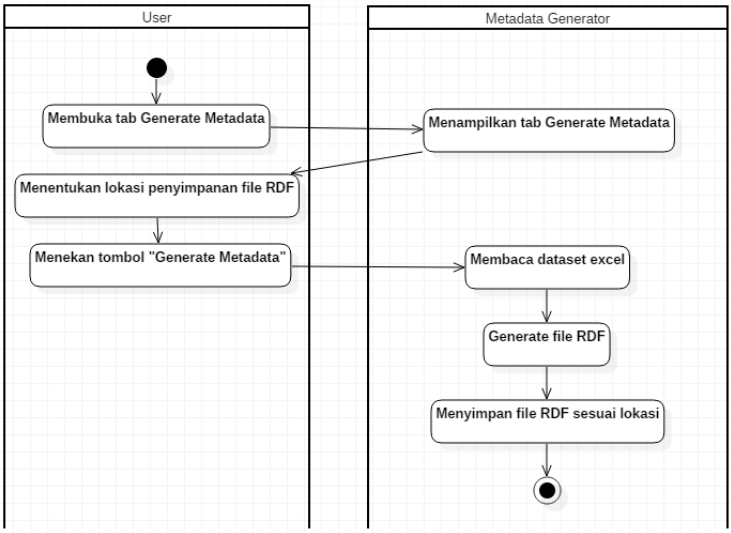
4.1.5.6. Kasus Penggunaan *Generate* Metadata

Pada kasus penggunaan *Generate* Metadata pengguna harus menentukan lokasi untuk menyimpan *file* RDF. Selanjutnya sistem akan menghasilkan *file* RDF/XML untuk setiap data artefak pada dataset Excel dan meletakkannya pada lokasi yang telah ditentukan. Penjelasan skenario yang lebih rinci dapat dilihat pada Tabel 4.8.

Tabel 4.8 Spesifikasi Kasus Penggunaan *Generate* Metadata

Nama Kasus Penggunaan	*Generate* Metadata
Nomor	UC06
Aktor	Pengguna
Kondisi Awal	Sistem telah memiliki konfigurasi untuk membangun metadata
Kondisi Akhir	Sistem men-*generate file* RDF/XML sesuai *dataset* dan konfigurasi
Alur Normal	1. Pengguna membuka tab *Generate* Metadata 2. Sistem menampilkan tab *Generate* Metadata 3. Pengguna menentukan lokasi penyimpanan 4. Pengguna menekan tombol “Next” 5. Sistem men-*generate file* RDF/XML 6. Sistem menyimpan *file* RDF/XML sesuai lokasi yang ditentukan.

Berdasarkan skenario kasus penggunaan pada Tabel 4.8 selanjutnya skenario tersebut digambarkan ke dalam diagram aktivitas. Diagram aktivitas dari kasus penggunaan Memuat *File* Konfigurasi dapat dilihat pada Gambar 4.8.



Gambar 4.8 Diagram Aktivitas *Generate* Metadata

4.2. Perancangan Sistem

Pada subbab ini dijelaskan mengenai tahapan perancangan sistem. Perancangan sistem ini dibagi menjadi beberapa bagian yang meliputi perancangan data, perancangan arsitektur sistem, perancangan proses aplikasi, dan perancangan antarmuka pengguna.

4.2.1. Perancangan Data

Pada Tugas Akhir ini data yang digunakan sebagai *dataset* adalah data artefak warisan budaya dari Museum Indonesia. Data tersebut berisi informasi tentang artefak yang ditemukan diseluruh wilayah Indonesia. Dalam *file* tersebut terdapat beberapa atribut antara lain: No.Urut, No.InvB, No_FotoGab, Nama_Benda, Tempat_Asal_Des_Kec, Des_Gabungan1, Kondisi, Lokasi_Gedung1, Lokasi Ruang1, Lokasi Lemari_Laci1, Lokasi_Lain1, Tempat_Asal_Kab, Tempat_Asal_Prop, Tempat Pembuatan, Tempat Perolehan, Tahun Pembuatan. Penamaan

atribut pada dataset tersebut kurang proporsional untuk digunakan sebagai dataset Tugas Akhir ini. Terdapat pengulangan data dan beberapa atribut yang menampung lebih dari satu informasi. Sehingga perlu dilakukan perubahan pada atribut untuk disesuaikan dengan sistem yang dibangun. Berikut beberapa perubahan yang dilakukan untuk menyesuaikan dataset.

- Menghapus No.InvB dan No_FotoGab dan menjadikan No Urut sebagai identitas unik dari setiap artefak.
- Menghilangkan atribut Des_Gabungan1, yang merupakan deskripsi gabungan dari setiap artefak. Pada atribut ini dideskripsikan beberap informasi mengenai artefak terkait dalam satu kolom. Sedangkan terdapat beberapa atribut yang telah dideskripsikan pada kolom atribut lain sehingga terjadi pengulangan data. Selain itu ada beberapa informasi mengenai dimensi artefak seperti panjang, lebar, tinggi, dan tebal yang perlu dideskripsikan secara terpisah. Sehingga ditambahkan beberapa atribut baru untuk menampung informasi dimensi artefak seperti: Ukuran panjang, ukuran lebar, ukuran tinggi, ukuran tebal, dan ukuran dimensi.
- Mendeskripsikan data kosong dengan karakter kosong. Data kosong pada dataset ini dideskripsikan dengan karakter ‘ - ’ dan karakter kosong. Dalam Tugas Akhir ini data kosong digeneralisasi dengan karakter kosong.
- Memperbaiki nama atribut menjadi lebih formal.

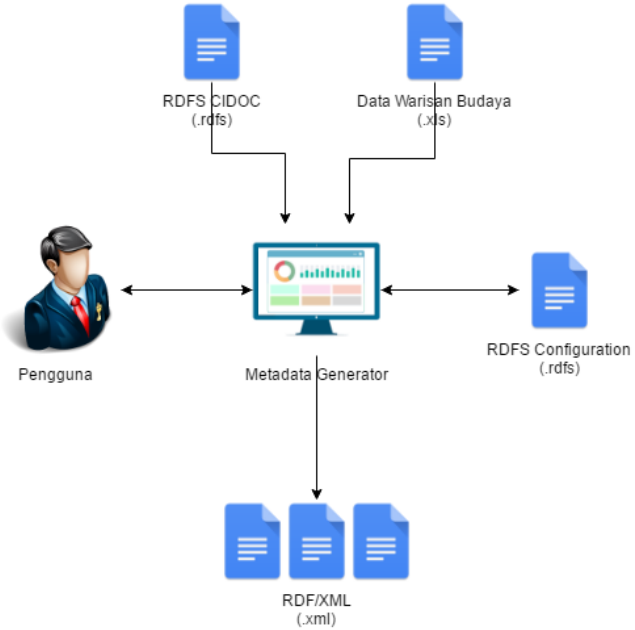
Pada akhir proses penyesuaian didapatkan 19 nama atribut yang digunakan dalam proses pembangkitan metadata. Penjelasan dari setiap atribut disajikan pada Tabel 4.9 berikut.

Tabel 4.9 Deskripsi Atribut Dataset

No	Nama Atribut	Penjelasan
1	No Urut	Berisi nomor urut penemuan artefak. Pada Tugas Akhir ini dijadikan sebagai identitas unik dari masing-masing artefak.
2	Nama Benda	Nama artefak
3	Tempat Asal Des/Kec	Desa atau kecamatan tempat ditemukannya artefak
4	Tempat Asal Kab	Kabupaten tempat ditemukannya artefak
5	Tempat Asal Prop	Proponsi tempat ditemukannya artefak
6	Lokasi Gedung	Lokasi gedung tempat penyimpanan artefak saat ini
7	Lokasi Ruang	Lokasi ruang tempat penyimpanan artefak saat ini
8	Lokasi Lemari/Laci	Lokasi lemari/laci tempat penyimpanan artefak saat ini
9	Lokasi Lain	Lokasi lain tempat penyimpanan artefak saat ini
10	Kondisi	Kondisi artefak
11	Bahan	Bahan yang digunakan untuk membuat artefak
12	Ukuran Panjang	Ukuran panjang artefak
13	Ukuran Lebar	Ukuran lebar artefak
14	Ukuran Tinggi	Ukuran tinggi artefak
15	Ukuran Tebal	Ukuran tebal artefak
16	Ukuran Diameter	Ukuran diameter artefak
17	Tempat Perolehan	Tempat perolehan artefak
18	Tempat Pembuatan	Perkiraan tempat pembuatan artefak
19	Tahun Pembuatan	Perkiraan bahan pembuatan artefak

4.2.2. Perancangan Arsitektur

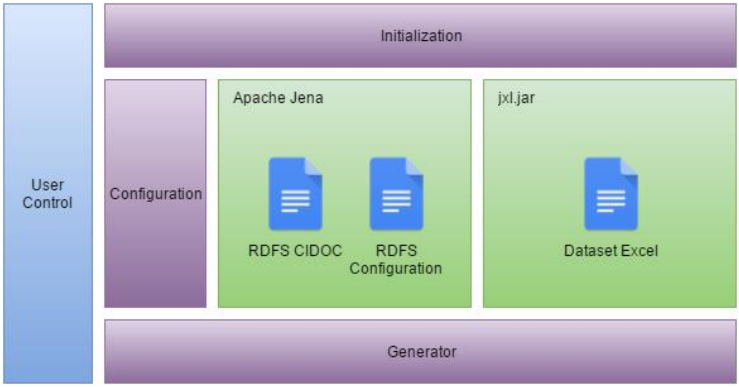
Aplikasi yang akan dibangun merupakan aplikasi berbasis *dekstop* dan dibangun menggunakan bahasa pemrograman Java. Untuk mendukung proses pembangkitan metadata digunakan *framework* Jena yang merupakan salah satu *framework* khusus untuk pembangunan metadata. Proses pembacaan dataset Excel dilakukan dengan bantuan *library* jlx.jar yang merupakan *library* khusus untuk pengolahan *file* Excel. Rancangan arsitektur sistem dapat dililhat pada Gambar 4.9 berikut.



Gambar 4.9 Lingkungan Sistem

Berdasarkan Gambar 4.9 dapat dilihat bahwa sistem membutuhkan masukan berupa *file* RDFS CIDOC yang berekstensi .rdfs dan Dataset Wariasan Budaya yang berekstensi .xls. Setelah melalui proses yang ada sistem akan

menghasilkan *file* RDFS *Configuration* yang berekstensi .rdfs. *File* ini juga bisa dijadikan sebagai masukan untuk memuat konfigurasi. Selain itu sistem juga menghasilkan kumpulan *file* RDF/XML yang merupakan hasil pembangkitan metadata warisan budaya.



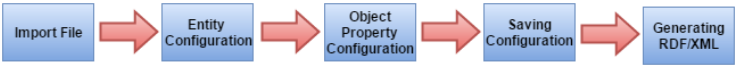
Gambar 4.10 Arsitektur Metadata Generator

Pada Gambar 4.10 dapat dilihat bahwa *User Control* dapat mengakses proses *Initialization, Configuration, dan Generator. Initialization* adalah proses pemasukan *file-file* yang dibutuhkan oleh sistem seperti *dataset,* RDFS CIDOC, dan RDFS *Configuration. Configuration* adalah proses pengaturan entitas dan *Object Property* yang dilakukan oleh pengguna sesuai dengan kebutuhan. *Generator* adalah proses pembangkitan metadata warisan budaya yang hasilnya disimpan dalam entuk *file* RDF/XML.

Ketiga proses tersebut (*Initialization, Configuration, dan Generator*) membutuhkan bantuan dari *framewrok* Apache Jena dan *library* jxl.jar. *Framework* Apache Jena digunakan untuk membantu proses pembacaan entitas dan *Object Property*. Sedangkan *library* jxl.jar digunakan untuk membantu proses pembacaan *dataset* pada *file* Excel.

4.2.3. Perancangan Proses Aplikasi

Secara garis besar proses pembangkitan metadata pada aplikasi ini ditunjukkan pada Gambar 4.11.



Gambar 4.11 Proses Pembangkitan Metadata

Dari Gambar 4.11 di atas dapat dilihat bahwa dalam men-*generate* metadata dibutuhkan lima proses utama: *Import File, Entity Configuration, Object Property Configuration, Saving Configuration,* dan *Generating RDF/XML*. Detail dari setiap proses dijelaskan sebagai berikut.

1) *Import File*

Pada tahap ini sistem melakukan inisialisasi terhadap *resources* yang dibutuhkan dalam proses pembangkitan metadata. *Resources* tersebut berupa *file* RDFS CIDOC dan *file* Excel yang berisi data artefak Museum Indonesia. Pengguna juga dapat memasukkan *file* konfigurasi, yang terdiri dari *mapping file* dan RDFS *Configuration,* yang telah dibangun sebelumnya.

2) Entity *Configuration*

Pada tahap ini dilakukan pemilihan terhadap entitas CIDOC yang akan digunakan dalam proses pembangkitan metadata. Sistem akan menampilkan semua entitas yang terdapat pada RDFS CIDOC. Selanjutnya pengguna akan memetekan setiap atribut dari *dataset* dengan entitas CIDOC. Setiap atribut dapat dipetakan dengan entitas yang sama atau berbeda tergantung dari kebutuhan pengguna.

Pada tahap ini juga diatur tingkat *threshold* yang merupakan batas kedalaman dari entitas. Kedalam entitas yang dimaksud adalah seberapa jauh sistem menelusuri *parent* dari entitas yang dipilih. Semakin tinggi nilai *threshold* maka entitas yang terlibat akan semakin banyak.

3) *Object Property Configuration*

Pada tahap ini pengguna harus menentukan *object property* apa yang akan dilibatkan dalam proses pembangkitan metadata. Sistem akan menampilkan daftar *object property* yang sesuai dengan entitas terpilih. Selanjutnya pengguna memilih *object property* sesuai dengan kebutuhan.

