SKRIPSI

KONVERSI JADWAL MENGAWAS UJIAN KE FORMAT ICS DENGAN APACHE POI, ICAL4J, DAN JAVAFX

ARIQ RAHMAERI

NPM: 2011730066

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2015

UNDERGRADUATE THESIS

CONVERTION OVERSEEN EXAM SCHEDULE TO ICS FORMAT WITH APACHE POI, ICAL4J, AND JAVAFX

ARIQ RAHMAERI

NPM: 2011730066

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2015

ABSTRAK

Setiap tahun dosen FMIPA UNPAR menerima *printout* jadwal mengawas ujian yang dibuat dengan excel. Walaupun datanya bersifat dijital, namun lebih baik jika data tersebut dibuat terstruktur sehingga dapat dibaca oleh mesin. Dari permalasahan tersebut maka dalam tugas akhir ini akan dibahas tentang pengembangan suatu program yang dapat membaca data excel tersebut dan merubahnya dalam format calendar dijital atau biasa disebut .ics, sehingga dosen dapat memasukan jadwal mengawas ujian kedalam gawai pribadinya. Program ini akan menggunakan tiga bahasa pemograman yaitu Apache POI, iCal4j, dan Java FX. Apache ROI bertugas membaca struktur data excel sehingga dapat dibaca oleh program, Java FX berfungsi sebagai *interface* program, dan Ical4j bertugas mengkonversi data yang telah dibaca program kedalam format iCalendar atau .ics .

Kata-kata kunci: Apache POI, iCal4j, Java FX

ABSTRACT

Every year lecture of FMIPA UNPAR get printout schedule of invigilation in excel format. Although, the excel data is digital data, but it's better if the data is structured so can be read by machine. Based of the problem above, then this thesis will be discussing about developing program that can read invigilation schedule in excel format and converting the schedule to digital calendar format or commonly called .ics, so that lecture can import the schedule to their personal gadget. This software will be build based on three programming language, that is Apache POI, iCal4j, and Java FX. Apache POI will be handle reading input data so the data can be imported to software, Java FX is functionate as interface of the software, and iCal4j handles of converting previously read data to iCalendar format or .ics .

Keywords: Apache POI, iCal4j, Java FX

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Jadwal mengawas ujian di FTIS merupakan hal yang rutin dipublikasikan kepada dosen setiap tengah dan akhir semester. Jadwal mengawas tersebut dipublikasikan oleh tata usaha. Sebelum dibagikan jadwal mengawas dibuat dalam file excel, lalu dicetak dan dibagikan kepada setiap dosen. Format jadwal mengawas ujian bersifat umum, dalam arti jadwal tersebut menyimpan nama semua dosen yang mengawas, nama mata kuliah, dan tempat pelaksanaan ujian. Dosen diharuskan melihat satu persatu baris untuk mendapatkan informasi mengenai waktu, nama matakuliah, dan tanggal dosen tersebut mengawas. Walalupun jadwal mengawas tersebut telah disusun dalam file excel, namun tetap dirasa kurang efisien karena tidak tersusun berdasarkan dosen yang mengawas dan memungkinkan terjadi kesalahan dalam membaca jadwal oleh dosen. iCalendar merupakan format file calendar pada komputer yang memudahkan penggunanya untuk mengirimkan undangan *meeting* dan melakukan pekerjaan bersama pengguna lainnya, via email, atau file *sharing* menggunakan ekstensi .ics . Format iCalendar sendiri telah didukung dan kompatibel dengan produk lainnya, seperti Google Calendar, Microsoft Outlook, Yahoo Calendar, Mozilla Thunderbird, Apple Calendar. Dari penjelasan diatas, tugas akhir ini dimaksudkan untuk memudahkan dosen untuk melihat jadwal mengawas ujian dimanapun dan kapanpun. Pengembangan perangkat lunak ini menggunakan tiga library yaitu Apache POI, Java FX, dan iCal4j.

1.2 Rumusan Masalah

Berdasarkan penjelasan di latar belakang, maka dapat dipaparkan rumusan masalah sebagai berikut :

1. Bagaimana perangkat lunak dapat membaca file excel jadwal mengawas ujian yang dibuat oleh TU ?
2. Bagaimana menampilkan jadwal ke layar ?
3. Bagaimana perangkat lunak mengkonversi jadwal mengawas menjadi iCalendar ?

1.3 Tujuan

Tujuan dari karya ilmiah ini dapat dipaparkan sebagai berikut:

1. Merancang PL yang mampu membaca file excel yang dipublikasikan oleh TU.
2. Membuat PL yang mampu menampilkan jadwal mengawas ujian yang telah dibaca ke layar.
3. Merancang PL dapat mengkonversi jadwal mengawas menjadi iCalendar.

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini agar dapat fokus pada pengembangan perangkat lunak konversi jadwal mengawas ujian :

- Diasumsikan TU menggunakan layout yang sama setiap tahunnya.

1.5 Metodologi Penelitian

Untuk menunjang penelitian maka diperlukan data untuk pengujian maupun pengetahuan teori yang akan diterapkan. Berikut adalah kegiatan yang akan dilakukan:

1. Melakukan studi pustaka mengenai
   - Apache POI
   - Java FX
   - iCal4j
   - Konsep MVC
   - Memperdalam Netbeans
2. Melakukan analisis pada file excel jadwal mengawas ujian yang dikeluarkan oleh TU.
3. Melakukan perancangan yang terdiri dari use case, diagram aktifitas, dan *user interface*.
4. Mengimplementasikan rancangan kedalam Netbeans.
5. Melakukan pengujian perangkat lunak dengan berbagai kemungkinan kasus.
6. Menyimpulkan atas serangkaian pengembangan yang dilakukan
7. Menulis dokumen skripsi

1.6 Sistematika Pembahasan

1. Bab 1 Pendahuluan
   Bab ini berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
2. Bab 2 Dasar Teori
   Bab ini berisi tentang teori dasar tentang Java FX, Apache POI, iCal4j, Konsep MVC.
3. Bab 3 Analisis
   Bab ini berisi tentang analisis kebutuhan dan fitur PL, diagram aktifitas PL, use case, diagram kelas.
4. Bab 4 Perancangan
   Bab ini berisi tentang perancangan kelas dalam PL dan gambaran *user interface*.
5. Bab 5 Implementasi dan Pengujian
   Bab ini berisi tentang penerapan hasil rancangan pada bab sebelumnya serta pengujian perangkat lunak.
6. Bab 6 Kesimpulan dan Saran
   Bab ini berisi tentang kesimpulan yang didapatkan dari hasil pengujian serta saran apabila ingin melanjutkan pengembangan ini.

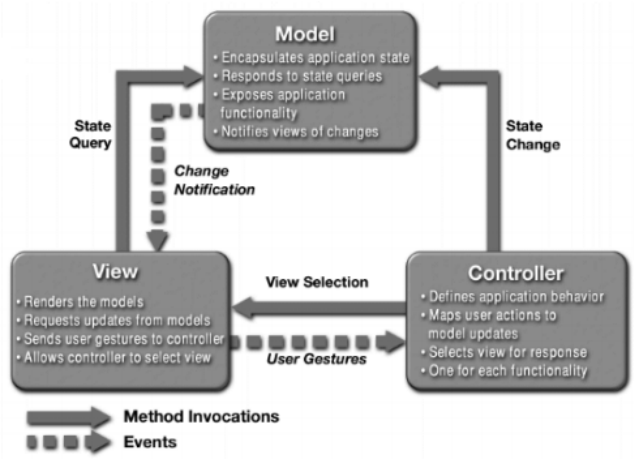
BAB 2

DASAR TEORI

Pada bab ini akan dijelsaskan mengenai konsep-konsep dasar pengukung BI, yaitu konsep MVC(*model view controller*), Java FX, Apache POI, iCal4j serta penjelasan singkat mengenai netbeans yang digunakan untuk membangun PL.