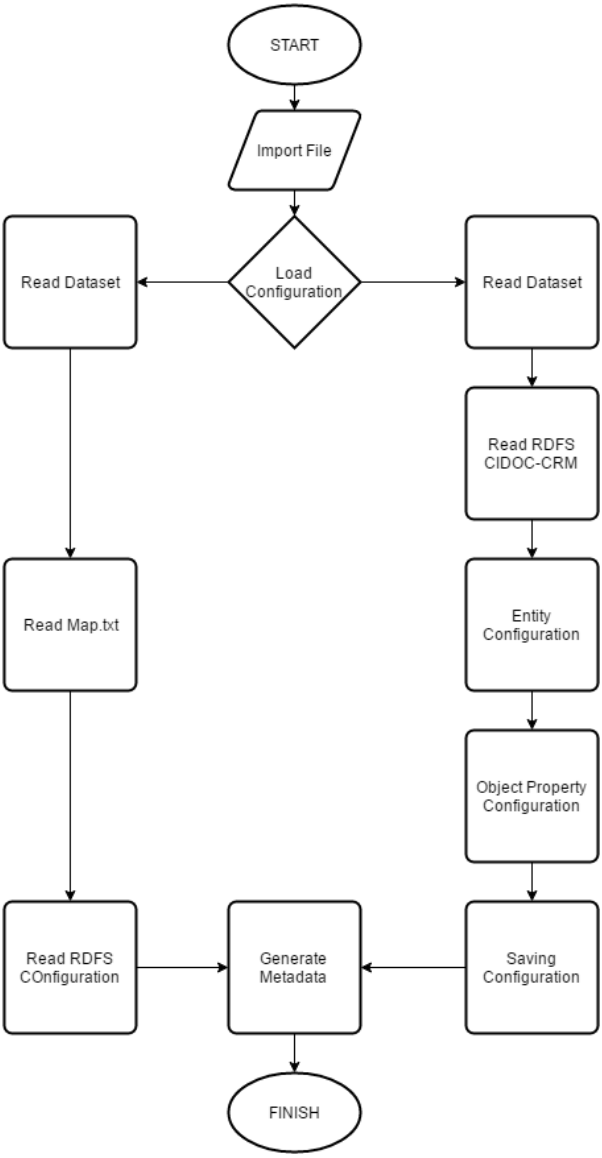
4) *Saving Configuration*

Pada tahap ini sistem akan menyimpan konfigurasi yang telah dilakukan oleh pengguna. Konfigurasi tersebut disimpan dalam dua buah *file* yaitu *mapping file* dan RDFS *Configuration. Mapping file* merupakan konfigurasi pemetaan atribut *dataset* dengan entitas CIDOC, sedangkan RDFS *Configuration* berisi deskripsi dari entitas dan *object property* yang telah dipilih oleh pengguna.

5) *Generating* RDF/XML

Tahap terakhir dari deretan proses pembangkitan metadata adalah pembangkitan *file* RDF/XML dari *dataset* Excel berdasarkan konfigurasi yang telah dilakukan oleh pengguna. Sebelumnya pengguna harus menentukan lokasi dari *file* tersebut. Selanjutnya sistem akan menghasilkan *file* RDF/XML untuk setiap data artefak pada *dataset* Excel dan meletakkanya pada lokasi yang telah ditentukan.

Kelima proses tersebut dijalankan secara berurutan sampai menghasilkan *file* metadata warisan budaya. Apabila digambarkan dalam bentuk diagram alir akan nampak seperti Gambar 4.12 berikut.



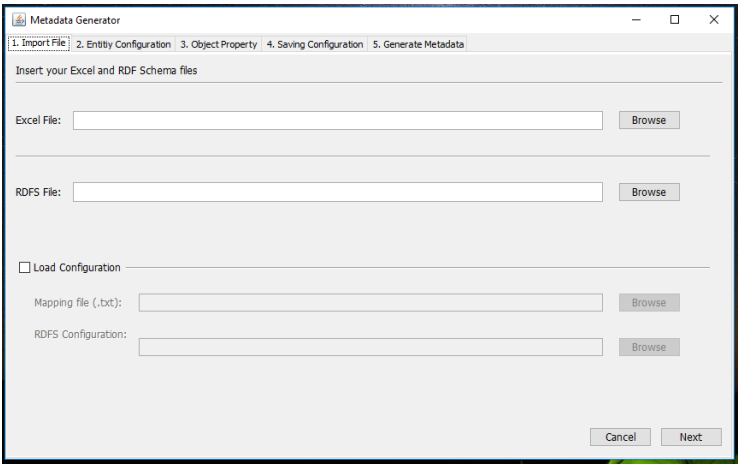
Gambar 4.12 Diagram Alir Proses Aplikasi

4.2.4. Perancangan Antarmuka

Pada bagian ini dijelaskan mengenai rancangan tampilan antarmuka pengguna dari sistem. Antarmuka pengguna dalam sistem ini terdiri dari lima bagian sesuai dengan proses yang ada yaitu *Import File*, *Entity Configuration*, *Object Property Configuration*, *Saving Configuration*, dan *Generating RDF/XML*. Berikut akan dijelaskan masing-masing rancangan antarmuka tersebut.

4.2.4.1. Antarmuka Halaman *Import File*

Pada tahap ini pengguna memasukkan *file-file* yang dibutuhkan oleh sistem yaitu *dataset*, RDFS CIDOC, dan RDFS *Configuration* dan Mapping.txt. Tampilan halaman *Import File* dapat dilihat pada Gambar 4.13 berikut.



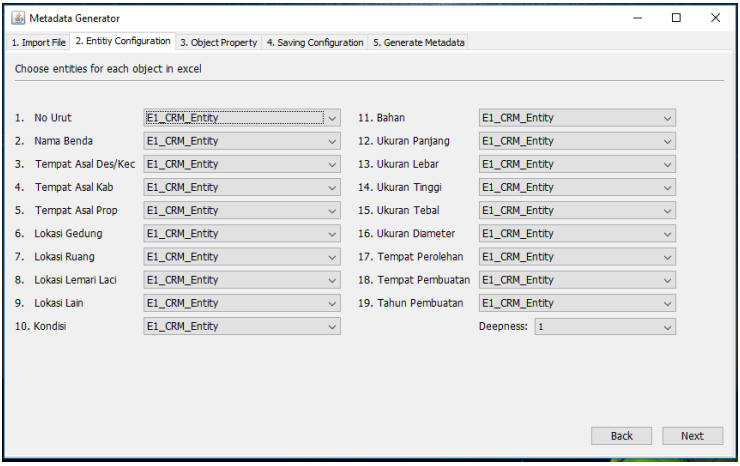
Gambar 4.13 Rancangan Antarmuka Halaman *Import File*

Kolom Excel *file* diisi dengan lokasi dari *file dataset* yang akan digunakan. Pengguna dapat mencari lokasi dengan menekan tombol *Browse* disampingnya. Kolom RDFS *File* diisi dengan lokasi dari *file* RDFS CIDOC.

Pengguna juga dapat memuat konfigurasi yang sudah ada dengan mengisi centang pada *Load Configuration.* Jika proses ini dipilih maka pengguna harus memasukkan *mapping file* dan RDFS *Configuration* sebagai konfigurasi.

4.2.4.2. Antarmuka Halaman Entity *Configuration*

Pada tahap ini pengguna melakukan konfigurasi entitas dengan memetakan setiap atribut pada *dataset* terhadap entitas yang ada sesuai dengan kebutuhan. Pada tahap ini pengguna juga menentukan tingkat *deepness/threshold* untuk proses pembangkitan metadata. Tampilan halaman untuk proses *Entity Configuration* dapat dilihat pada Gambar 4.14 berikut.

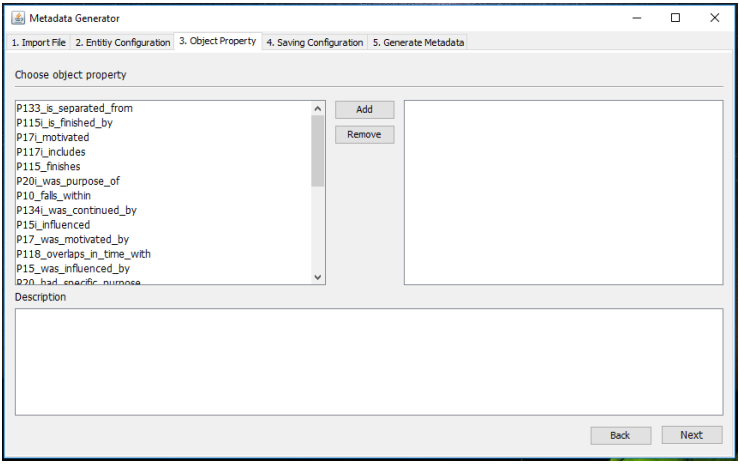


Gambar 4.14 Rancangan Antarmuka Halaman Entity *Configuration*

Pada Gambar 4.14 dapat dilihat bahwa daftar entitas disajikan dalam bentuk *dropdown list*. Hal ini dilakukan untuk memudahkan pengguna dalam memilih entitas di setiap atribut *dataset.*

4.2.4.3. Antarmuka Halaman *Object Property Configuration*

Pada tahap ini pengguna memilih *objetc property* apa saja yang akan digunakan dalam proses pembangkitan metadata. Tampilan halaman untuk proses ini dapat dilihat pada Gambar 4.15 berikut.

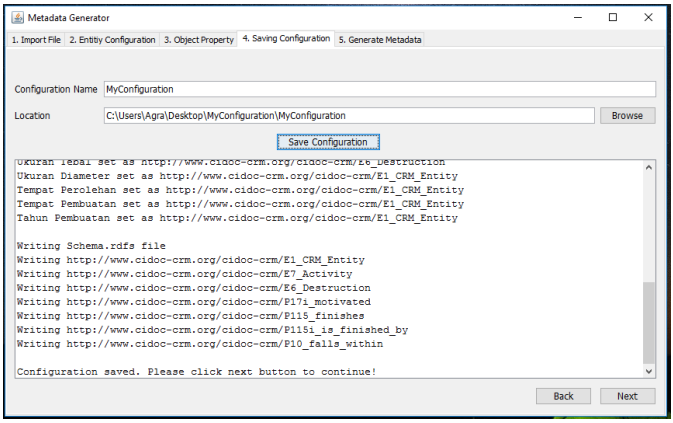


Gambar 4.15 Rancangan Antarmuka Halaman *Object Property Configuration*

Sistem akan menampilkan daftar *Object Property* yang tersedia sesuai dengan pilihan entitas pada kolom sebelah kiri. Pengguna akan memilih salah satu daftar dan menambahkan ke kolom sebelah kanan sebagai daftar terpilih.

4.2.4.4. Antarmuka Halaman *Saving Configuration*

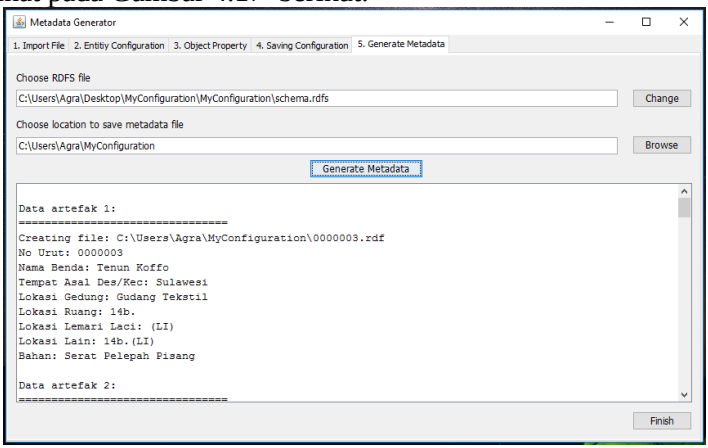
Pada tahap ini pengguna akan menyimpan konfigurasi yang telah dilakukan dengan menentukan nama dan lokasi penyimpanan konfigurasi. Tampilan halaman *Saving Configuration* dapat dilihat pada Gambar 4.16 berikut.



Gambar 4.16 Rancangan Antarmuka Halaman *Saving Configuration*

4.2.4.5. Antarmuka Halaman *Generating* RDF/XML

Pada tahap ini sistem akan men-*generate file* metadata dalam bentuk RDF/XML sesaui konfigurasi dan isi *dataset.* Pengguna diharuskan menentukan lokasi penyimpanan *file-file* tersebut. Tampilan halaman proses *Generating RDF/XML* dapat dilihat pada Gambar 4.17 berikut.



Gambar 4.17 Rancangan Antarmuka Halaman *Generating* RDF/XML

4.2.5. Perancangan Kelas

Pada Tugas Akhir ini dibutuhkan beberapa kelas baru untuk menunjang jalannya proses pembangkitan metadata warisan budaya seperti *MetadataGenerator, ExcelHandler, RDFHandler, Entity, dan ObjProperty*. Kelas-kelas tersebut dan fungsinya disajikan dalam Tabel 4.10 berikut.

Tabel 4.10 Kelas dan Fungsinya

No	Nama Kelas	Penjelasan	*Include*
1	MetadataGenerator.java	Merupakan kelas utama yang berisi main function. Kelas ini adalah yang pertama kali dijalankan ketika mamulai program.	
2	ExcelHandler.java	Berperan dalam proses pembacaan dataset Excel. Kelas ini berisi fungsi-fungsi untuk membaca setiap baris pada *file* Excel.	*Library jxl.jar* untuk proses pembacaan *dataset* Excel
3	RDFHandler.java	Berisi fungsi-fungsi untuk proses pembacaan entitas dan *object property* pada RDFS CIDOC dan RDFS *Configuration.*	Apache Jena RIOT untuk proses pembacaan entitas dan *object property*
4	Entity.java	Merupakan sebuah kelas untuk mendeskripsikan Entitas.	

5	ObjProperty.java	Merupakan sebuah kelas untuk mendeskripsikan *Object Property*.	

Sedangkan hubungan dari setiap kelas yang telah disebutkan pada Tabel 4.10 disajikan dalam bentuk diagram kelas pada Lampiran 2.

(Halaman ini sengaja dikosongkan)

BAB V
IMPLEMENTASI

Pada bab ini akan dibahas implementasi dari rancangan sistem yang telah dianalisis dan dirancang dalam Bab IV. Implementasi ini meliputi implementasi kelas-kelas beserta fungsi yang ada di dalamnya, dan implementasi antarmuka pengguna.