2.1 *Model View Controller*(MVC)

Setiap perangkat lunak memiliki pola bawaan dalam proses pembuatannya. Salah satu pola yang cukup terkenal adalah konsep MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, user interface, dan bagian yang menjadi kontrol aplikasi.



Gambar 2.1: Diagram MVC

Penjelasan MVC sebagai berikut [1]:

- *Model*
  Komponen ini berhubungan dengan proses yang ada dan juga terdapat di dalam sistem. Contoh model dalam skripsi ini adalah library Apache POI, iCal4j .

- *Controller*
  Komponen yang mengatur komponen dari model ke view maupun sebaliknya. Fungsi dari komponen ini adalah sebagai penghubung komponen model dan view.
  Contoh *Controller* di dalam skripsi ini adalah kode-kode dalam bahasa java yang meng-inherit dari kelas-kelas dalam *library* Apache POI dan iCal4j untuk membaca *file* excel dari TU dan mengkonversikannya dalam format iCalendar.

- *View*
  komponen yang berhubungan dengan *end user*. Bentuknya berupa tampilan yang akan digunakan oleh *user* atau sering disebut sebagai *user-interface*. Contoh *view* dalam pengerjaan skripsi ini adalah kode-kode yang dituliskan dalam Java FX berupa *file*

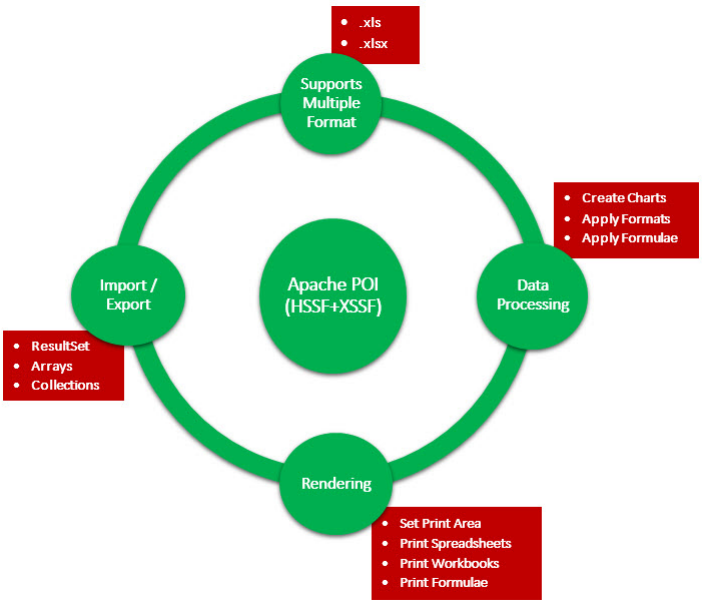
yang berekstensi fxml.
Banyak keuntungan dengan pendekatan MVC, yakni [1]:

1. Pembagian modul pengerjaan yang jelas untuk setiap orang di dalam suatu tim.
2. Kemudahan untuk *maintenance* dimasa depan.
3. Mempunyai fleksibilitas tinggi dalam pengembangan,

[1]

2.2 Apache POI

Apache POI pada hakikatnya merupakan API(*application programming interface*) yang populer digunakan oleh programmer untuk membuat, memodifikasi, dan menampilkan *file* MS Office menggunakan bahasa pemograman Java. Untuk mengingatkan bahwa API atau yang dalam bahasa indonesia disebut antarmuka pemograman aplikasi adalah sekumpulan perintah, fungsi, serta protokol yang digunakan oleh programmer saat membangun perangkat lunak [2].



Gambar 2.2: Kelebihan Apache POI(HSSF dan XSSF)
[2]

2.2.1 Komponen Apache POI

Dalam Apache POI terdapat kelas dan method yang dapat bekerja dengan semua file OLE2 yang merupakan format file yang dipakai oleh dokumen dari Microsoft Office , seperti MS Word, Excel, dll [3]. Berikut ini adalah komponen - komponen yang terdapat dalam Apache POI.[2]

1. **POIFS (*Poor Obfuscation Implementation File System*)** : Komponen ini merupakan faktor dasar dari semua elemen POI. Komponen ini dapat membaca file yang berbeda secara eksplisit.

[1] http://elmolya.blogspot.co.id/2010/03/belajar-mvc.html
[2] https://id.wikipedia.org/wiki/Antarmuka_pemrograman_aplikasi
[3] http://www.file-extensions.org/ole2-file-extension

2. **HSSF (*Horrible Spreadsheet Format*)** : Komponen ini digunakan untuk membaca dan menulis format xls dari file MS-Excel.

3. **XSSF (XML *Spreadsheet Format*)** : Komponen ini digunakan untuk membaca format file xlsx dari MS-Excel.

4. **HPSF (*Horrible Property Set Format*)** : Komponen ini digunakan untuk mengekstrak perangkat file yang dimiliki oleh MS-Office.

5. **HWPF (*Horrible Word Proccessor Format*)** : Komponen ini digunakan untuk menulis dan membaca ekstensi file doc dari MS-Word.

6. **XWPF (*XML Word Processor Format*)** : Komponen ini digunakan untuk membaca dan menulis ekstensi file docx dari MS-Word.

7. **HSLF (*Horrible Slide Layout Format*)** : Komponen ini digunakan untuk membaca, menulis, dan mengedit presentasi PowerPoint.

8. **HDGF (*Horrible Diagram Format*)** : komponen ini berisi kelas dan method dari file binary MS-Visio.

9. **HPBF (*Horrible Publisher Format*)** : Komponen ini digunakan untuk membaca dan menulis file MS-Publisher.

2.2.2 Kelas Inti Apache POI

Pada sub bab ini akan membahas sedikit intoduksi mengenai beberapa kelas dan method yang ada di Apache POI API yang merupakan bagian penting untuk bekerja dengan file excel mengunakan program Java.[2]

Workbook

org.apache.poi.ss.usermodel *package* merupakan *super-interface* dari semua kelas yang berhubungan dengan pembuatan atau*maintain* Excel workbook. Dua kelas yang mengimplementasikan *interface* diatas sebagai berikut:[2]

- **HSSFWorkbook** : Kelas ini mempunyai method yang dapat membaca dan menulis file Microsoft Excel dengan format .xls. Kelas ini kompatibel dengan MS-Office versi 97-2003.

- **XSSFWorkbook** : Kelas ini mempunyai method untuk menulis dan membaca Microsoft Excel dan OpenOffice xml dengan format .xls atau .xlsx. Kelas ini kompatibel dengan MS-Office versi 2007 atau versi barunya.

HSSFWorkbook

HSSFWorkbook merupakan *high-level class* dibawah **org.apache.xssf.usermodel** *package*. HSSFWorkbook juga mengimplementasikan antarmuka workbook yang digunakan oleh file Excel dalam format .xls. Berikut ini list dari bebeberapa method dan constructor dalam kelas ini.[2]

Class Constructor

No	Constructor dan Deskripsi
1	**HSSFWorkbook()** Membuat baru objek HSSFWorkbook.
2	**HSSFWorkbook(DirectoryNode directory, boolean perserveNodes)** Membuat objek HSSFWorkbook baru dalam direktori yang spesifik.
3	**HSSFWorkbook(DirectoryNode directory, POIFSFileSystem fs, boolean perserveNodes)** Memberikan sebuah objek POIDSFileSystem dan sebuah spesifik didalamnya, serta membuat objek SSFWorkbook untuk membaca sebuah workbook yang spesifik.
4	**HSSFWorkbook(java.io.InputStream s)** Membuat baru objek HSSFWorkbook menggunakan input stream.
5	**HSSFWorkbook(java.io.InputStream s, boolean preserveNodes)** Membangun sebuah POI *file system* disekeliling input stream.
6	**HSSFWorkbook(POIFSFileSystem fs)** Membangun sebuah objek HSSFWorkbook baru menggunakan sebuah objek POIFSFileSystem.
7	**HSSFWorkbook(POIFSFileSystem fs, boolean preserveNodes)** Memberikan sebuah objek POIFSFileSystem dan membuat HSSFWorkbook baru untuk membacu sebuah workbook spesifik.