5.1. Lingkungan Implementasi

Lingkungan pengembangan sistem yang digunakan untuk mengembangkan Tugas Akhir ini dilakukan pada lingkungan perangkat keras dan lingkungan perangkat lunak. Perangkat keras yang digunakan dalam pengembangan sistem berupa laptop dengan spesifikasi sebagai berikut:

- Prosesor: Intel(R) Core (TM) i5-6200U CPU @ 2.30GHz (4 CPUs), ~2.4GHz.
- Memori: 4096MB RAM.

Sedangkan spesifikasi perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut:

- Sistem Operasi Windows 10 Professional 64 bit
- Java versi 1.8.
- Netbeans sebagai IDE untuk implementasi aplikasi.
- *Framework* Apache Jena untuk proses pembacaan isi dari *file* RDF.
- *Library* jlx.jar versi 2.6.3 untuk proses pembacaan data Excel.
- StarUML 5 untuk merancang diagram kasus penggunaan.
- Protege 4.3 untuk pengujian validasi metadata.

5.2. Implementasi Data

Data yang digunakan adalah data warisan budaya dari Museum Indonesia yang berupa *file* Excel. Dalam *file* tersebut ada beberapa atribut antara lain: No.Urut, No.InvB, No_FotoGab, Nama_Benda, Tempat_Asal_Des_Kec, Des_Gabungan1, Kondisi, Lokasi_Gedung1, Lokasi Ruang1, Lokasi Lemari_Laci1, Lokasi_Lain1, Tempat_Asal_Kab, Tempat_Asal_Prop, Tempat Pembuatan, Tempat Perolehan, Tahun Pembuatan. Data tersebut diolah terlebih dahulu untuk disesuaikan dengan kebutuhan sistem. Setelah prosses penyesuaian didapatkan 19 nama atribut yang digunakan dalam proses pembangkitan metadata. Penjelasan dari setiap atribut disajikan pada Tabel 5.1 berikut.

Tabel 5.1 Deskripsi Atribut Dataset

No	Nama Atribut	Penjelasan
1	No Urut	Berisi nomor urut penemuan artefak. Pada Tugas Akhir ini dijadikan sebagai identitas unik dari masing-masing artefak.
2	Nama Benda	Nama artefak
3	Tempat Asal Des/Kec	Desa atau kecamatan tempat ditemukannya artefak
4	Tempat Asal Kab	Kabupaten tempat ditemukannya artefak
5	Tempat Asal Prop	Proponsi tempat ditemukannya artefak
6	Lokasi Gedung	Lokasi gedung tempat penyimpanan artefak saat ini
7	Lokasi Ruang	Lokasi ruang tempat penyimpanan artefak saat ini
8	Lokasi Lemari/Laci	Lokasi lemari/laci tempat penyimpanan artefak saat ini
9	Lokasi Lain	Lokasi lain tempat penyimpanan artefak saat ini

10	Kondisi	Kondisi artefak
11	Bahan	Bahan yang digunakan untuk membuat artefak
12	Ukuran Panjang	Ukuran panjang artefak
13	Ukuran Lebar	Ukuran lebar artefak
14	Ukuran Tinggi	Ukuran tinggi artefak
15	Ukuran Tebal	Ukuran tebal artefak
16	Ukuran Diameter	Ukuran diameter artefak
17	Tempat Perolehan	Tempat perolehan artefak
18	Tempat Pembuatan	Perkiraan tempat pembuatan artefak
19	Tahun Pembuatan	Perkiraan bahan pembuatan artefak

5.3. Implementasi Proses Aplikasi

Pada subbab ini akan dibahas implementasi dari proses yang berjalan pada sistem metadata generator. Ada lima tahapan proses yang dilakukan oleh sistem pada tugas akhir ini, yaitu: *Import File, Entity Configuration, Object Property Configuration, Saving Configuration,* dan *Generating RDF/XML*. Implementasi pada setiap tahapnya akan dijelaskan pada subbab ini.

5.3.1. Implementasi Proses *Import File*

Proses *Import File* adalah proses pemilihan *file* yang dibutuhkan dalam proses pembangkitan metadata. Proses ini terbagi menjadi beberapa tahapan yang berurutan sebagai berikut.

- Tahap pertama adalah proses pemilihan *file* Excel dan *file* RDFS CIDOC. Hasil implementasi terhadap proses ini disajikan pada Kode Sumber 5.1 dan Kode Sumber 5.2.
- Tahap kedua adalah proses memuat konfigurasi yang bersifat optional. Hasil implementasi terhadap proses ini disajikan pada Kode Sumber 5.3, Kode Sumber 5.4, dan Kode Sumber 5.5.

- Selanjutnya adalah proses pembacaan data Entitas dan *Object Property* pada RDFS CIDOC. Proses ini diakhiri dengan berpindahnya tab ke Entity *Configuration*. Hasil implementasi terhadap proses ini disajikan pada Kode Sumber 5.6.

```
private void
btnBrowseExcelActionPerformed(java.awt.event.Actio
nEvent evt) {
        JFileChooser chooser = new JFileChooser();

        if(ExcelDir == null){
            chooser.setCurrentDirectory(new
File(System.getProperty("pengguna.home")));
        }else {
            chooser.setCurrentDirectory(ExcelDir);
        }

        chooser.setFileFilter(new
FileNameExtensionFilter("Excel
Files","xls","xlsx"));
        int result = chooser.showOpenDialog(this);

        if (result ==
JFileChooser.APPROVE_OPTION){

tfExcelFile.setText(chooser.getSelectedFile().getP
ath());
            ExcelDir =
chooser.getCurrentDirectory();
            ExcelFile = chooser.getSelectedFile();
        }
    }
```

Kode Sumber 5.1 Fungsi Untuk Mencari *File* Excel

```
private void
btnBrowserdfActionPerformed(java.awt.event.ActionEv
ent evt) {
        JFileChooser chooser = new JFileChooser();

        if(ExcelDir == null){
            chooser.setCurrentDirectory(new
File(System.getProperty("pengguna.home")));
        }else {
            chooser.setCurrentDirectory(ExcelDir);
        }

        chooser.setFileFilter(new
FileNameExtensionFilter("Resource Description
Framework Schema","rdfs"));
        int result = chooser.showOpenDialog(this);

        if (result == JFileChooser.APPROVE_OPTION){

tfRdfFile.setText(chooser.getSelectedFile().getPath
());
            rdfDir = chooser.getCurrentDirectory();
            rdfFile = chooser.getSelectedFile();
        }
    }
```

Kode Sumber 5.2 Fungsi Untuk Mencari *File* RDFS CIDOC

```
private void
jCheckBox1ActionPerformed(java.awt.event.ActionEven
t evt) {
        jCheckBox1.addChangeListener(new
ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e)
{
if(jCheckBox1.isSelected()){

btnBrowserdf.setEnabled(Boolean.FALSE);
tfRdfFile.setEnabled(Boolean.FALSE);
labelRdfFile.setEnabled(Boolean.FALSE);
jButtonBrowseConf.setEnabled(Boolean.TRUE);
jButtonBrowseMapping.setEnabled(Boolean.TRUE);
tfRDFSConf.setEnabled(Boolean.TRUE);
```

```
tfMapping.setEnabled(Boolean.TRUE);
labelMapping.setEnabled(Boolean.TRUE);
labelRDFSConf.setEnabled(Boolean.TRUE);
loadConfiguration = Boolean.TRUE;
}
else{

btnBrowserdf.setEnabled(Boolean.TRUE);
tfRdfFile.setEnabled(Boolean.TRUE);
labelRdfFile.setEnabled(Boolean.TRUE);
jButtonBrowseConf.setEnabled(Boolean.FALSE);
jButtonBrowseMapping.setEnabled(Boolean.FALSE);
tfRDFSConf.setEnabled(Boolean.FALSE);
tfMapping.setEnabled(Boolean.FALSE);
labelMapping.setEnabled(Boolean.FALSE);
labelRDFSConf.setEnabled(Boolean.FALSE);
loadConfiguration = Boolean.FALSE;
}
}
});
}
```

Kode Sumber 5.3 Fungsi Untuk Centang *Load Configuration*

```
private void
jButtonBrowseMappingActionPerformed(java.awt.event.
ActionEvent evt) {
       JFileChooser chooser = new JFileChooser();

        if(mappingFile == null){
            chooser.setCurrentDirectory(new
File(System.getProperty("pengguna.home")));
        }else {

chooser.setCurrentDirectory(mappingFile);
        }

        chooser.setFileFilter(new
FileNameExtensionFilter("Text Files","txt"));
        int result = chooser.showOpenDialog(this);

        if (result == JFileChooser.APPROVE_OPTION){
```

```
tfMapping.setText(chooser.getSelectedFile().getPath
());
            mappingFile =
chooser.getCurrentDirectory();
        }
    }
```

Kode Sumber 5.4 Fungsi Untuk Memilih *Mapping File*

```
private void
jButtonBrowseConfActionPerformed(java.awt.event.Act
ionEvent evt) {
        JFileChooser chooser = new JFileChooser();

        if(rdfsConfFile == null){
            chooser.setCurrentDirectory(new
File(System.getProperty("pengguna.home")));
        }else {

chooser.setCurrentDirectory(rdfsConfFile);
        }

        chooser.setFileFilter(new
FileNameExtensionFilter("RDFS Files","rdfs"));
        int result = chooser.showOpenDialog(this);

        if (result == JFileChooser.APPROVE_OPTION){

tfRDFSConf.setText(chooser.getSelectedFile().getPat
h());

            rdfsConfFile =
chooser.getCurrentDirectory();
        }
    }
```

Kode Sumber 5.5 Fungsi Untuk Memilih RDFS *Configuration*

```
private void
btnNext1ActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
         // Membaca file Excel
        ExcelHandler = new ExcelHandler();
        System.out.println("Loading file " +
ExcelFile.toString() + "...");

ExcelHandler.setExcelFile(ExcelFile.toString());
        try {
            ExcelHandler.readEntity();
        } catch (IOException ex) {

Logger.getLogger(MetadaGeneratorGUI.class.getName()
).log(Level.SEVERE, null, ex);
        }
        header = ExcelHandler.getHeader();

        if (loadConfiguration){

            // Membaca File RDF
            rdfHandler = new RDFHandler();
            System.out.println("Loading RDFS
Configuration file: " + tfRDFSConf.getText());

rdfHandler.setRdfFile(tfRDFSConf.getText());
            rdfHandler.readFile();

            // Membaca kelas pada RDF
            rdfHandler.readClass();
            listEntity =
rdfHandler.getListEntity();

            Comparator<Entity> comparator = (Entity
o1, Entity o2) ->
Integer.parseInt(o1.getId().substring(1))-
Integer.parseInt(o2.getId().substring(1));

Collections.sort(listEntity,comparator);

            File txtFile;
            txtFile = new
File(tfMapping.getText());
            try {
                Scanner input = new
Scanner(txtFile);
```

```
                System.out.println("Loading Mapping
file: " + tfMapping.getText());
                Entity ent = new Entity();
                String entityName;
                for (int i=0; i<19; i++){
                    entityName = input.nextLine();
                    for (int j=0;
j<listEntity.size(); j++){
                        if
(listEntity.get(j).getName().equals(entityName)){
                            ent =
listEntity.get(j);

listSelectedEntity.get(i).add(ent);
                        }
                    }
                }

            } catch (Exception ex){
                System.out.println(ex);
            }

            // Membaca property pada RDF
            rdfHandler.readProperty();
            listSelectedObjProperty =
rdfHandler.getListObjectProperty();
            for(int i=0;
i<listSelectedObjProperty.size(); i++){

listSelectedObjProperty.get(i).setDerivateDomain(li
stSelectedObjProperty.get(i).getDomain());

listSelectedObjProperty.get(i).setDerivateRange(lis
tSelectedObjProperty.get(i).getRange());

            // Berpindah ke tab 4


jTextFieldRdfsFile.setText(tfRDFSConf.getText());

            jTabbedPaneMenu.setSelectedIndex(4);
        }

        else {
```

```
            // Membaca file RDF
            rdfHandler = new RDFHandler();
            System.out.println("Loading file " +
rdfFile.toString());

rdfHandler.setRdfFile(rdfFile.toString());
            rdfHandler.readFile();

            // Membaca kelas pada RDF
            rdfHandler.readClass();
            listEntity =
rdfHandler.getListEntity();

            Comparator<Entity> comparator = (Entity
o1, Entity o2) ->
Integer.parseInt(o1.getId().substring(1))-
Integer.parseInt(o2.getId().substring(1));

Collections.sort(listEntity,comparator);

            // Membaca property pada RDF
            rdfHandler.readProperty();
            listObjectProperty =
rdfHandler.getListObjectProperty();
            // Membuat model RDF
            // rdfHandler.createModel();

            // Mengisi content comboBox dengan
entitas terpilih
            for (int i=0; i<listEntity.size();
i++){

jComboBoxNoUrut.addItem(listEntity.get(i).getName()
.substring(35));

jComboBoxNamaBenda.addItem(listEntity.get(i).getNam
e().substring(35));

jComboBoxAsalDes.addItem(listEntity.get(i).getName(
).substring(35));

jComboBoxAsalKab.addItem(listEntity.get(i).getName(
).substring(35));
```

```
jComboBoxAsalProp.addItem(listEntity.get(i).getName
().substring(35));

jComboBoxLokasiGedung.addItem(listEntity.get(i).get
Name().substring(35));

jComboBoxLokasiRuang.addItem(listEntity.get(i).getN
ame().substring(35));

jComboBoxLokasiLemari.addItem(listEntity.get(i).get
Name().substring(35));

jComboBoxLokasiLain.addItem(listEntity.get(i).getNa
me().substring(35));

jComboBoxKondisi.addItem(listEntity.get(i).getName(
).substring(35));

jComboBoxBahan.addItem(listEntity.get(i).getName().
substring(35));

jComboBoxUkuranPanjang.addItem(listEntity.get(i).ge
tName().substring(35));

jComboBoxUkuranLebar.addItem(listEntity.get(i).getN
ame().substring(35));

jComboBoxUkuranTinggi.addItem(listEntity.get(i).get
Name().substring(35));

jComboBoxUkuranTebal.addItem(listEntity.get(i).getN
ame().substring(35));

jComboBoxUkuranDiameter.addItem(listEntity.get(i).g
etName().substring(35));

jComboBoxTempatPerolehan.addItem(listEntity.get(i).
getName().substring(35));

jComboBoxTempatPembuatan.addItem(listEntity.get(i).
getName().substring(35));
```

```
jComboBoxTahunPembuatan.addItem(listEntity.get(i).g
etName().substring(35));
            }

            for  (int i=1; i<10; i++){

jComboBoxDeepness.addItem(String.valueOf(i));
            }

            // Berpindah ke tab selanjutnya
            jTabbedPaneMenu.setSelectedIndex(1);
        }
    }
```