Berikut ini penjelasan parameter yang sering dipakai pada constructor :

- **directory** : direktori proses dari POI filesystem
- **fs** : POI filesystem yang mengandung workbook stream.
- **preservenodes** : Opsional parameter yang memutuskan menjaga node lain, selain itu parameter ini menggunakan banyak memori seperti menyimpan semua POIFileSystem dalam memori(jika diset).

XSSFWorkbook

Kelas ini merepresentasikan baik *high* dan *low* level format file excel. XSSFWorkbook merupakan kelas yang berada dalam *package* **org.apache.xssf.usermodel** dan mengimplementasikan antarmuka workbook. Berikut ini list method dan constructor dalam kelas ini.[2]

Class Constructor

No	Constructor dan Deskripsi
1	**XSSFWorkbook()** Membuat baru objek XSSFWorkbook.
2	**XSSFWorkbook(java.io.File file)** Membangun sebuah objek XSSFWorkbook dari file yang diberikan.
3	**XSSFWorkbook(java.io.InputStream is)** Membangun sebuah object XSSFWorkbook dengan *buffering* semua input stream kedalam memory, dilanjutkan dengan membuka objek OPCPackage.
4	**XSSFWorkbook(java.lang.String path)** Membangun sebuah objek XSSFWorkbook dengan diberikan *full path* dari sebuah file.

Class Methods

No	Method dan Deskripsi
1	**CreateSheet()** Menciptakan sebuah XSSFSheet pada workbook, lalu menambahkan sheet, dan mengembalikannya dalam representasi *high level* .
2	**createSheet(java.lang.String sheetname)** Membuat sheet baru untuk workbook dan mengembalikannya dalam representasi *high level.*
3	**createFont()** Membuat font baru dan menambahkannya pada tabel font workbook.
4	**createCellStyle()** Membuat XSSFCellStyle Baru dan mmenambahkannya pada tabel style workbook.
5	**setPrintArea(int sheetIndex, int startColumn, int endColumn, int startRow,int endRow)** Menentukan area print dari kertas yang diberikan dengan parameter yang spesifik.

Sheet

Sheet merupakan sebuah interface dibawah package org.apache.ss.usermodel dan sheet merupakan super-interface dari semua kelas yang menciptakan *high* atau *low level spreadsheet* dengan nama yang spesifik. Jenis yang paling umum dari spreadsheet adalah worksheet yang direpresentasikan sebagai sebuah *grid* dari cell.[2]

HSSFSheet

HSSFSheet merupakan kelas dibawah *package***org.apache.poi.hssf.usermodel**. HSSFSheet dapat membuat excel spreadsheet dan memungkinkan untuk memformat style dari sheet dan data sheet.[2]

Class Constructor

No	Constructor dan Deskripsi
1	**HSSFSheet(HSSFWorkbook workbook)** Membuat baru HSSFSheet yang disebut HSSFWorkbook dalam pembuatan sheet baru .
2	**HSSFSheet(HSSFWorkbook workbook, InternalSheet sheet)** Membuat sebuah HSSFSheet yang mewakili objek sheet yang diberikan.

XSSFSheet

Kelas ini merupakan representasi dari *high level* excel spreadsheet. Kelas ini berada dibawah package org.apache.poi.hssf.usermodel.[2]

Class Constructor

No	Constructor dan Deskripsi
1	**XSSFSheet()** Membuat baru XSSFSheet yang disebut XSSFWorkbook dalam pembuatan sheet baru .
2	**XSSFSheet(PackagePart part, PackageRelationship rel)** Membuat sebuah XSSFSheet yang mewakili bagian package dan *relationship.*

Class Method

No	Constructor dan Deskripsi
1	**addMergedRegion(CellRangeAddress region))** Menambahkan gabungan wilayah dari cell.(beberapa cell menjadi satu) .
2	**autoSizeColumn(int column)** Menyesuaikan lebar kolom agar sesuai dengan isinya.
3	**iterator()** Method ini alias rowIterator() untuk memungkinkan foreach loop .
4	**addHyperlink(XSSFHyperlink hyperlink)** Mendaftarkan sebuah hyperlink kedalam koleksi hyperlink yang ada di sheet.

Row

Row merupakan interface berada dibawah *package* **org.apache.poi.ss.usermodel**. Row ini digunakan untuk *high-level representation* dari sebuah row pada sebuah spreadsheet. Row juga merupakan super-interface dari semua kelas yang mewakili row dalam POI *Library*.[2]

XSSFRow

XSSFRow merupakan sebuah kelas dibawah *package* **org.apache.poi.xssf.usermodel** dan mengimplementasi Row interface. Selain itu, kelas ini dapat membuat row dalam sebuah spreadsheet. List dibawah ini merupakan method dan constructors pada kelas ini.[2]

Class Method

No	Deskripsi
1	**createCell(int columnIndex)** Membuat cell baru dalam baris.
2	**setHeight(short height)** Mengatur tinggi dalam satuan short.

Cell

Cell merupakan interface yang berada dibawah *package* **org.apache.poi.ss.usermodel**. Cell merupakan sebuah super-interface dari semua kelas yang mewakili cell dalam baris sebuah spreadsheet.

Cell dapat beruba berbagai atrubut seperti *blank, numeric, date, error,* dll. Sebelum ditambahkan ke baris cell memiliki nomer tersendiri(dari mulai 0).[2]

XSSFCell

Kelas ini berada dibawah *package* **org.apache.poi.xssf.usermodel**. Kelas ini mewakili cell interface. XSSFCell adalah *high-level representation* cell dalam row dari sebuah spreadsheet.[2]

Ringkasan Tipe Cell

List dibawah ini adalah sebagian *field* dari kelas XSSFCell beserta deskripsinya.

Tipe Cell	Deskripsi
CELL_TYPE_BLANK	Representasi cell kosong
CELL_TYPE_BOOLEAN	Representasi cell Boolean (True atau False)
CELL_TYPE_ERROR	Representasi nilai error dari cell
CELL_TYPE_FORMULA	Representasi dari hasil sebuah formula dalam cell
CELL_TYPE_NUMERIC	Representasi dari data numerik dalam cell
CELL_TYPE_STRING	Representasi dari String(teks) dalam cell

Class Method

No	Deskripsi
1	**setCellStyle(CellStyle style)** Mengatur style untuk cell.
2	**setCellType(int cellType)** Mengatur tipe cell (numeric, formula, atau String).
3	**setCellValue(boolean value)** Mengatur nilai bolean dalam sebuah cell.
4	**setCellValue(java.util.Calendar value)** Mengatur nilai tanggal dari cell .
5	**setCellValue(double value)** Mengatur nilai numerik dari cell.
6	**setCellValue(java.lang.String str)** Mengatur nilai String dari cell.
7	**setHyperlink(Hyperlink hyperlink)** Menambahkan sebuah hyperlink kedalam cell

XSSFCellStyle

XSSFCellStyle merupakan sebuah kelas yang berada dibawah *package* **org.apache.poi.usermodel**. kelas ini memberikan infomarsi yang mungkin mengenai format konten pada suatu cell dari spreadsheet. Kelas ini juga memberikan opsi untuk merubah format tersebut. Kelas ini mewakili CellStyle interface.[2]

Ringkasan Cell Style

List dibawah ini adalah sebagian *field* yang diwariskan dari CellStyle interface.[2]

Nama Field	Deskripsi Field
ALIGN_CENTER	Rata tengah konten cell
ALIGN_CENTER_SELECTION	Posisi seleksi tengah horizontal
ALIGN_FILL	Mencocokan ukuran konten cell
ALIGN_JUSTIFY	Mencocokan ukuran konten cell terhadap lebarnya
ALIGN_LEFT	Rata kiri konten cell
ALIGN_RIGHT	Rata kanan konten cell
BORDER_DASH_DOT	Cell style dengan garis dan titik
BORDER_DOTTED	Cell style dengan border titik
BORDER_DASHED	Cell Style dengan border garis
BORDER_THICK	Cell Style dengan border tebal
BORDER_THIN	Cell Style dengan border tipis
VERTICAL_BOTTOM	Posisi konten cell vertikal kebawah
VERTICAL_CENTER	Posisi konten cell vertikal ketengah
VERTICAL_JUSTIFY	Posisi konten cell sejajar secara vertikal
VERTICAL_TOP	Posisi selaras keatas secara vertikal

Class Constructor

No	Constructor dan Deskripsi
1	**XSSFCellStyle(int cellXfId, int cellStyleXfId, StylesTable stylesSource, ThemesTable theme)** Menciptakan cell style dengan bagian yang sudah disediakan.
2	**XSSFCellStyle(StylesTable stylesSource)** Membuat cell Style kosong.