Kode Sumber 5.6 Fungsi Untuk Membaca Entitas dan *Object Property*

5.3.2. Implementasi Proses *Entity Configuration*

Pada proses ini sistem akan menampilkan daftar entitas pada setiap atribut *dataset*. Fungsi untuk menampilkan entitas telah terangkum pada Kode Sumber 5.5 diatas. Selanjutnya pengguna akan memilih entitas untuk setiap atribut dan menentukan *deepness*/*threshold*. Tahap terakhir adalah sistem menyimpan entitas terpilih dan menentukan *object property* apa saja yang terlibat. Implementasi proses pemilihan ini dapat diliha pada Kode Sumber 5.7 berikut.

```
private void
btnNext2ActionPerformed(java.awt.event.ActionEvent
evt) {

// Menyimpan entitas terpilih pada Array List

listSelectedEntity.get(0).add(listEntity.get(jCombo
BoxNoUrut.getSelectedIndex()));

listSelectedEntity.get(1).add(listEntity.get(jCombo
BoxNamaBenda.getSelectedIndex()));

listSelectedEntity.get(2).add(listEntity.get(jCombo
BoxAsalDes.getSelectedIndex()));
```

```
listSelectedEntity.get(3).add(listEntity.get(jCombo
BoxAsalKab.getSelectedIndex()));

listSelectedEntity.get(4).add(listEntity.get(jCombo
BoxAsalProp.getSelectedIndex()));

listSelectedEntity.get(5).add(listEntity.get(jCombo
BoxLokasiGedung.getSelectedIndex()));

listSelectedEntity.get(6).add(listEntity.get(jCombo
BoxLokasiRuang.getSelectedIndex()));

listSelectedEntity.get(7).add(listEntity.get(jCombo
BoxLokasiLemari.getSelectedIndex()));

listSelectedEntity.get(8).add(listEntity.get(jCombo
BoxLokasiLain.getSelectedIndex()));

listSelectedEntity.get(9).add(listEntity.get(jCombo
BoxKondisi.getSelectedIndex()));

listSelectedEntity.get(10).add(listEntity.get(jComb
oBoxBahan.getSelectedIndex()));

listSelectedEntity.get(11).add(listEntity.get(jComb
oBoxUkuranPanjang.getSelectedIndex()));

listSelectedEntity.get(12).add(listEntity.get(jComb
oBoxUkuranLebar.getSelectedIndex()));

listSelectedEntity.get(13).add(listEntity.get(jComb
oBoxUkuranTinggi.getSelectedIndex()));

listSelectedEntity.get(14).add(listEntity.get(jComb
oBoxUkuranTebal.getSelectedIndex()));

listSelectedEntity.get(15).add(listEntity.get(jComb
oBoxUkuranDiameter.getSelectedIndex()));

listSelectedEntity.get(16).add(listEntity.get(jComb
oBoxTempatPerolehan.getSelectedIndex()));
```

```
listSelectedEntity.get(17).add(listEntity.get(jComb
oBoxTempatPembuatan.getSelectedIndex()));

listSelectedEntity.get(18).add(listEntity.get(jComb
oBoxTahunPembuatan.getSelectedIndex()));

        System.out.println("Entitas terplih adalah:
");
        for (int i=0; i<listSelectedEntity.size();
i++){
            System.out.println(i+1);
            System.out.println("Name    : " +
listSelectedEntity.get(i).get(0).getName());
        }

        // Deepnes value
        deepnessValue =
jComboBoxDeepness.getSelectedIndex()+1;
        System.out.println("Deepness Value= " +
String.valueOf(deepnessValue));

        // Menambah entitas sesuai deepness
        Entity root;
        String next;
        int entitySize = listEntity.size();
        for (int i=0; i<19; i++){
            root =
listSelectedEntity.get(i).get(0);
            next = root.getResource();
            for (int j=1; j<=deepnessValue; j++){
                if (next!=null){
                    for (int k=0; k<entitySize;
k++){
                        if
(listEntity.get(k).getName().compareTo(next)==0){

listSelectedEntity.get(i).add(listEntity.get(k));

next=listEntity.get(k).getResource();
                            break;
                        }

                    }
```

```
                }else{
                    break;
                }
            }
        }

        System.err.println("List entity: ");
        for (int l=0; l<19; l++){
            for (int m=0;
m<listSelectedEntity.get(l).size(); m++){

        listPossibilityObj*Property*.clear();

        int *Object*PropSize =
list*ObjectProperty*.size();
        for (int i=0; i<*Object*PropSize; i++){

list*ObjectProperty*.get(i).setSelected(Boolean.FALSE
);
        }

        String domain;
        String range;
        for (int i=0; i<*Object*PropSize; i++){
            domain =
list*ObjectProperty*.get(i).getDomain();
            range =
list*ObjectProperty*.get(i).getRange();
            for (int j=0; j<19; j++){
                for (int k=0;
k<listSelectedEntity.get(j).size(); k++){
                    if
(domain.compareTo(listSelectedEntity.get(j).get(k).
getName())==0){

                        for (int l=0; l<19; l++){
                            for (int m=0;
m<listSelectedEntity.get(l).size(); m++){
                                if
(range.compareTo(listSelectedEntity.get(l).get(m).g
etName())==0 &&
!list*ObjectProperty*.get(i).getSelected()){
```

```
listObjectProperty.get(i).setDerivateDomain(listSel
ectedEntity.get(j).get(0).getName());

listObjectProperty.get(i).setDerivateRange(listSele
ctedEntity.get(l).get(0).getName());

listPossibilityObjProperty.add(listObjectProperty.g
et(i));

listObjectProperty.get(i).setSelected(Boolean.TRUE)
;
                                    }
                                }
                            }

                        }
                    }
                }
        }

        listModel = new DefaultListModel();
        listModelSelected = new DefaultListModel();

        jTextAreaDescription.setText("");

listModel.addElement(listPossibilityObjProperty.get
(i).getName().substring(35));
        }


jListPossibilityObjectPreperty.setModel(listModel);

jListPossibilityObjectPreperty.addListSelectionList
ener(new ListSelectionListener() {
            @Override
            public void
valueChanged(ListSelectionEvent e) {
                int selectedIndex =
jListPossibilityObjectPreperty.getSelectedIndex();
                if (selectedIndex!=-1){

jListSelectedObjectProperty.clearSelection();
```

```
                    String name =
listPossibilityObjProperty.get(selectedIndex).getNa
me();
                    String domain =
listPossibilityObjProperty.get(selectedIndex).getDo
main();
                    String range =
listPossibilityObjProperty.get(selectedIndex).getRa
nge();
                    String derivateDomain =
listPossibilityObjProperty.get(selectedIndex).getDe
rivateDomain();
                    String derivateRange =
listPossibilityObjProperty.get(selectedIndex).getDe
rivateRange();
                    String comment =
listPossibilityObjProperty.get(selectedIndex).getCo
mment();
                    String subPropertyOf =
listPossibilityObjProperty.get(selectedIndex).getSu
bPropertyOf();

jTextAreaDescription.setText("Name:\n" + name +
"\nDomain: \n" + domain + "\n" + derivateDomain +
"\nRange:\n" + range + "\n" + derivateRange +
"\nSub Property Of:\n" + subPropertyOf +
"\nComment:\n" + comment);
                }
            }
        });

        listSelectedObjProperty.clear();

jListSelectedObjectProperty.setModel(listModelSelec
ted);

jListSelectedObjectProperty.addListSelectionListene
r(new ListSelectionListener() {
            @Override
            public void
valueChanged(ListSelectionEvent e) {
                int selectedIndex =
jListSelectedObjectProperty.getSelectedIndex();
                if (selectedIndex!=-1){
```

```
System.out.println(selectedIndex);

jListPossibility*Object*Preperty.clearSelection();
                    String name =
listSelectedObj*Property*.get(selectedIndex).getName(
);
                    String domain =
listSelectedObj*Property*.get(selectedIndex).getDomai
n();
                    String range =
listSelectedObj*Property*.get(selectedIndex).getRange
();
                    String comment =
listSelectedObj*Property*.get(selectedIndex).getComme
nt();
                    String sub*Property*Of =
listSelectedObj*Property*.get(selectedIndex).getSub*Pr
*operty*Of();

jTextAreaDescription.setText("Name:\n" + name +
"\nDomain: \n" + domain + "\nRange:\n" + range +
"\nSub *Property* Of: \n" + sub*Property*Of +
"\nComment:\n" + comment);
                }
            }
        });

        // Berpindah ke tab selanjutnya
        jTabbedPaneMenu.setSelectedIndex(2);
    }
```

Kode Sumber 5.7 Fungsi Untuk Menyimpan Konfigurasi Entitas

5.3.3. Implementasi Proses *Object Property Configuration*

Pada tahap ini sistem akan menampilkan *object property* yang sesuai dengan entitas terpilih. Pengguna dapat menambahkan *object property* dengan menakan tombol *Add*, dan menghapus pilihan dengan menekan tombol *Remove*. Tombol *Add* akan memindahkan *object property* dari pilihan *object property* ke kolom *object property* terpilih. Sedangkan tombol *Remove*

berfungsi untuk sebaliknya. Hasil implentasi fungsi *Add* dan *Remove* disajikan pada , dan Kode Sumber 5.9 Setelah proses pemilihan selesai sistem akan menyimpan konfigurasi.

```
private void
jButtonAddActionPerformed(java.awt.event.ActionEven
t evt) {
        // TODO add your handling code here:
        int selectedIndex =
jListPossibility[Object]Preperty.getSelectedIndex();
        if (selectedIndex!=-1){
            String selected = (String)
listModel.get(selectedIndex);

            Obj[Property] temp=
listPossibilityObj[Property].get(selectedIndex);

listModel.removeElementAt(selectedIndex);

listPossibilityObj[Property].remove(selectedIndex);

            listModelSelected.addElement(selected);
            listSelectedObj[Property].add(temp);


jListSelected[ObjectProperty].setModel(listModelSelec
ted);
        }
    }
```

Kode Sumber 5.8 Fungsi Untuk Menambahkan *Object Property*

```
private void
jButtonRemoveActionPerformed(java.awt.event.ActionE
vent evt) {
        // TODO add your handling code here:
        int selectedIndex =
jListSelected[ObjectProperty].getSelectedIndex();
        if (selectedIndex!=-1){
            String selected = (String)
listModelSelected.get(selectedIndex);
```

```
            Obj Property temp =
listSelectedObj Property.get(selectedIndex);

listModelSelected.removeElementAt(selectedIndex);

listSelectedObj Property.remove(selectedIndex);

listModel.addElement(selected);
listPossibilityObj Property.add(temp);

jListPossibility Object Preperty.setModel(listModel);
        }
    }
```

Kode Sumber 5.9 Fungsi Untuk Menghapus *Object Property*

5.3.4. Implementasi Proses *Saving Configuration*

Pada tahap ini sistem akan menyimpan konfigurasi dengan menghasilkan *file* mapping.txt dan *file* RDFS *Configuration*. Pengguna akan menentukan lokasi penyimpanan dari *file* tersebut. Hasil implementasi dari proses ini dapat dilihat pada Kode Sumber 5.10 berikut.

```
rivate void
jButtonSaveConfActionPerformed(java.awt.event.Actio
nEvent evt) {
        File Handler file Handler;
        String location;

        jTextAreaProgress.append("Saving
Configuration...\n");

        file Handler = new File Handler();
        location =
jTextField Configuration Location.getText();

        jTextAreaProgress.append("\nCreating
Directory in " + location + "\n");
        file Handler.makeDir(location);
        jTextAreaProgress.append("Directory
created\n");

        // Create mapping file
```

```
        jTextAreaProgress.append("\nWriting Map.txt
file\n");
        File txtFile = new
File(location+"\\Map.txt");
        try {
            PrintWriter output = new
PrintWriter(txtFile);
            for (int i=0; i<19; i++){
                String name =
listSelectedEntity.get(i).get(0).getName();
                output.println(name);

jTextAreaProgress.append(header.get(i) + " set as "
+ name + "\n");
            }
            output.close();
        } catch (FileNotFoundException ex) {
            System.out.println("Error: " + ex);
        }
        //==== End of Create Mapping File

        // Create RDFS Configuration File
        jTextAreaProgress.append("\nWriting
Schema.rdfs file\n");
        File rdfsFile = new
File(location+"\\Schema.rdfs");
        ArrayList<String> ent = new ArrayList<>();
        try {
            PrintWriter output = new
PrintWriter(rdfsFile);
            output.println("<?xml version=\"1.0\"
encoding=\"UTF-8\"?>");
            output.println("\n<rdf:RDF
xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-
syntax-ns#\"
xmlns:rdfs=\"http://www.w3.org/2000/01/rdf-
schema#\" xml:lang=\"en\"
xml:base=\"http://www.cidoc-crm.org/cidoc-
crm/\">");

            output.println("\n\n<!-- \n" +
                "
////////////////////////////////////////////////
////////////////////////////////////\n" +
```

```
                "    //\n" +
                "    // Classes \n" +
                "    //\n" +
                "
////////////////////////////////////////////////
////////////////////////////////\n" +
                "     -->");

            for (int i=0; i<19; i++){
                String id =
listSelectedEntity.get(i).get(0).getId();
                boolean flag = Boolean.TRUE;
                for(int j=0; j<ent.size(); j++){
                    if
(id.compareTo(ent.get(j))==0){
                        flag = Boolean.FALSE;
                        break;
                    }
                }
                if (flag){

jTextAreaProgress.append("Writing " +
listSelectedEntity.get(i).get(0).getName() + "\n");
                    ent.add(id);
                    output.println("\n\n<rdfs:Class
rdf:about=\""+listSelectedEntity.get(i).get(0).getN
ame()+"\">");
                    output.println("\t<rdfs:label
xml:lang=\"en\">"+listSelectedEntity.get(i).get(0).
getLabel()+"</rdfs:label>");
                    if
(listSelectedEntity.get(i).get(0).getComment()!=nul
l){
                        String comment =
listSelectedEntity.get(i).get(0).getComment();

output.println("\t<rdfs:comment>");
//                        output.println(comment);

output.println("\t</rdfs:comment>");
                    }
                    if
(listSelectedEntity.get(i).get(0).getResource()!=nu
ll){
```

```
output.println("\t<rdfs:subClassOf
rdf:resource=\""+listSelectedEntity.get(i).get(0).g
etResource()+"\"/>");
                    }

output.println("</rdfs:Class>");
                }
            }

            output.println("\n\n<!-- \n" +
                "
////////////////////////////////////////////////
////////////////////////////////////\n" +
                "    //\n" +
                "    // Object properties \n" +
                "    //\n" +
                "
////////////////////////////////////////////////
////////////////////////////////////\n" +
                "     -->");

            int length =
listSelectedObjProperty.size();
            for (int i=0; i<length; i++){
                jTextAreaProgress.append("Writing "
+ listSelectedObjProperty.get(i).getName() + "\n");

output.println("\n\n<rdf:Property
rdf:about=\""+listSelectedObjProperty.get(i).getNam
e()+"\">");
                    output.println("\t<rdfs:label
xml:lang=\"en\">"+listSelectedObjProperty.get(i).ge
tLabel()+"</rdfs:label>");
                    if
(listSelectedObjProperty.get(i).getComment()!=null)
{
                        String comment =
listSelectedObjProperty.get(i).getComment();

output.println("\t<rdfs:comment>");
//                        output.println(comment);

output.println("\t</rdfs:comment>");
```

```
                    }
                    output.println("\t<rdfs:domain
rdf:resource=\""+listSelectedObjProperty.get(i).get
Domain()+"\"/>");
                    output.println("\t<rdfs:range
rdf:resource=\""+listSelectedObjProperty.get(i).get
Range()+"\"/>");
                    if
(listSelectedObjProperty.get(i).getSubPropertyOf()!
=null){

output.println("\t<rdfs:subPropertyOf
rdf:resource=\""+listSelectedObjProperty.get(i).get
SubPropertyOf()+"\"/>");
                    }

output.println("</rdf:Property>");
            }

            output.println("\n</rdf:RDF>");
            output.close();
        } catch (FileNotFoundException ex) {
            System.out.println("Error: " + ex);
        }
        //==== End of Create RDFS Configuration
File

        jTextAreaProgress.append("\nConfiguration
saved. Please click next button to continue!");

    }
```

Kode Sumber 5.10 Fungsi Untuk Proses *Saving Configuration*

5.3.5. Implementasi Proses *Generate* Metadata

Pada proses ini sistem akan menghasilkan metadata warisan budaya dalam bentuk *file* RDF/XML. Isi dari metadata diambil dari *dataset* yang telah dipilih. Sebelumnya pengguna menentukan lokasi dari penyimpanan file tersebut. Hasil implenetasi dari proses ini disajikan pada Kode Sumber 5.11**Error! Reference source not found.** berikut.

```
 private void
jButtonGenerateMetadataActionPerformed(java.awt.eve
nt.ActionEvent evt) {

        ArrayList<String> content = new
ArrayList<>();

        try {
            content = ExcelHandler.getRecord(1);
        } catch (IOException ex) {
            System.err.println(ex);
        }

        String
base="http://www.semanticweb.org/agra/ontologies/20
16/tugasAkhir#";

        for (int j=2; j<12; j++){

jTextAreaGenerateMetadata.append("\nData artefak "
+ (j-1) + ": \n================================
\n");


jTextAreaGenerateMetadata.append("Creating file: "
+
jTextFieldMetadaLocation.getText()+"\\"+content.get
(0)+".rdf\n");
            File txtFile = new
File(jTextFieldMetadaLocation.getText()+"\\"+conten
t.get(0)+".rdf");
            try {
                PrintWriter output = new
PrintWriter(txtFile);
                output.println("<?xml
version=\"1.0\" encoding=\"UTF-8\"?>");
                output.println("\n<rdf:RDF
xmlns=\""+ base +"\"
xmlns:rdfs=\"http://www.w3.org/2000/01/rdf-
schema#\"
xmlns:owl=\"http://www.w3.org/2002/07/owl#\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema#\"
xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-
```

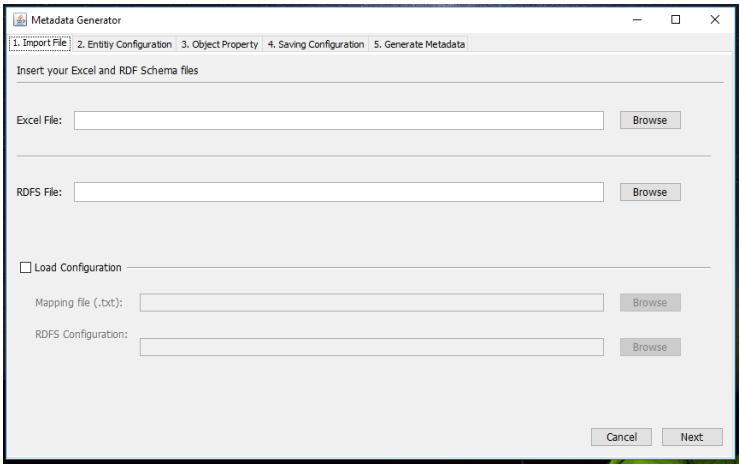
```
syntax-ns#\" xmlns:cidoc-crm=\"http://www.cidoc-
crm.org/cidoc-crm/\" xml:base=\""+base+"\">");

                output.println("\n\n<!-- \n" +
                "
////////////////////////////////////////////////
////////////////////////////////\n" +
                "    //\n" +
                "    // Individuals \n" +
                "    //\n" +
                "
////////////////////////////////////////////////
////////////////////////////////\n" +
                "     -->");

                for (int k=0; k<19; k++){
                    if (content.get(k).compareTo("-
")!=0){

jTextArea*Generate*Metadata.append(header.get(k) + ":
" + content.get(k)+"\n");
                        output.println("\n\n<!-- "
+base + listSelectedEntity.get(k).get(0).getName()
+" -->");

output.println("\n<owl:NamedIndividual
rdf:about=\""+base+content.get(k).replaceAll(" ",
"_")+"\">");
                        output.println("\t<rdf:type
rdf:*resource*=\"cidoc-crm:"+
listSelectedEntity.get(k).get(0).getName().substrin
g(35)+"\"/>");
                        for (int l=0;
l<listSelectedObj*Property*.size(); l++){
                            if
(listSelectedObj*Property*.get(l).getDerivateDomain()
.compareTo(listSelectedEntity.get(k).get(0).getName
())==0){
                                String res= "";
                                for (int m=0;
m<listSelectedEntity.size(); m++){

                                    if
(listSelectedEntity.get(m).get(0).getName().compare
```

```
To(listSelectedObjProperty.get(l).getDerivateRange(
))==0){
                                              res =
content.get(m).replaceAll(" ", "_");
                                          }
                                      }
                                      if
(res.compareTo("-")!=0)

output.println("\t<cidoc-
crm:"+listSelectedObjProperty.get(l).getName().subs
tring(35)+" rdf:resource=\""+base+ res +"\"/>");
                                  }
                          }

output.println("</owl:NamedIndividual>");
                      }
                  }
                  output.println("\n</rdf:RDF>");
                  output.close();

              } catch (FileNotFoundException ex) {
                  System.out.println("Error: " + ex);
              }

              try {
                  content =
ExcelHandler.getRecord(j);
              } catch (IOException ex) {
                  System.err.println(ex);
              }
          }
      }
```

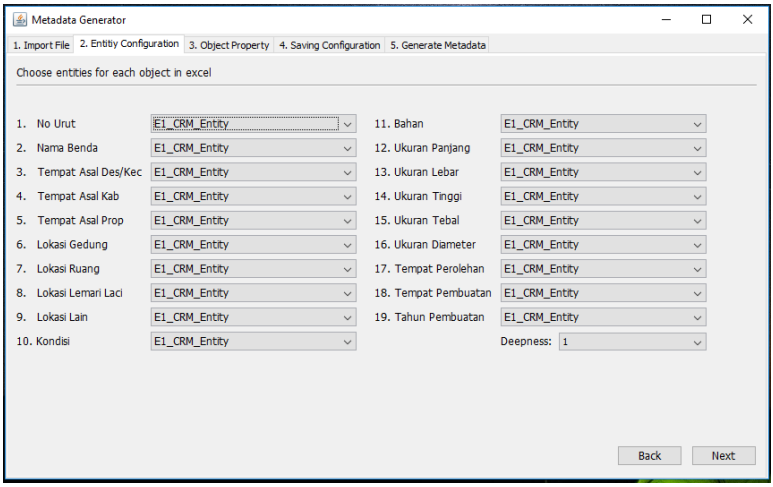
Kode Sumber 5.11 Fungsi Untuk Proses *Generate* Metadata

5.4. Implementasi Antarmuka Pengguna

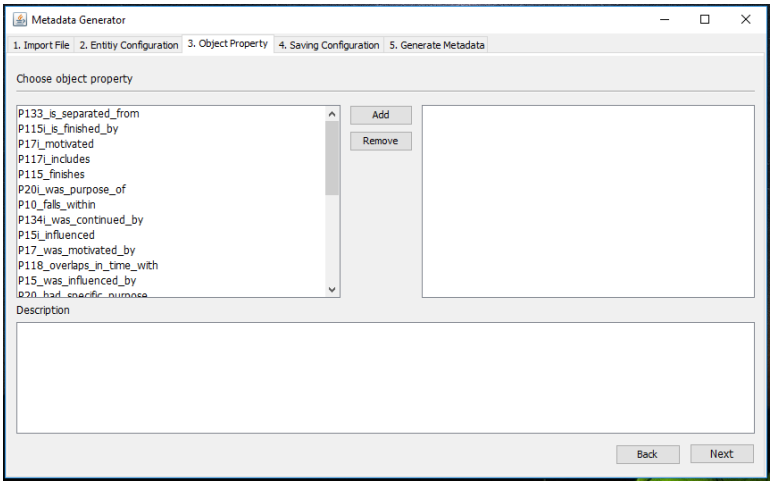
Subbab ini akan membahas implementasi antarmuka berdasarkan rancangan antarmuka yang telah dibahas pada bab IV. Pembuatan antarmuka tersebut menggunakan proses *drag and drop.* Antarmuka yang akan dibahas terdiri dari antarmuka halaman *Import File*, *Entity Configuration*, *Object Property Configuration*, *Saving Configuration*, dan *Generating RDF/XML*. Hasil dari Implementasi antarmuka dapat dilihat pada Gambar 5.1, Gambar 5.2, Gambar 5.3, Gambar 5.4, dan Gambar 5.5 berikut.



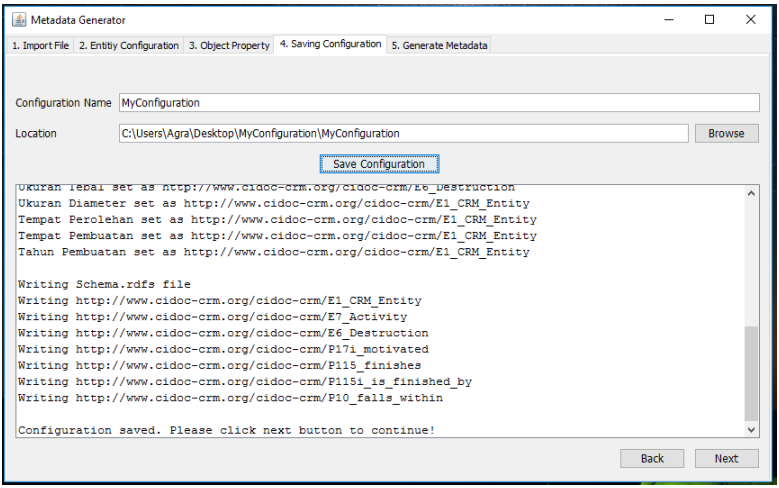
Gambar 5.1 Implementasi Halaman *Import File*



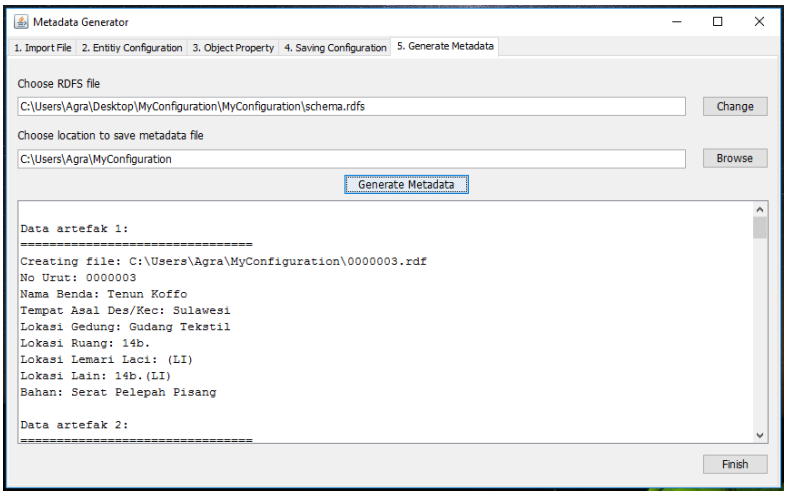
Gambar 5.2 Implementasi halaman Entity *Configuration*



Gambar 5.3 Implementasi halaman *Object Property Configuration*



Gambar 5.4 Implementasi halaman *Saving Configuration*



Gambar 5.5 Implementasi halaman *Generating* RDF/XML

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tentang pengujian dan evaluasi aplikasi yang dikembangkan. Pengujian dilakukan untuk mengetahui fungsionalitas aplikasi. Sistem akan diuji coba fungsionalitasnya dengan menjalankan skenario yang sudah ditentukan. Hasil evaluasi menjelaskan rangkuman pengujian yang dilakukan pada aplikasi ini.