Class Method

No	Method dan Deskripsi
1	**setAlignment(short align)** Mengatur style secara horizontal untuk cell.
2	**setBorderBottom(short border)**
3	**setBorderColor(XSSFCellBorder.BorderSide side, XSSFColor color)** Mengatur warna untuk border yang dipilih.
4	**setBorderLeft(Short border)** Mengatur tipe border untuk border kiri dari cell .
5	**setBorderRight(short border)** Mengatur tipe border untuk border kanan dari cell .
6	**setBorderTop(short border)** Mengatur tipe border untuk border atas dari cell
7	**setFillBackgroundColor(XSSFColor color)** Mengatur latar belakang warna yang diwakili oleh nilai XSSFColor
8	**setFillForegroundColor(XSSFColor color)** Mengatur latar depan warna yang diwakili oleh nilai XSSFColor
9	**setFillPattern(short fp)** Menentukan isi informasi cell dengan pola dan warna solid
10	**setFont(Font font)** Mengatur font
11	**setRotation(short rotation)** Mengatur derajat rotasi pada teks dalam cell.
12	**setVerticalAlignment(short align)** Menetapkan tipe posisi vertical pada cell

HSSFColor

HSSFColor merupakan sebuah kelas dibawah *package* **org.apache.poi.hssf.util.package**. Kelas ini memberikan warna berbeda terhadap *nested class*. Biasanya *nested class* diwakili dengan menggunakan index masing-masing. Kelas ini mengimplementasikan Color interface.[2]

Nested Class

Semua *nested class* dari kelas ini adalah static dan setiap kelas memiliki index masing-masing. Warna kelas ini digunakan pada format cell seperti konten cell, border, latar depan(*foreground*), dan latar belakang(*background*). List dibawah ini merupakan sebagian dari *nested class*.[2]

No	Nama Kelas(warna)
1	HSSFColor.AQUA
2	HSSFColor.AUTOMATIC
3	HSSFColor.BLACK
4	HSSFColor.BLUE
5	HSSFColor.BRIGHT_GREEN
6	HSSFColor.BRIGHT_GRAY
7	HSSFColor.CORAL
8	HSSFColor.DARK_BLUE
9	HSSFColor.DARK_GREEN
10	HSSFColor.SKY_BLUE
11	HSSFColor.WHITE
12	HSSFColor.YELLOW

Class Method

Hanya satu method dalam kelas ini yang penting dan digunakan untuk mendapat nilai indeks.

No	Method dan Deskripsi
1	**getIndex()** Method ini digunakan untuk mendapatkan nilai indeks dari sebuah *nested class*.

XSSFColor

XSSFColor merupakan sebuah kelas dibawah *package* **org.apache.poi.xssf.usermodel**. Kelas ini mewakili warnma pada spreadsheet. Kelas ini mengimplementasika interface warna. List dibawah ini merupakan beberapa method XSSFColor dan constructornya.[2]

Class Constructor

No	Constructor dan Deskripsi
1	**XSSFColor()** Menciptakan *instance* baru dari XSSFColor.
2	**XSSFColor(byte[] rgb)** Membuat *instance* baru dari XSSFColor menggunakan RGB.
3	**XSSFColor(java.awt.Color clr))** Membuat *instance* baru dari XSSFColor menggunakan kelas warna dari *awt package*.

Class Methods

No	Method dan Deskripsi
1	**setAuto(boolean auto)** Mengatur sebuah nilai boolean untuk mengindikasikan bahwa ctColor bersifat otomatis dan bergantung pada ctColor sistem.
2	**setIndexed(int indexed)** Mengatur nilai indeks ctColor sebagai sistem ctColor.

XSSFFFont

XSSFFont merupakan kelas dibawah *package* **org.aoache.poi.xssf.usermodel**. Kelas ini mengimplementasikan *Font interface* dan oleh sebab itu kelas ini dapat menangani font berbeda pada sebuah workbook.[2]

Class Constructor

No	Constructor dan Deskripsi
1	**XSSFFont()** Menciptakan *instance* baru dari XSSFFont.

Class Methods

No	Method dan Deskripsi
1	**setBold(boolean bold)** Mengatur sebuah nilai boolean untuk atribut 'bold'.
2	**setColor(short color)** Mengatur nilai indeks warna untuk font.
3	**setColor(XSSFColor color)** Mengatur warna untuk font dalam standar nilai warna Alpha RGB.
4	**setFontHeight(short height)** Mengatur tinggi font dalam poin.
5	**setFontName(java.lang.String name)** Mengatur nama dari font.
6	**setItalic(boolean italic)** Mengatur nilai boolean pada poperti 'italic'.

XSSFHyperlink

XSSFHyperlink merupakan kelas dibawah *package* **org.apache.poi.xssf.usermodel**. Kelas ini mengimplementasikan *Hyperlink interface*. Kelas ini digunakan untuk mengatur sebuah hyperlink pada konten cell dalam sebuah spreadsheet.[2]

Field

Field dalam kelas ini akan didefinisikan sebagai berikut. Field disini dalam arti tipe dari hyperlink yang dipakai.

Field	Deskripsi
LINK_DOCUMENT	Dipakai untuk menghubungkan dengan dokumen lainnya
LINK_EMAIL	Digunakan untuk menghubungkan dengan email
LINK_FILE	Digunakan untuk menghubungkan dengan file lain dalam berbagai format
LINK_URL	Digunakan untuk menghubungkan dengan URL *website*

Class Methods

No	Method dan Deskripsi
1	**setAddress(java.lang.String address)** Alamat Hyperlink.

XSSFCreationHelper

XSSFCreationHelper merupakan kelas dibawah *package* **org.apache.poi.xssf.usermodel**. Kelas ini mengimplementasikan *CreationHelper interface*. Kelas ini digunakan sebagai bentuk kelas pendukung untuk *formula evaluation* dan penyusun hyperlink.[2]

Class Methods

No	Method dan Deskripsi
1	**createFormulaEvaluator()** Membuat sebuh *instance* XSSFFormulaEvaluator, objek yang dapat mengevaluasi formula dalam cell.
2	**createHyperlink(int type)** Membuat sebuah XSSFHyperlink baru.

XSSFPrintSetup

XSSFPrintSetup merupakan kelas dibawah *package* **org.apache.poi.xssf.usermodel**. Kelas ini mengimplementasikan *PrintSetup interface*. Kelas ini digunakan untuk mengatur ukuran cetak pada halaman, wilayah cetak, opsi, dan pengaturan.[2]

Class Methods

No	Method dan Deskripsi
1	**setLandscape(boolean ls)** Mengatur sebuah nilai boolean yang dapat mengijinkan atau menolak *landscape printing*.
2	**setLeftToRight(boolean ltor)** Mengatur perintah ke kiri, kanan, atas, atau bawah ketika proses cetak.
3	**setPaperSize(short size)** Mengatur ukuran kertas.

2.3 iCal4j

iCal4j merupakan *Java library* yang digunakan untuk membaca dan menulis data iCalendar yang didefinisikan dalam RFC2445. iCalendar standar menyediakan sebuah format data yang umumnya digunakan untuk menyimpan informasi tentang spesifikasi kalender seperti acara, pertemuan, *to-do list*, dll. Semua *tool* kalender yang populer, seperti Lotus Notes, Outlook, Google Calendar, Apple iCal mensupport standar iCalendar.[3]

Sebagai pengurai kalender dan *object model*, iCal4j memudahkan untuk memodifikasi data kalender yang sudah ada atau membuat model data baru. Validasi juga diperlukan untuk memastikan data terjaga baik dan konsisten dengan spesifikasi yang diperlukan.[3]

2.3.1 Komponen iCal4j

Berikut ini merupakan kumpulan *package* yang ada dalam iCal4j.[3]

No	Package dan Deskripsi
1	**net.fortuna.ical4j.data** Menyediakan berbagai tipe RFC2445 input, output, serta fungsi parsing.
2	**net.fortuna.ical4j.filter** Aturan untuk menyaring list komponen yang digunakan, *properties*, maupun parameter yang digunakan.
3	**net.fortuna.ical4j.model** Berisikan komponen utama yang digunakan untuk mendefinisikan model iCalendar.
4	**net.fortuna.ical4j.model.component** Berisikan respresentasi tipe yang digunakan dalam komponen model iCalendar.
5	**net.fortuna.ical4j.model.parameter** Berisikan respresentasi tipe yang digunakan dalam parameter model iCalendar.
6	**net.fortuna.ical4j.model.property** Berisikan respresentasi tipe yang digunakan dalam properti model iCalendar.
7	**net.fortuna.ical4j.transform** Berisikan perubahan tipe yang digunakan komponen model iCalendar sesuai RFC2446 .
8	**net.fortuna.ical4j.util** Berisikan tipe utilitas yang mendukung fungsi dari iCal4j.

2.3.2 Kelas Inti dari iCal4j

Pada bagian ini *package* yang ditulis di sub bab sebelumnya akan dijelaskan lebih dalam apa kegunaannya.[3]

2.3.3 net.fortuna.ical4j.data

Ringkasan Interface

No	Method dan Deskripsi
1	**CalendarParser** Pelaksana yang menyediakan fungsi parsing pada iCalendar.
2	**ContentHandler** Pelaksana yang menyediakan fungsi yang berlaku selama parsing aliran data dari iCalendar(misalnya membangun model objek).

Ringkasan Kelas

No	Method dan Deskripsi
1	**AbstractOutputter** kelas dasar untuk model *output*.
2	**CalendarBuilder** Parsing dan memnagan sebuah model iCalendar dari input stream.
3	**CalendarOutputter** Menuliskan sebuah model iCalendar pada output stream.
4	**CalendarParserFactory** Menyediakan akses pada CalenderParser yang telah dikonfigurasi.
5	**CalendarParserImpl** Implementasi *default* dari CalenderParser.
6	**DefaultCalendarParserFactory** Implementasi *default* dari CalenderParser.
7	**FoldingWriter** Fungsi penulisan yang mendukung penulisan iCalendar berlipat.
8	**HCalendarParser** Menguraikan dokumen XHTML yang meliputi data kalender, ditandai dengan mikroformat hCalendar.
9	**HCalendarParserFactory** kumpulan parser untuk mikroformat hCal
10	**UnfoldingReader** Fungsi membaca bagian iCalendar yang wajib dibaca.

2.3.4 net.fortuna.ical4j.filter

Ringkasan Interface[3]

No	Method dan Deskripsi
1	**Rule** Pelaksana yang menentukan apakah suatu objek tertentu diklasifikasikan sebagai pasangannya dapat dijadikan sebagai filter lampiran.

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**DateInRangeRule** Mengimplementasikan Rule.
2	**Filter** Melakukan filtering dari seperangkat aturan. Sebuah filter dapat menentukan apakah setidaknya satu aturan tersebut cocok atau tidak.
3	**HasPropertyRule** Sebuah aturan yang mencocokan komponen memuat poperti yang spesifik.
4	**PeriodRule<T extends Component>** Sebuah aturan yang mencocokan komponen terjadi atau tidak dalam jangka waktu yang ditentukan.

2.3.5 net.fortuna.ical4j.model

Ringkasan Interface[3]

No	Method dan Deskripsi
2	**Escapable** Pelaksana yang mengkonversi ke/dari nilai string kedalam bentuk iCalendar.
4	**ParameterFactory<T extends Parameter>** Pelaksana yang menyediakan pembuatan *service* parameter.
5	**PropertyFactory<T extends Property>** Membuat properti iCalendar.
6	**TimeZoneRegistry** Menyediakan daftar definisi wilayah yang berlaku untuk digunakan objek iCalendar.

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**AddressList** Mendefinisikan list dari alamat pada iCalendar.
2	**Calendar** Mendefinisikan kalendar pada iCalendar.
3	**CalendarDateFormatFactory** Membuat objek dateFormat untuk optimisasi pola tanggal pada iCalendar.
4	**Date** Representasi dari objek DATE sesuai RFC5445.
5	**DateList** Representasi list tanggal dari iCalendar.
6	**DateTime** Representasi dari objek DATE-TIME sesuai RFC5445.
7	**LocationTypeList** Menetapkan sebuah list tipe lokasi dari iCalendar.
8	**NumberList** Menetapkan list dari nomer.
9	**Parameter** Mendefinisikan parameter.
10	**Period** Mendefinisikan tenggat waktu.
11	**Property** Mendefinisikan properti dari iCalendar.
12	**Time** Sebuah tipe yang merepresentasikan nilai waktu pada iCalendar.
13	**TimeZone** Implementasi zona waktu java.
14	**WeekDay** Mendefinisikan hari dalam seminggu dengan diimbangi terkait dengan kejadian bulanan atau tahunan.

2.3.6 net.fortuna.ical4j.model.component

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**Available** Mendefinisikan komponen tersedia di iCalendar.
2	**Daylight** Mendefinisikan waktu siang dalam zona waktu.
3	**Standard** Mendefinisikan komponen zona waktu standar.
4	**Standard.Factory VAlarm** Mendefinisikan komponen VALARM pada iCalendar.
5	**Standard.Factory VAvailability** Mendefinisikan komponen VAvailability pada iCalendar.
6	**VAvailability.Factory VEvent** Mendefinisikan komponen VEvent pada iCalendar.
7	**VEvent.Factory VFreeBusy** Mendefinisikan komponen VFreeBusy pada iCalendar.
8	**VFreeBusy.Factory VJournal** Mendefinisikan komponen VJournal pada iCalendar.
9	**VJournal.Factory VTimeZone** Mendefinisikan komponen VTimeZone pada iCalendar.
10	**VTimeZone.Factory VToDo** Mendefinisikan komponen VToDo pada iCalendar.
11	**VToDo.Factory VVenue** Mendefinisikan komponen VVenue pada iCalendar.

2.3.7 net.fortuna.ical4j.model.parameter

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**Abbrev** Mendefinisikan parameter singkatan.
2	**AltRep** Mendefinisikan alternatif representasi parameter teks.
3	**Cn** Mendefinisikan parameter dengan nama umum .
4	**CuType** Mendefinisikan tipe calender user.
5	**DelegatedFrom** Mendefinisikan parameter delegator.
6	**DelegatedTo** Mendefinisikan parameter delegasi.
7	**Dir** Mendefinisikan parameter referensi directory entri.
8	**Encoding** Mendefinisikan parameter inline Encoding.
9	**VJournal.Factory VTimeZone** Mendefinisikan komponen VTimeZone pada iCalendar.
10	**FbType** Mendefinisikan tipe parameter *free/busy*.
11	**FmtType** Mendefinisikan parameter tipe format.
12	**Language** Mendefinisikan parameter bahasa.
13	**Member** Mendefinisikan parameter list group peserta.
14	**PartStat** Mendefinisikan parameter satus partisipasi.
15	**Range** Mendefinisikan parameter identifikasi perulangan .
16	**Related** Mendefinisikan parameter pemicu alarm.
17	**RelType** Mendefinisikan parameter tipe hubungan.
18	**Rsvp** Mendefinisikan parameter RSVP.
19	**ScheduleAgent** Mendefinisikan penjadwalan.
20	**ScheduleStatus** Mendefinisikan status penjadwalan.
21	**SentBy** Mendefinisikan parameter pengirim.
22	**Type** Mendefinisikan parameter tipe.
23	**TzId** Mendefinisikan parameter zona waktu.
24	**Value** Mendefinisikan parameter nilai tipe data.
25	**Vvenue** Mendefinisikan parameter Vvenue.
26	**XParameter** Mendefinisikan parameter pemicu alarm.
27	**Related** Mendefinisikan penambahan parameter.