6.1. Lingkungan Pengujian

Pada subbab ini dijelaskan mengenai gambaran lingkungan yang digunakan untuk melakukan uji coba aplikasi. Uji coba aplikasi ini menggunakan *Notebook* Lenovo™ ideapad™ 300 dengan spesifikasi sebagai berikut:

- Spesifikasi Perangkat Keras:
  - Prosesor: Intel(R) Core (TM) i5-6200U CPU @ 2.30GHz (4 CPUs), ~2.4GHz
  - Memori: 4096MB RAM
- Spesifikasi Perangkat Lunak:
  - Sistem Operasi Windows 10 Professional 64 bit
  - Java versi 1.8
  - Protege 4.3

6.2. Data Uji

Data yang digunakan adalah data warisan budaya dari Museum Indonesia yang berupa *file* Excel yang berisi 50 data artefak. *Dataset* tersebut telah diolah sedemikian ruapa untuk disesuaikan dengan kebutuhan sistem. Setelah prosses penyesuaian didapatkan 19 nama atribut yang digunakan dalam proses pembangkitan metadata. Penjelasan dari setiap atribut disajikan pada Tabel 6.1 berikut.

Tabel 6.1 Deskripsi Atribut Dataset

No	Nama Atribut	Penjelasan
1	No Urut	Berisi nomor urut penemuan artefak. Pada Tugas Akhir ini dijadikan sebagai identitas unik dari masing-masing artefak.
2	Nama Benda	Nama artefak
3	Tempat Asal Des/Kec	Desa atau kecamatan tempat ditemukannya artefak
4	Tempat Asal Kab	Kabupaten tempat ditemukannya artefak
5	Tempat Asal Prop	Proponsi tempat ditemukannya artefak
6	Lokasi Gedung	Lokasi gedung tempat penyimpanan artefak saat ini
7	Lokasi Ruang	Lokasi ruang tempat penyimpanan artefak saat ini
8	Lokasi Lemari/Laci	Lokasi lemari/laci tempat penyimpanan artefak saat ini
9	Lokasi Lain	Lokasi lain tempat penyimpanan artefak saat ini
10	Kondisi	Kondisi artefak
11	Bahan	Bahan yang digunakan untuk membuat artefak
12	Ukuran Panjang	Ukuran panjang artefak
13	Ukuran Lebar	Ukuran lebar artefak
14	Ukuran Tinggi	Ukuran tinggi artefak
15	Ukuran Tebal	Ukuran tebal artefak
16	Ukuran Diameter	Ukuran diameter artefak
17	Tempat Perolehan	Tempat perolehan artefak
18	Tempat Pembuatan	Perkiraan tempat pembuatan artefak
19	Tahun Pembuatan	Perkiraan bahan pembuatan artefak

6.3. Skenario Pengujian

Pada sub bab ini dijelaskan skenario pengujian kakas. Pengujian yang dilakukan adalah pengujian fungsionalitas dan pengujian performa. Pengujian fungsionalitas dilakukan dengan metode *black box*. Metode pengujian ini cenderung melihat hasil keluaran sistem.

6.4. Pengujian Fungsionalitas

Pengujian ini dilakukan dengan sejumlah skenario sebagai tolok ukur keberhasilan sistem. Pengujian didasarkan pada kasus penggunaan yang digunakan dalam pembangunan sistem, seperti yang telah dijelaskan pada subbab 4.1.5.

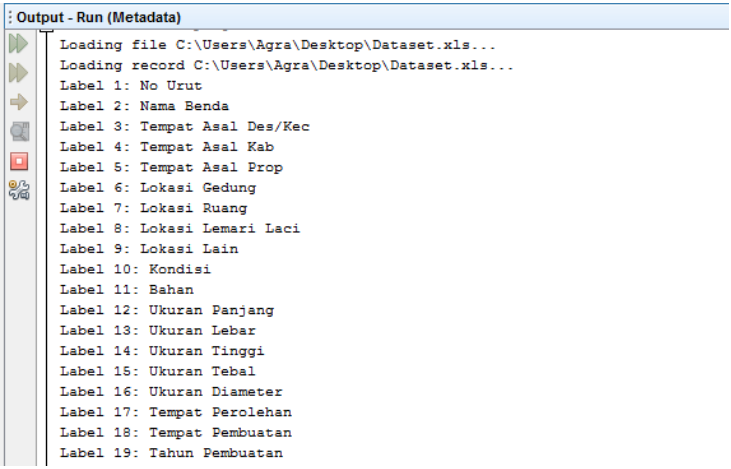
6.4.1. Pengujian Memuat *File* Pendukung

Pengujian fungsionalitas untuk memuat *file* pendukung dilakukan dengan memuat *file* Excel sebagai *dataset* dan *file* RDFS CIDOC-CRM. Rincian pengujian dijelaskan pada Tabel 6.2. Hasil pengujian ditunjukkan pada Gambar 6.1, Gambar 6.2, dan Gambar 6.3.

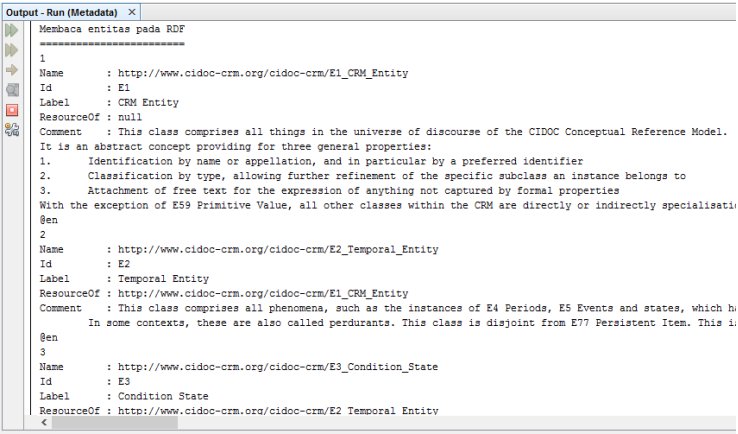
Tabel 6.2 Pengujian Fitur Memuat *File* Pendukung

ID	UJ.UC01
Referensi Kasus Penggunaan	UC01
Nama	Memuat *File* Pendukung
Kondisi Awal	Sistem meminta masukan *file* Excel sebagai *dataset* dan *file* RDFS CIDOC
Masukan	Dataset Artefak Museum Indonesia dalam bentuk *file* Excel dan RDFS CIDOC.

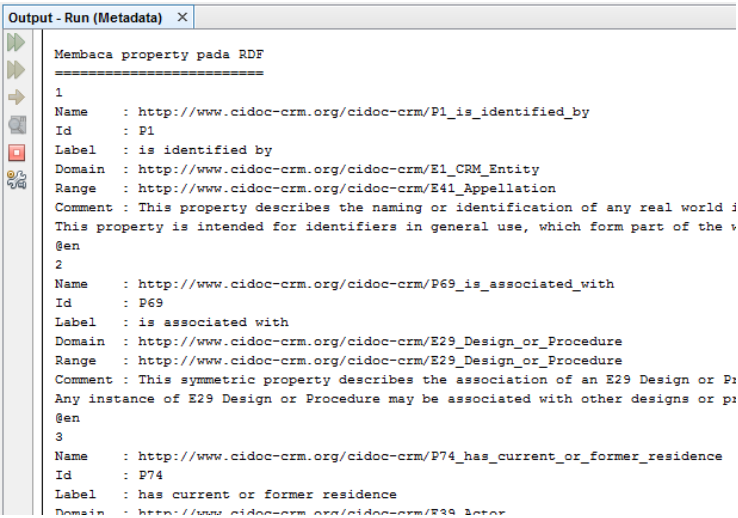
Skenario	1. Pengguna membuka tab *Import File* 2. Pengguna memasukkan *dataset* aritfak yang berupa *file* Excel 3. Pengguna memasukkan *file* RDFS CIDOC-CRM 4. Pengguna menekan tombol "Next" 5. Sistem membaca *file* Excel 6. Sistem membaca entitas dan *data property* pada RDFS CIDOC 7. Sistem menyimpan lokasi *file dataset* 8. Sistem menyimpan entitas dan *object property* dari RDFS CIDOC-CRM
Kondisi Akhir	Sistem menampilkan lokasi penyimpanan *dataset file* Excel dan menampilkan atribut sesuai *dataset.* Sistem menampilkan entitas dan *object property* dari RDFS CIDOC-CRM
Hasil Pengujian	Berhasil



Gambar 6.1 Hasil Keluaran Pembacaan Dataset Excel



Gambar 6.2 Hasil Keluaran Pengujian Pembacaan Entitas RDFS CIDOC-CRM



Gambar 6.3 Hasil Keluaran Pengujian Pembacaan *Object Property* RDFS CIDOC-CRM

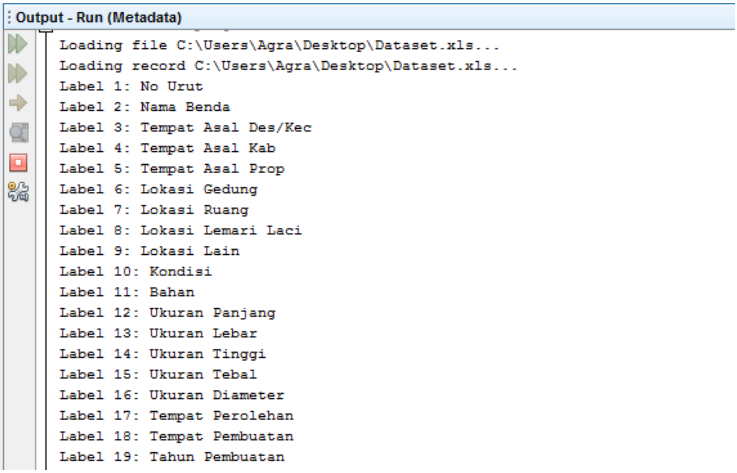
6.4.2. Pengujian Memuat *File* Konfigurasi

Pengujian fungsionalitas untuk memuat *file* konfigurasi dilakukan dengan memuat dataset dan *file* konfigurasi berupa mapping.txt dan RDFS Configuration. Rincian pengujian dijelaskan pada Tabel 6.3. Hasil pengujian ditunjukkan pada Gambar 6.4, Gambar 6.5, dan Gambar 6.6.

Tabel 6.3 Pengujian Fitur Memuat *File* Konfigurasi

ID	UJ.UC02
Referensi Kasus Penggunaan	UC02
Nama	Memuat *File* Konfigurasi
Kondisi Awal	Sistem meminta masukan *file* Excel sebagai *dataset* dan *file* konfigurasi berupa mapping.txt dan RDFS *Configuration*.
Masukan	Dataset Artefak Museum Indonesia dalam bentuk *file* Excel dan RDFS CIDOC.
Skenario	1. Pengguna membuka tab *Import File* 2. Pengguna mencentang pilihan *Load Configuration* 3. Pengguna memasukkan *dataset* aritfak yang berupa *file* Excel 4. Pengguna memasukkan *file* mapping.txt 5. Pengguna memasukkan *file* RDFS *Configuration* 6. Sistem menyimpan lokasi *file dataset* 7. Sistem menyimpan entitas dan *object property* dari RDFS *Configuration*

Kondisi Akhir	Sistem menampilkan lokasi penyimpanan *dataset file* Excel dan menampilkan atribut sesuai *dataset.* Sistem menampilkan entitas dan *object property* dari RDFS *Configuration.*
Hasil Pengujian	Berhasil



Gambar 6.4 Hasil Keluaran Pembacaan Dataset Excel

```
Membaca entitas pada RDF
========================
Loading Mapping file: C:\Users\Agra\Desktop\MyConfiguration\Map.txt
1
Name       : http://www.cidoc-crm.org/cidoc-crm/E4_Period
Id         : E4
Label      : Period
ResourceOf : http://www.cidoc-crm.org/cidoc-crm/E2_Temporal_Entity
Comment    :
        @en
2
Name       : http://www.cidoc-crm.org/cidoc-crm/E1_CRM_Entity
Id         : E1
Label      : CRM Entity
ResourceOf : null
Comment    :
        @en
3
Name       : http://www.cidoc-crm.org/cidoc-crm/E1_CRM_Entity
Id         : E1
Label      : CRM Entity
```

Gambar 6.5 Hasil Keluaran Pembacaan Entitas Sesuai *Mapping File*

```
Membaca property pada RDF
=========================
1
Name    : http://www.cidoc-crm.org/cidoc-crm/P20i_was_purpose_of
Id      : P20i
Label   : was purpose of
Domain  : http://www.cidoc-crm.org/cidoc-crm/E5_Event
Range   : null
Comment : null
2
Name    : http://www.cidoc-crm.org/cidoc-crm/P17i_motivated
Id      : P17i
Label   : motivated
Domain  : http://www.cidoc-crm.org/cidoc-crm/E1_CRM_Entity
Range   : http://www.cidoc-crm.org/cidoc-crm/E7_Activity
Comment : null
3
Name    : http://www.cidoc-crm.org/cidoc-crm/P117i_includes
Id      : P117i
Label   : includes
Domain  : http://www.cidoc-crm.org/cidoc-crm/E2 Temporal Entity
```

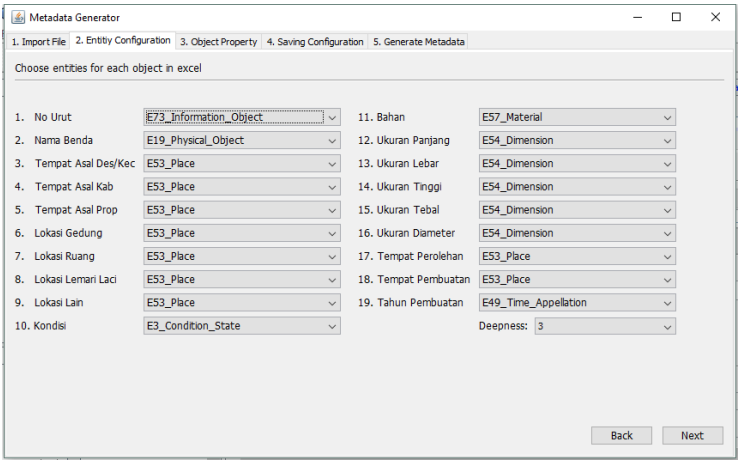
Gambar 6.6 Hasil Keluaran Pembacaan *Object Property* Pada RDFS *Configuration*

6.4.3. Pengujian Konfigurasi Entitas

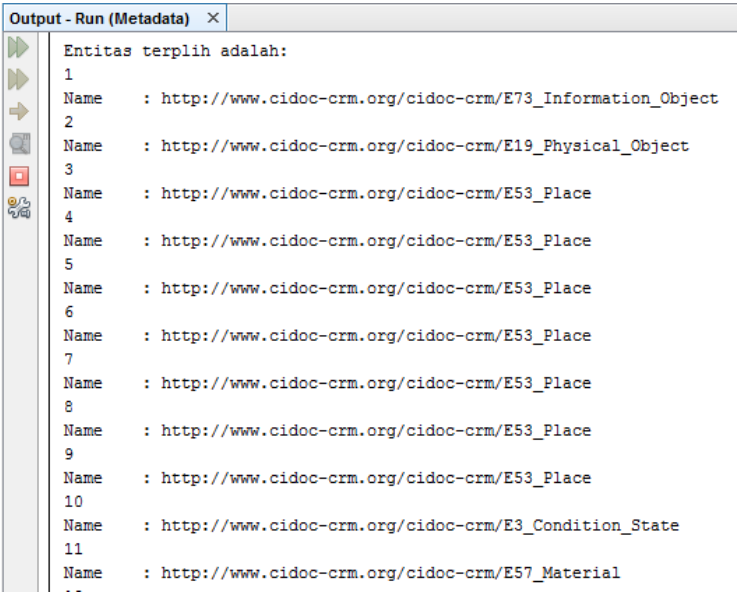
Pengujian fungsionalitas untuk konfigurasi entitas dilakukan dengan memilih entitas dari RDFS CIDOC. Rincian pengujian dijelaskan pada Tabel 6.4. Hasil pengujian ditunjukkan pada Gambar 6.7 dan Gambar 6.8.