2.3.8 net.fortuna.ical4j.model.property

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**Action** Mendefinisikan aksi dari komponen properti iCalendar .
2	**Attach** Mendefinisikan lampiran dari komponen properti iCalendar.
3	**Attendee** Mendefinisikan kedatangan dari komponen properti iCalendar.
4	**BusyType** Mendefinisikan tipe sibuk pada komponen properti.
5	**Categories** Mendefinisikan kategori pada komponen properti.
6	**Clazz** Mendefinisikan kelas pada komponen properti.
7	**Comment** Mendefinisikan komen pada komponen properti.
8	**Completed** Mendefinisikan status selesai pada komponen properti.
9	**Contact** Mendefinisikan kontak pada komponen properti.
10	**Country** Mendefinisikan negara pada komponen properti.
11	**Created** Mendefinisikan pembuatan pada komponen properti.
12	**Description** Mendefinisikan deskripsi pada komponen properti.
13	**DtEnd** Mendefinisikan DtEnd pada komponen properti.
14	**DtStamp** Mendefinisikan DtStamp pada komponen properti.
15	**DtStart** Mendefinisikan DtStart pada komponen properti.
16	**Due** Mendefinisikan Due pada komponen properti.
17	**Duration** Mendefinisikan Durasi pada komponen properti.
18	**LastModified** Mendefinisikan terakhir dirubah pada komponen properti.
19	**Location** Mendefinisikan lokasi pada komponen properti.
20	**LocationType** Mendefinisikan tipe lokasi pada komponen properti.
21	**Name** Mendefinisikan nama pada komponen properti
22	**PercentComplete** Mendefinisikan progress pada komponen properti.
23	**Priority** Mendefinisikan prioritas pada komponen properti.
24	**RelatedTo** Mendefinisikan berhubungan dengan siapa pada komponen properti.
25	**Status** Mendefinisikan status pada komponen properti.
26	**StreetAddress** Mendefinisikan alamat pada komponen properti.
27	**Summary** Mendefinisikan ringkasan pada komponen properti.
28	**Url** Mendefinisikan url pada komponen properti.

2.3.9 net.fortuna.ical4j.model.transform

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**PublishTransformer** Merubah kalendar untuk dipublikasikan.
2	**Transformer** *Base Class* untuk transforasi kalender.

2.3.10 net.fortuna.ical4j.model.transform

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**PublishTransformer** Merubah kalendar untuk dipublikasikan.
2	**Transformer** *Base Class* untuk transforasi kalender.

2.3.11 net.fortuna.ical4j.model.util

Ringkasan Interface[3]

No	Method dan Deskripsi
1	**HostInfo** Menyediakan informasi host berupa*paltform* yang independen.

Ringkasan Kelas[3]

No	Method dan Deskripsi
1	**Calendars** Method utility untuk bekerja dengan kalender.
2	**Dates** Mengimplementasikan koleksi dari method utility yang relevan untuk memproses tanggal.
3	**Numbers** kelas utility untuk memproses nomer.
4	**Strings** method utility yang bekerja dengan parameter.
5	**TimeZones** method utility yang relevan dengan zona waktu Java.

2.4 Java FX

Java FX merupakan seperangkat grafis dan paket media yang memungkinkan pengembang untuk merancang, membuat, menguji, debug , dan dapat beroperasi secara konsisten di seluruh platform yang beragam.
Java FX dapat membuat berbagai macam aplikasi dari mulai aplikasi jaringan dasar hingga antar muka yang memiliki fitur audio, video, grafik, dan animasi

BAB 3

ANALISIS

Pada bab ini, akan dijelaskan mengenai analisis Input dan fitur perangkat lunak, Diagram pengembangan perangkat lunak, *use case* dari perangkat lunak serta diagram aktifitas dari perangkat lunak.

3.1 Analisis Input

3.1.1 Analisis File Excel Jadwal Mengawas Ujian

Sub bab ini akan membahas analisis file excel yang dikeluarkan oleh TU.
TU FTIS mengeluarkan jadwal setiap tahunnya yang dibagikan kepada dosen FTIS. berikut ini contoh file excel yang dikeluarkan oleh TU.

No.	Hari, Tgl.	Jam	Sem.	PS	Nama Mata kuliah	Ruang					
						9120	9121	9122	10316	10317	10323
1	Senin, 14 Mrt. 2016	08.00-10.00	2	MA	Kalkulus 2	Mariskha	Pascal		Ferry	Maria	
2	Senin, 14 Mrt. 2016	08.00-10.00	6p	IT	Keamanan Informasi, Reologi						
3	Senin, 14 Mrt. 2016	11.00-13.00	4	FI	Elektronika 2, Analisis Real	Vero	Gede	Owen	Elok	Iwan	Philips
4	Senin, 14 Mrt. 2016	11.00-13.00	4	IT	Manajemen Informasi dan Basis Data						
5	Senin, 14 Mrt. 2016	14.00-16.00	2	IT	Matematika Informatika	Maria	Haryanto	Janto			
6	Senin, 14 Mrt. 2016	14.00-16.00	2	IT	Matematika Informatika						
7	Selasa, 15 Mrt. 2016	08.00-10.00	2	FI	Fisika Dasar 2, Pemr. Apl. Bergerak				Reinard	Gede	Liem
8	Selasa, 15 Mrt. 2016	08.00-10.00	p	MA	Komputasi Keuangan						
9	Selasa, 15 Mrt. 2016	11.00-13.00	6	FI	Mekanika Kuantum, Pers. Dif. Biasa				Bagoes	Iwan	Sylvia
10	Selasa, 15 Mrt. 2016	11.00-13.00	6p	IT	Jaringan Syaraf Tiruan						
11	Selasa, 15 Mrt. 2016	14.00-16.00	8p	IT	Majemen Pengetahuan	Taufik	Janto				
12	Selasa, 15 Mrt. 2016	14.00-16.00	8p	IT	Majemen Pengetahuan						
13	Rabu, 16 Mrt. 2016	08.00-10.00	4	FI	Fisika Matematika 4, Proy. Inform., Logika Inf.	Risti	Pascal	Mariskha	Philips	Vania	Benny
14	Rabu, 16 Mrt. 2016	08.00-10.00	p	MA	Pengantar Matematika Asuransi						
15	Rabu, 16 Mrt. 2016	10.00-12.00	4	IT	Desain dan Analisis Algoritma	Shift 1: Luciana, Flaviana					
16	Rabu, 16 Mrt. 2016	12.00-14.00	4	IT	Desain dan Analisis Algoritma	Shift 2: Pascal , Luciana					
17	Rabu, 16 Mrt. 2016	11.00-13.00	2	MA	Aljabar Matriks	Taufik	Sylvia		Heni	Rusli	Haryanto
18	Rabu, 16 Mrt. 2016	11.00-13.00	8/8	FI/IT	Etika Profesi						
19	Rabu, 16 Mrt. 2016	14.00-16.00	6	IT	Penulisan Ilmiah	Lab. : Anung, Vania					

PGW_UTS | PGW_UAS

Gambar 3.1: Jadwal mengawas ujian FTIS

Dari gambar **3.1** dapat anlisa bahwa :

1. Terdapat 7 kolom pada file jadwal tersebut yaitu :
   (a) No
   (b) Hari, tanggal
   (c) Jam
   (d) Semester
   (e) Peserta Studi
   (f) Nama Mata Kuliah
   (g) Ruangan (09120, 9121, 9122, 10316, 10317, 10323)

2. Pada kolom semster terdapat hurf p yang menunjukan ujian tersebut merupakan mata kuliah pilihan pada semester tertentu.
3. Pada Kolom peserta studi dapat di isi lebih dari satu jurusan ditandai dengan tanda '\'
4. Terdapat singkatan seperti :
   (a) Tgl. pada *header* merupakan singkatan dari tanggal.
   (b) Sem. pada *header* merupakan singkatan dari semester.
   (c) PS pada *header* merupakan singkatan dari peserta studi.
   (d) MA pada kolom PS merupakan singkatan dari Matematika
   (e) FI pada kolom PS merupakan singkatan dari Fisika
   (f) IT pada kolom PS merupakan singkatan dari Informatika
   (g) Mrt pada kolom Hari, Tgl. merupakan singkatan dari Maret
5. kolom *header* terdiri dari 2 baris, ada yang menggunakan *merge*, ada juga yang dibagi menjadi 2 bagian.
6. Kolom Ruang merupakan gabungan dari 6 kolom, dan dibawahnya tertulis nomer ruang kelas ujian.
7. Selama satu hari jatah dosen mengawas yaitu selama 2 jam, hal itu ditandai dengan *merge* 2 baris dengan nama dosen didalamnya.
8. Ruangan yang tidak terpakai ditandai dengan warna abu bergaris - garis.
9. Khusus untuk matakuliah pemograman tempat ujian di lab dan ada pembagian jadwal mengawas shift 1 dan shift 2.

3.1.2 Analisis Fitur Perangkat Lunak

Perangkat Lunak ini akan memiliki fitur sebagai berikut :

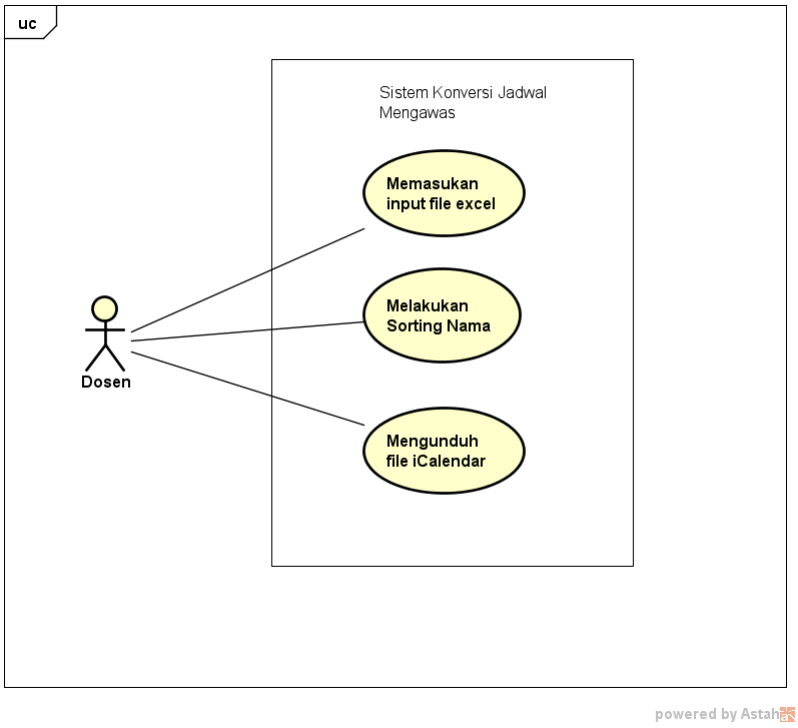
1. *Tool* ini dapat menerima dan membaca*input* file excel jadwal mengawas ujian yang dikeluarkan TU FTIS.
2. *Tool* ini dapat mengubah file excel menjadi iCalendar.
3. File iCalendar dapat di unduh oleh pengguna.
4. Pengguna dapat melakukan *sort* sesuai dengan nama yang di inginkan.

3.2 Permodelan Tool

Berikut diagram use case berserta skenario yang tertera pada gambar **3.2**

1. Skenario Memasukan input file excel

   Deskripsi : Kegiatan memasukan input file excel.
   Aktor : Dosen
   Prakondisi : -
   Skenario :
   - Dosen memasukan file excel mengawas ujian yang keluarkan oleh TU



Gambar 3.2: Diagram use case *tool* konversi jadwal mengawas ujian

2. Skenario Melakukan Sorting nama

   Deskripsi : Kegiatan mensorting jadwal mengawas.

   Aktor : Dosen

   Prakondisi : -

   Skenario :

   – Dosen dapat melakukan sorting nama dari jadwal ujian yang telah berupa iCal sesuai nama yang di inginkan.

3. Skenario Mengunduh File iCal

   Deskripsi : Kegiatan Mengunduh file iCal.

   Aktor : Dosen

   Prakondisi : -

   Skenario :

   – Dosen mengunduh file iCal yang telah dikonversi oleh *tool*

BAB 4

PERANCANGAN

Berdasarkan analisis pada Bab 3, pada bab ini akan dipaparkan mengenai perancangan lojik basisdata, perancangan fisik basisdata, perancangan kelas MVC pada ODOO, diagram kelas, perancangan antar muka BI *tool*, rancangan *method-method* utama.

4.1 Perancangan Lojik Basisdata

Setelah pada bab 3 dijelaskan mengenai pemodelan basis data BI *tool*, pada bagian ini akan digambarkan secara jelas rancangan struktur basisdata dari BI *tool*. Rancangan basisdata ini merupakan impementasi dari skema *star* menempatkan ba_etl_detail sebagai *fact table*. Keuntungan menerapkan skema *star* adalah :

1. Lebih mudah dipahami sehingga memudahkan pengguna jika ingin mengembangkan BI *tool* ini lebih lanjut.

2. Secara teknis memudahkan *developer* jika ingin mengembangkan fitur baru pada BI *tool* ini.

3. Efisien dalam mengakses data sehingga memudahkan untuk menampilkannya.

Gambar 4.1: Diagram Relational pada Basisdata BI *tool*

4.2 Perancangan Fisik Basisdata

Pada bagian sebelumnya telah digambarkan rancangan lojik basisdata BI *tool*, berikut ini penjelasan detil kolom-kolom dalam tabel tersebut.

Tabel 4.1: Tabel *ba_ bussiness*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
name	VARCHAR	200	NOT NULL

Tabel 4.2: Tabel *ba_ etl_ header*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
bussiness_id	MANY2ONE ba_business		NOT NULL; FOREIGN KEY
name	VARCHAR	200	NOT NULL
is_active	BOOLEAN		DEFAULT TRUE

Tabel 4.3: Tabel *ba_ etl_ detail*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_header		NOT NULL; FOREIGN KEY
process_type	ENUM(read_data, format_data, lookup, sort_data, derived_column, script, merge, aggregate, sort_data, save_data)		NOT NULL
name	VARCHAR		NOT NULL
sequence	INTEGER		NOT NULL
read_data_source_type	ENUM(CSV)	20	
read_data_csv_trial_file	BINARY		
read_data_csv_delimiter	VARCHAR	2	
read_data_csv_auto_read	BOOLEAN		
read_data_csv_auto_read_path	VARCHAR		
read_data_csv_is_header_present	BOOLEAN		DEFAULT TRUE
script_text	TEXT		
save_data_model_id	MANY2ONE ir_model		
lookup_source_key_column	MANY2ONE ba_etl_columns		
lookup_target_key_column	varchar	64	
lookup_from	ENUM(fixed, existing_table)		
lookup_model_id	MANY2ONE ir_model		
lookup_model_key_column	VARCHAR	64	
lookup_model_value_column	VARCHAR	64	
merge_source_type_1	ENUM(CSV)		
merge_source_type_2	ENUM(CSV)		
merge_csv_trial_file_1	BINARY		
merge_csv_trial_file_2	BINARY		
merge_csv_delimiter_1	VARCHAR	3	
merge_csv_delimiter_2	VARCHAR	3	
merge_type	ENUM(left, right, inner)		