Tabel 6.4 Pengujian Fitur Konfigurasi Entitas

ID	UJ.UC03
Referensi Kasus Penggunaan	UC03
Nama	Konfigurasi Entitas
Kondisi Awal	Sistem menampilkan daftar entitas untuk setiap atribut pada dataset
Masukan	-
Skenario	1. Pengguna membuka tab *Entity Configuration* 2. Sistem menampilkan tab *Entity Configuration* 3. Pengguna memilih entitas CIDOC unutk setiap atribut *dataset* sesuai Gambar 6.6 4. Pengguna menentukan threshold 5. Pengguna menekan tombol “Next” 6. Sistem menyimpan entitas yang dipilih 7. Sistem meanampilkan entitas yang dipilih pengguna 8. Sistem membuka tab *Data Property*
Kondisi Akhir	Sistem menampilkan entitas yang dipilih pengguna
Hasil Pengujian	Berhasil



Gambar 6.7 Percobaan Konfigurasi Entitas



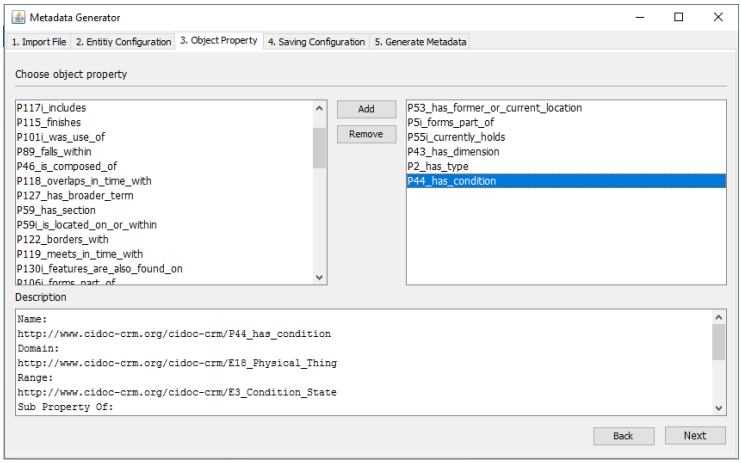
Gambar 6.8 Hasil Keluaran Konfigurasi Entitas

6.4.4. Pengujian Konfigurasi *Object Property*

Pengujian fungsionalitas untuk konfigurasi *object property* dilakukan dengan memilih *object property* dari entitas terpilih. Rincian pengujian dijelaskan pada Tabel 6.5. Hasil pengujian dapat dilihat pada Gambar 6.9.

Tabel 6.5 Pengujian Fitur Konfigurasi *Object Property*

ID	UJ.UC04
Referensi Kasus Penggunaan	UC04
Nama	Konfigurasi *Object Property*
Kondisi Awal	Sistem menampilkan daftar *object property* dari entitas terpilih
Masukan	-
Skenario	1. Pengguna membuka tab *Object Property Configuration* 2. Sistem menampilkan tab *Entity Configuration* 3. Sistem menampilkan *object property* yang sesuai dengan entitas terpilih 4. Pengguna memilih *object property* sesuai Gambar 6.8 5. Sistem menyimpan *object property* yang dipilih 6. Sistem menampilkan *object property* yang dipilih 7. Sistem menampilkan tab *Save Configuration*
Kondisi Akhir	Sistem menampilkan *object property* yang dipilih pengguna
Hasil Pengujian	Berhasil



Gambar 6.9 Percobaan Konfigurasi *Object Property*

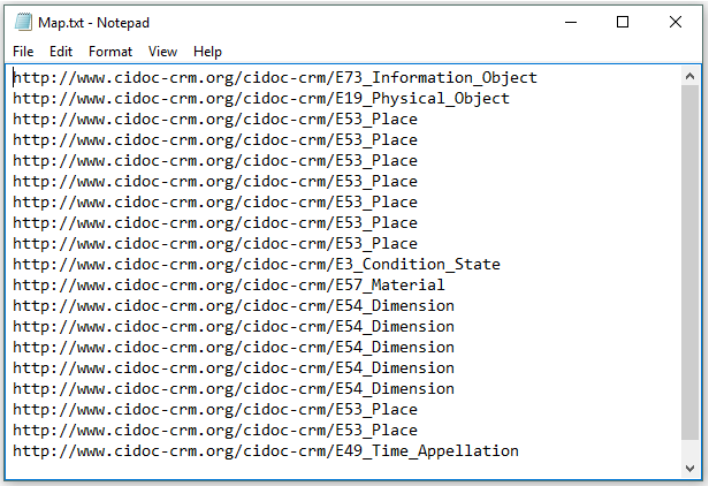
6.4.5. Pengujian Menyimpan Konfigurasi

Pengujian fungsionalitas untuk menyimpan konfigurasi dilakukan dengan menentukan lokasi penyimpanan untuk *file* konfigurasi yang telah dibuat. Rincian pengujian dijelaskan pada Tabel 6.6. Hasil pengujian dapat dilihat pada Gambar 6.10, Gambar 6.11, dan Gambar 6.12

Tabel 6.6 Pengujian Fitur Menyimpan Konfigurasi

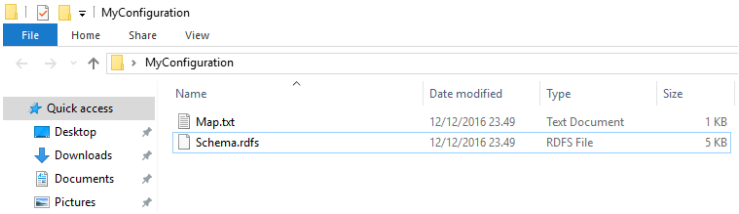
ID	UJ.UC05
Referensi Kasus Penggunaan	UC05
Nama	Menyimpan Konfigurasi
Kondisi Awal	Sistem menyimpan konfigurasi dalam bentuk variabel program
Masukan	-

Skenario	1. Pengguna membuka tab *Save Configuration* 2. Sistem menampilkan tab *Save Configuration* 3. Pengguna memberi nama konfigurasi 4. Pengguna menentukan lokasi penyimpanan 5. Pengguna menekan tombol "Next" 6. Sistem men-*generate file* mapping.txt dan RDFS *Configuration* 7. Sistem menyimpan *file* mapping.txt dan RDFS *Configuration* ke lokasi yang telah ditentukan 8. Sistem menampilkan tab *Generate*
Kondisi Akhir	Sistem menghasilkan *file* map.txt dan *schema*.txt sebagai *file* konfigurasi
Hasil Pengujian	Berhasil



Gambar 6.10 Isi *File* Map.txt

Gambar 6.11 Isi *File* RDFS *Configuration*



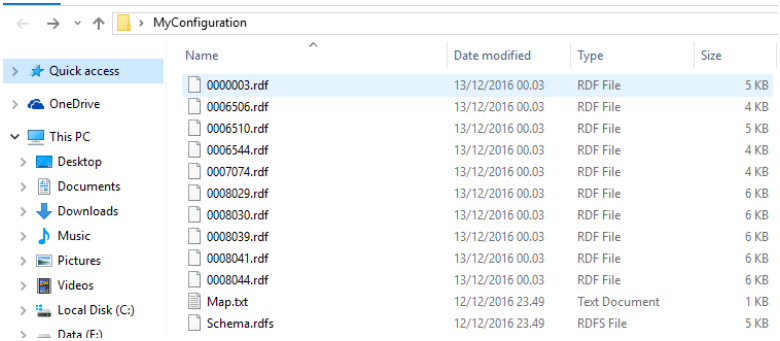
Gambar 6.12 Lokasi Penyimpanan *File* Konfigurasi

6.4.6. Pengujian *Generate* Metadata

Pengujian fungsionalitas untuk *generate* metadata dilakukan dengan menentukan lokasi penyimpanan untuk *file* RDF yang dihasilkan. Sistem akan menghasilkan *file* RDF/XML untuk setiap data artefak. Rincian pengujian dijelaskan pada Tabel 6.7. Hasil pengujian dapat dilihat pada Gambar 6.13 dan Gambar 6.14.

Tabel 6.7 Pengujian Fitur *Generate* Metadata

ID	UJ.UC06
Referensi Kasus Penggunaan	UC06
Nama	*Generate* Metadata
Kondisi Awal	Sistem meminta masukan nama dan lokasi penyimpanan *file* RDF/XML
Masukan	-
Skenario	1. Pengguna membuka tab *Generate* Metadata 2. Sistem menampilkan tab *Generate* Metadata 3. Pengguna menentukan lokasi penyimpanan 4. Pengguna menekan tombol “Next” 5. Sistem men-*generate file* RDF/XML 6. Sistem menyimpan *file* RDF/XML
Kondisi Akhir	Sistem menghasilkan *file* map.txt dan *schema*.txt sebagai *file* konfigurasi
Hasil Pengujian	Berhasil

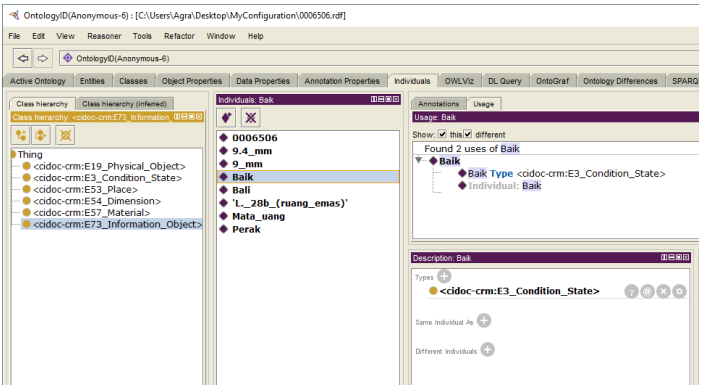


Gambar 6.13 Lokasi Penyimpanan *File* RDF/XML

```
<!--
http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#http://www.
cidoc-crm.org/cidoc-crm/E1_CRM_Entity -->

<owl:NamedIndividual rdf:about="
http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#0006506">
    <rdf:type rdf:resource="cidoc-crm:E1_CRM_Entity"/>
    <cidoc-crm:P17i_motivated rdf:resource="
    http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#Bali"/>
</owl:NamedIndividual>


<!--
http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#http://www.
cidoc-crm.org/cidoc-crm/E6_Destruction -->

<owl:NamedIndividual rdf:about="
http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#Mata_uang">
    <rdf:type rdf:resource="cidoc-crm:E6_Destruction"/>
    <cidoc-crm:P20i_was_purpose_of rdf:resource="
    http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#Bali"/>
</owl:NamedIndividual>


<!--
http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#http://www.
cidoc-crm.org/cidoc-crm/E8_Acquisition -->

<owl:NamedIndividual rdf:about="
http://www.semanticweb.org/agra/ontologies/2016/tugasAkhir#Bali">
    <rdf:type rdf:resource="cidoc-crm:E8_Acquisition"/>
</owl:NamedIndividual>
```

Gambar 6.14 Isi dari *File* RDF/XML

Untuk membuktikan kebenaran sintaknya, metadata tersebut juga dapat dibuka menggunakan kakas bantu Protege tanpa muncul pesan *error*. Sehingga dapat disimpulkan bahwa sintak metadata tersebut valid. Hasil pengujian ini dapat dilihat pada Gambar 6.15 berikut.



Gambar 6.15 Pengujian Metadata Menggunakan Protege

6.5. Evaluasi Pengujian

Rangkuman hasil pengujian dapat dilihat pada Tabel 6.8. Berdasarkan hasil rangkuman pada tabel tersebut, dapat dilihat bahwa fungsional sistem berjalan dengan baik. Sistem dapat melakukan rentetan proses pembangkitan metadata dengan 50 dataset dalam waktu kurang dari 5 menit (dengan asumsi pengguna telah merencanakan konfigurasi).

Tabel 6.8 Rangkuman Hasil Pengujian

ID	Nama	Hasil
UJ.UC01	Memuat *File* Pendukung	Berhasil
UJ.UC02	Memuat *File* Konfigurasi	Berhasil
UJ.UC03	Konfigurasi Entitas	Berhasil

UJ.UC04	Konfigurasi *Object Property*	Berhasil
UJ.UC05	Menyimpan Konfigurasi	Berhasil
UJ.UC06	*Generate* Metadata	Berhasil

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan kesimpulan yang dapat diambil dalam pengerjaan tugas akhir dan saran tentang kemungkinan pengembangan yang dapat dilakukan pada tugas akhir ini di masa yang akan datang.

7.1. Kesimpulan

Dari proses analisa, perancangan, implementasi dan pengujian perangkat lunak aplikasi metadata generator dapat ditarik kesimpulan sebagai berikut:

1. Aplikasi dapat men-*generate* metadata koleksi warisan budaya melalui beberapa tahap konfigurasi yaitu: *Import File, Entity Configuration, Object Property Configuration, Saving Configuration dan Generate* Metadata.
2. Aplikasi dapat menyimpan *file* konfigurasi untuk memudahkan pengguna dalam proses pembangkitan metadata di waktu selanjutnya.
3. Proses pembangkitan metadata melibatkan bantuan *framework* Jena dalam proses pembacaan Entitas dan *Object Property* pada RDF *Schema*.
4. Proses pembangkitan metadata koleksi warisan budaya melalui aplikasi menjadi lebih mudah dan lebih cepat dibandingkan jika dilakukan secara manual.

7.2. Saran

Berikut adalah beberapa saran untuk kemungkinan pengembangan perangkat lunak metadata generator di masa mendatang, berdasarkan hasil rancangan, implementasi dan uji coba yang telah dilakukan.

1. Aplikasi yang dibangun hanya berfokus pada relasi entitas dan *object property*. Aplikasi belum dapat melibatkan relasi *data property* dalam proses pembangunan metadata

warisan budaya. Sehingga akan menjadi lebih baik jika aplikasi dapat melibatkan relasi data *property* pada prosesnya.

2. Aplikasi yang dibangun hanya dapat membaca *dataset* warisan budaya yang telah disesuaikan dengan kebutuhan sistem. Sehingga dibutuhkan waktu tambahan untuk melalui proses ini. Aplikasi akan menjadi lebih baik jika dapat membaca *dataset* yang lebih fleksibel.

DAFTAR PUSTAKA

[1] Asalasah, "Ringkasan Materi RDF – *Semantic Web*," 29 11 2015. [Online]. Available: http://asalasah.net/ringkasan-materi-rdf-semantic-web/. [Diakses 28 Nopember 2016].

[2] Cultural in Development (CiD), "What is Culturaal Heritage," 2012. [Online]. Available: http://www.culturein development.nl/culturalheritage/what_is_cultural_heritage. [Diakses 14 Desember 2015].

[3] Tjahjono, "Memelihara Warisan Budaya Tak Benda," Kompas, 24 Nopember 2014.

[4] H. M. Szczepanowska, Conservation of Cultural Heritage: Key Principles and Approaches, Routledge, 2013.

[5] Wikipedia, "Metadata," Wikipedia, 16 9 2015. [Online]. Available: https://id.wikipedia.org/wiki/Metadata. [Diakses 30 Nopember 2015].

[6] SmitDev Community, "Metadata," Metadata, 9 7 2010. [Online]. Available: http://www.smitdev.com/posts/metadata362.php. [Diakses 29 Nopember 2015].

[7] I. Kurniawan, "*Semantic Web*, RDF, ontologi," 6 7 2012. [Online]. Available: http://studyinformatics.blogspot.co.id/2012/07/semantic-web-rdf-ontologi.html. [Diakses 30 Nopember 2015].

[8] F. Indra, Case Tool Untuk Pemodelan Ulang Metadata, Surabaya: Jurusan Teknik Informatika - ITS, 2010.

[9] D. Sahputra, Pembangkit Metadata Untuk Sistem Temu Kembali Citra Berbasis *Semantic Web*, Surabaya: Jurusan Teknik Informatika-ITS, 2010.

[10] W3C, "RDF *Schema*," W3C, 25 2 2014. [Online]. Available: http://www.w3.org/TR/rdf-*schema*/. [Diakses 2 Desember 2015].

[11] International Council of Museums (ICoM), "What is the CIDOC CRM," 5 12 2014. [Online]. Available: http://www.cidoc-crm.org/. [Diakses 2 Desember 2015].

[12] Wikipedia, "Jena (*Framework*)," 30 9 2015. [Online]. Available: https://en.wikipedia.org/wiki/Jena_%28*framework*%29. [Diakses 14 Desember 2015].

[13] Apache , "Writing RDF in Apache Jena," Apache, [Online]. Available: https://jena.apache.org/documentation/io/rdf-output.html. [Diakses 14 Desember 2015].

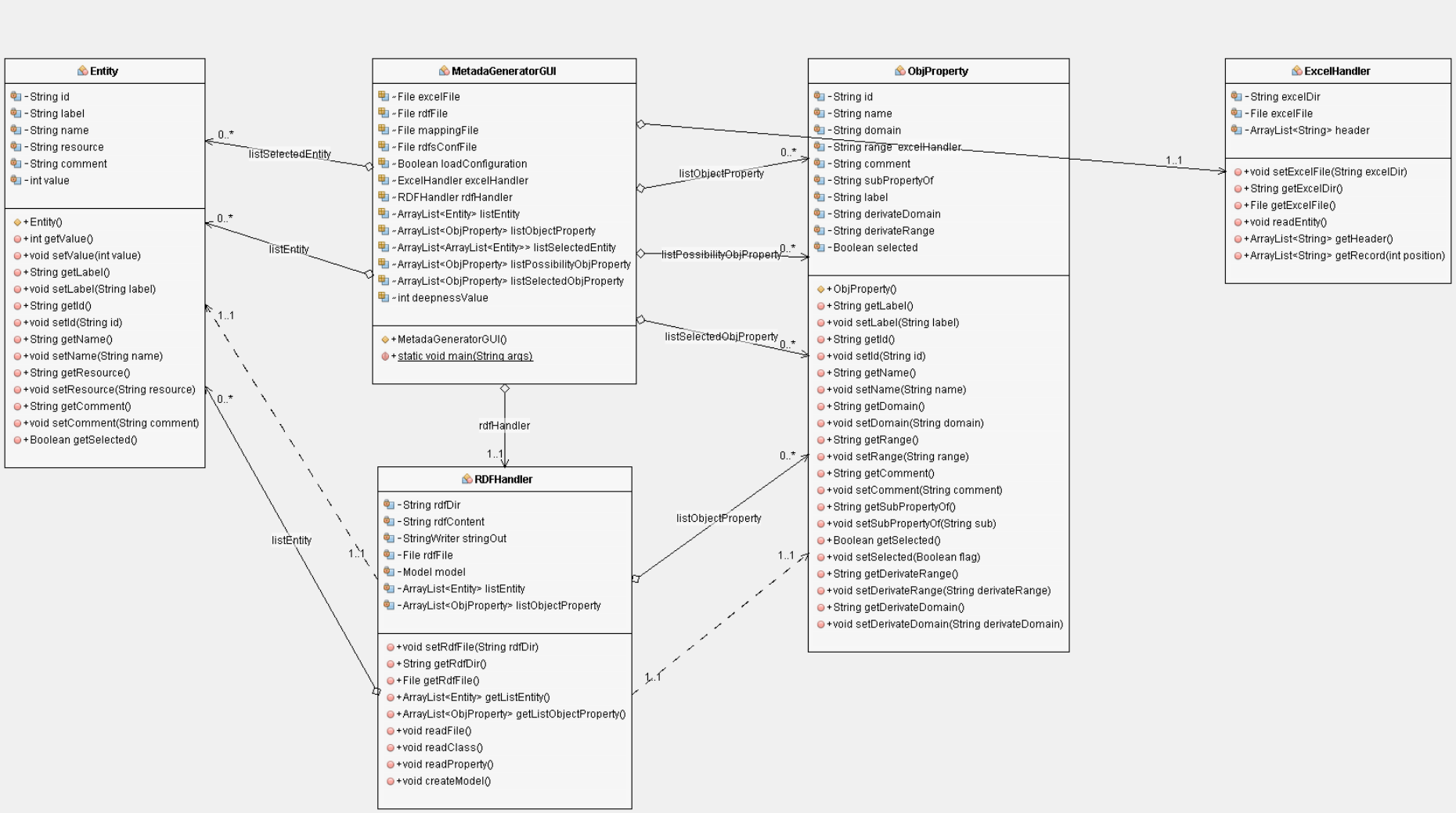
[14] I. Herman, "W3C *Semantic Web*," W3C, 12 11 2009. [Online]. Available: http://www.w3.org/RDF/FAQ. [Diakses 1 Desember 2015].

LAMPIRAN 1
CONTOH DATASET WARISAN BUDAYA MUSEUM INDONESIA

No Urut	Nama Benda	Tempat Asal Des/Kec	Tempat Asal Kab	Tempat Asal Prop	Lokasi Gedung	Lokasi Ruang	Lokasi Lemari Laci	Lokasi Lain
0000003	Tenun Koffo	Sulawesi			Gudang Tekstil	14b.	(LI)	14b.(LI)
0006506	Mata uang			Bali				L. 28b (ruang emas)
0006510	Gelang Kaki	Mojosari		Jawa Timur		L. 7b		
0006544	Mata Uang			Bali				L. 28b (ruang emas)
0007074	Mata Uang			Bali				L. 28b (ruang emas)
0008029	Tangkai Cermin	Semanten	Madiun	Jawa Timur				L. 6B (perunggu baru)
0008030	Tangkai Cermin	Piyungan	Sleman	Yogyakarta				L. 6B (perunggu baru)
0008039	Genta Pendeta		Kedu	Jawa Tengah	Ruang Perunggu Baru			Ruang Perunggu Baru
0008041	Genta Pendeta	Cikandang	Garut	Jawa Barat	Ruang Perunggu Baru			Ruang Perunggu Baru
0008044	Genta Pendeta		Kediri	Jawa Timur	Ruang Perunggu Baru			Ruang Perunggu Baru
0008053	Genta Pendeta			Jawa Barat	Ruang Perunggu Baru			Ruang Perunggu Baru
0008055	Genta Pendeta	Sengguruh	Malang	Jawa Timur	Ruang Perunggu Baru			Ruang Perunggu Baru
0008056	Genta Pendeta	Batu Agung	Jembrana	Bali	Ruang Perunggu Baru			Ruang Perunggu Baru
0008076	Bandul Genta Pendeta		Rembang	Jawa Tengah	Ruang Perunggu Baru			Ruang Perunggu Baru
0008102	Puncak Genta Pendeta	Bangil	Pasuruan	Jawa Timur	Ruang Perunggu Baru			Ruang Perunggu Baru
0008105	Puncak Genta Pendeta		Cirebon	Jawa Barat	Ruang Perunggu Baru			Ruang Perunggu Baru
0008107	Genta Pendeta	Bagelen	Purworejo	Jawa Tengah	Ruang Perunggu Baru			Ruang Perunggu Baru
0008108	Genta Pendeta	Ledok/Begelan	Purworejo	Jawa Tengah	Ruang Perunggu Baru			Ruang Perunggu Baru
0008119	Genta Pendeta		Ponorogo	Jawa Timur	Ruang Perunggu Baru			Ruang Perunggu Baru

Lokasi Lain	Kondisi	Bahan	Ukuran Panjang	Ukuran Lebar	Ukuran Tinggi	Ukuran Tebal	Ukuran Diameter	Tempat Perolehan	Tempat Pembuatan	Tahun Pembuatan
14b.(LI)		Serat Pelepah Pisang								
L. 28b (ruang emas)	Baik	Perak	9.4 mm	9 mm						
	Baik	Tembaga				1.2 cm	10.5 cm			
L. 28b (ruang emas)	Baik	Perak	8.1 mm	8 mm						
L. 28b (ruang emas)	Baik	Perak	8 mm	7 mm						
L. 6B (perunggu baru)	Cukup baik	Perunggu		8.5 cm	9 cm					Abad 12-14 Masehi
L. 6B (perunggu baru)	Baik	Perunggu		6.7 mm	8.2 mm					Abad 10-11 Masehi
Ruang Perunggu Baru	Baik	Perunggu			15.3 cm		8 cm			Abad ke-8 - 10 Masehi
Ruang Perunggu Baru	Baik	Perunggu			13.5 cm		5.5 cm			Abad ke-13 - 15 Masehi
Ruang Perunggu Baru	Baik	Perunggu	19.2 cm				9.6 cm			Abad ke-10 - 11 Masehi
Ruang Perunggu Baru	Cukup	Perunggu			17.3 cm		8.9 cm			Abad ke-14 Masehi
Ruang Perunggu Baru	Baik	Perunggu			14.8 cm		6.2 cm			Abad ke-10 - 12 Masehi
Ruang Perunggu Baru	Baik	Perunggu			18 cm		10.2 cm			Abad ke-7 - 8 Masehi
Ruang Perunggu Baru	Baik	Perunggu			16 cm		8.4 cm			Abad ke-8 - 10 Masehi
Ruang Perunggu Baru	Baik	Perunggu			10 cm					Abad ke-13 - 15 Masehi
Ruang Perunggu Baru	Sedikit Rusak	Perunggu			12.2 cm					Abad ke-8 - 10 Masehi
Ruang Perunggu Baru	Baik	Perunggu			14.2 cm		5.7 cm			Abad ke-7 - 9 Masehi
Ruang Perunggu Baru	Baik	Perunggu			13.8 cm		4.8 cm			Abad ke-8 - 9 Masehi
Ruang Perunggu Baru	Baik	Perunggu			14.7 cm		6 cm			Abad ke-11 - 13 Masehi

LAMPIRAN 2
DIAGRAM KELAS APLIKASI METADATA GENERATOR



(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Wimba Agra Wicesa, anak pertama dari dua bersaudara yang lahir di Banyuwangi pada 3 Agustus 1994. Penulis telah menempuh pendidikan formal mulai dari SD Negeri 1 Cluring Banyuwangi (2000-2006), SMP Negeri 1 Cluring Banyuwangi (2006-2009), SMA Negeri 1 Genteng Banyuwangi (2009-2012), penulis mengikuti tes SNMPTN dan diterima di Jurusan Teknik Informatika FTIf-ITS pada tahun 2012 dan terdaftar dengan NRP 5112100102.

Di Jurusan Teknik Informarika ini Penulis mengambil Bidang Studi Manajemen Informasi. Penulis sempat aktif mengikuti berbagai kepanitiaan diantaranya *Schema*tics 2013, *Schema*tics 2014, dan kegiatan lain di jurusan. Penulis juga aktif menjadi anggota Himpunan Mahasiswa Teknik Computer-Informatika pada tahun 2013 dan sempat menjadi Asisten Dosen PIKTI-ITS. Penulis dapat dihubungi email agrawimba@gmail.com.