Tabel 4.4: Tabel *ba_ etl_ columns*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_header		FOREIGN KEY
name	VARCHAR	64	

Tabel 4.5: Tabel *ba_etl_columns_mapping*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		NOT NULL; FOREIGN KEY
source_column	VARCHAR	64	
detination_column	VARCHAR	64	

Tabel 4.6: Tabel *ba_etl_format_data_lines*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		NOT NULL; FOREIGN KEY
column_key	MANY2ONE ba_etl_columns		NOT NULL
column_type	ENUM(char, integer, float, date, datetime, boolean, selection)		
string_length	INTEGER		
column_format	VARCHAR	50	
column_default	VARCHAR	500	

Tabel 4.7: Tabel *ba_etl_sort_data_columns*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		NOT NULL; FOREIGN KEY
sequence	INTEGER		NOT NULL
column_key	MANY2ONE ba_etl_columns		
sort_order	ENUM(asc, desc)		DEFAULT ASC

Tabel 4.8: Tabel *ba_etl_derived_column_lines*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		NOT NULL; FOREIGN KEY
column_key	MANY2ONE ba_etl_columns		
new_or_modify	ENUM(new, modify)		DEFAULT NEW
new_column_key	VARCHAR	64	
expression	TEXT		
data_type	ENUM(char, integer, float, date, datetime, boolean, selection)		
length	INTEGER		

Tabel 4.9: Tabel *ba_etl_lookup_fixed_lines*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		NOT NULL; FOREIGN KEY
lookup	VARCHAR	500	
value	VARCHAR	500	

Tabel 4.10: Tabel *ba_etl_save_data_column*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		
source_column	MANY2ONE ba_etl_column		
destination_column	VARCHAR	64	

Tabel 4.11: Tabel *ba_etl_lookup_model_mappings*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		
source_column	MANY2ONE ba_etl_column		
destination_column	VARCHAR	64	

Tabel 4.12: Tabel *ba_ etl_ merge_ columns_ 1*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		
source_column	VARCHAR	64	
destination_column	VARCHAR	64	
is_included	BOOLEAN		DEFAULT 1
join_field	VARCHAR	64	

Tabel 4.13: Tabel *ba_ etl_ merge_ columns_ 2*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		
source_column	VARCHAR	64	
destination_column	VARCHAR	64	
is_included	BOOLEAN		DEFAULT 1
join_field	VARCHAR	64	

Tabel 4.14: Tabel *ba_ etl_ aggregate_ columns*

Field	Tipe	Panjang	Constraint
id	NUMBER	10	NOT NULL; PRIMARY KEY
header_id	MANY2ONE ba_etl_detail		
sequence	INTEGER		
column_key	MANY2ONE ba_etl_columns		
operation	ENUM(none, count, sum, avg, max, min, group_by)		

4.3 Diagram Kelas

Berikut ini merupakan gambaran diagram kelas dari BI *tool*. Setiap kelas pada diagram ini meng-*extend* kelas osv yang merupakan bawaan dari *framework* ODOO.

Tabel 4.15: Tabel Kelas *ba_ bussiness*

Atribut	
Nama atribut	Tipe Data
name	CHAR(200)

Tabel 4.16: Tabel Kelas *ba_ etl_ header*

Atribut	
Nama atribut	Tipe Data
name	CHAR(100)
business_id	MANY2ONE('ba.business')
is_active	BOOLEAN
etl_columns	ONE2MANY('ba.etl.columns')
etl_detail	ONE2MANY('ba.etl.detail')
Method	
action_run_etl()	
action_open_etl_run()	
action_fillin_columns()	

Tabel 4.17: Tabel Kelas *ba_ etl_ columns*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.header')
name	CHAR

Tabel 4.18: Tabel Kelas *ba_ etl_ detail*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.header')
process_type	selection
name	CHAR(255)
sequence	INT
column_mapping	ONE2MANY('ba.el.column.mapping)
read_data_source_type	SELECTION
read_data_csv_trial_file	BINARY
read_data_csv_delimiter	CHAR
read_data_csv_auto_read	BOOLEAN
read_data_csv_auto_read_path	CHAR
read_data_csv_is_header_present	BOOLEAN
read_data_model_id	MANY2ONE('ir.model')
format_data_type_lines	ONE2MANY('ba.etl.format.data.lines')
script_text	TEXT
save_data_model_id	MANY2ONE('ir.model')
save_data_columns	ONE2MANY('ba.etl.save.data.columns')
sort_data_columns	ONE2MANY('ba.etl.sort.data.column')
derived_column_lines	ONE2MANY('ba.etl.derived.column')
lookup_source_key_column	MANY2ONE('ba.etl.columns')
lookup_target_key_column	CHAR
lookup_from	SELECTION
lookup_model_id	MANY2ONE('ir.model')
lookup_model_value_column	MANY2ONE(ir.model.fields)
lookup_fixed_values	ONE2MANY('ba.etl.lookup.fixed.lines')
lookup_model_mappings	ONE2MANY('ba.etl.lookup.model.mappings')
merge_source_type_1	SELECTION
merge_source_type_2	SELECTION
merge_csv_trial_file_1	BINARY
merge_csv_trial_file_2	BINARY
merge_csv_delimiter_1	CHAR
merge_csv_delimiter_2	CHAR
merge_type	SELECTION
merge_column_1	ONE2MANY('ba.etl.merge.columns1')
merge_column_2	ONE2MANY('ba.etl.merge.columns2')
Method	
get_columns_from_csv(csv_content, delimiter)	
onchange_header_id(header_id)	
onchange_csv_trial_file(read_data_csv_trial_file, delimiter)	
onchange_csv_trial_file1(merge_csv_trial_file_1, merge_csv_delimiter_1)	
onchange_csv_trial_file2(merge_csv_trial_file_2, merge_csv_delimiter_2)	
onchange_read_data_model(read_data_model_id)	
get_etl_columns(header_id)	
onchange_merge_source_type_1(header_id, merge_source_type_1)	
onchange_save_data_model(save_data_model_id)	
onchange_lookup_model_id(lookup_model_id)	

Tabel 4.19: Tabel Kelas *ba_etl_columns_mapping*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
source_column	CHAR(255)
destination_column	CHAR
Method	
constraint_datakey_name()	

Tabel 4.20: Tabel Kelas *ba_etl_format_data_lines*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
column_key	MANY2ONE('ba.etl.column')
column_type	SELECTION
string_length	INT
column_format	CHAR(50)
column_default	CHAR(500)

Tabel 4.21: Tabel Kelas *ba_etl_sort_data_columns*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
sequence	INT
column_key	MANY2ONE('ba.etl.column')
sort_order	SELECTION

Tabel 4.22: Tabel Kelas *ba_etl_derived_column_lines*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
column_key	MANY2ONE('ba.etl.column')
new_or_modify	SELECTION
new_column_key	CHAR(64)
expression	TEXT
data_type	SELECTION
length	INT

Tabel 4.23: Tabel Kelas *ba_etl_save_data_columns*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
source_column	MANY2ONE('ba.etl.column')
destination	CHAR(64)

Tabel 4.24: Tabel Kelas *ba_etl_lookup_fixed_lines*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
lookup	CHAR
value	CHAR

Tabel 4.25: Tabel Kelas *ba_etl_lookup_model_mappings*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
source_column	MANY2ONE('ba.etl.column')
model_column	MANY2ONE('ir.model.fields')

Tabel 4.26: Tabel Kelas *ba_etl_merge_columns1*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
source_column	CHAR(64)
destination_column	CHAR(64)
is_included	BOOLEAN
join_field	CHAR

Tabel 4.27: Tabel Kelas *ba_etl_merge_columns2*

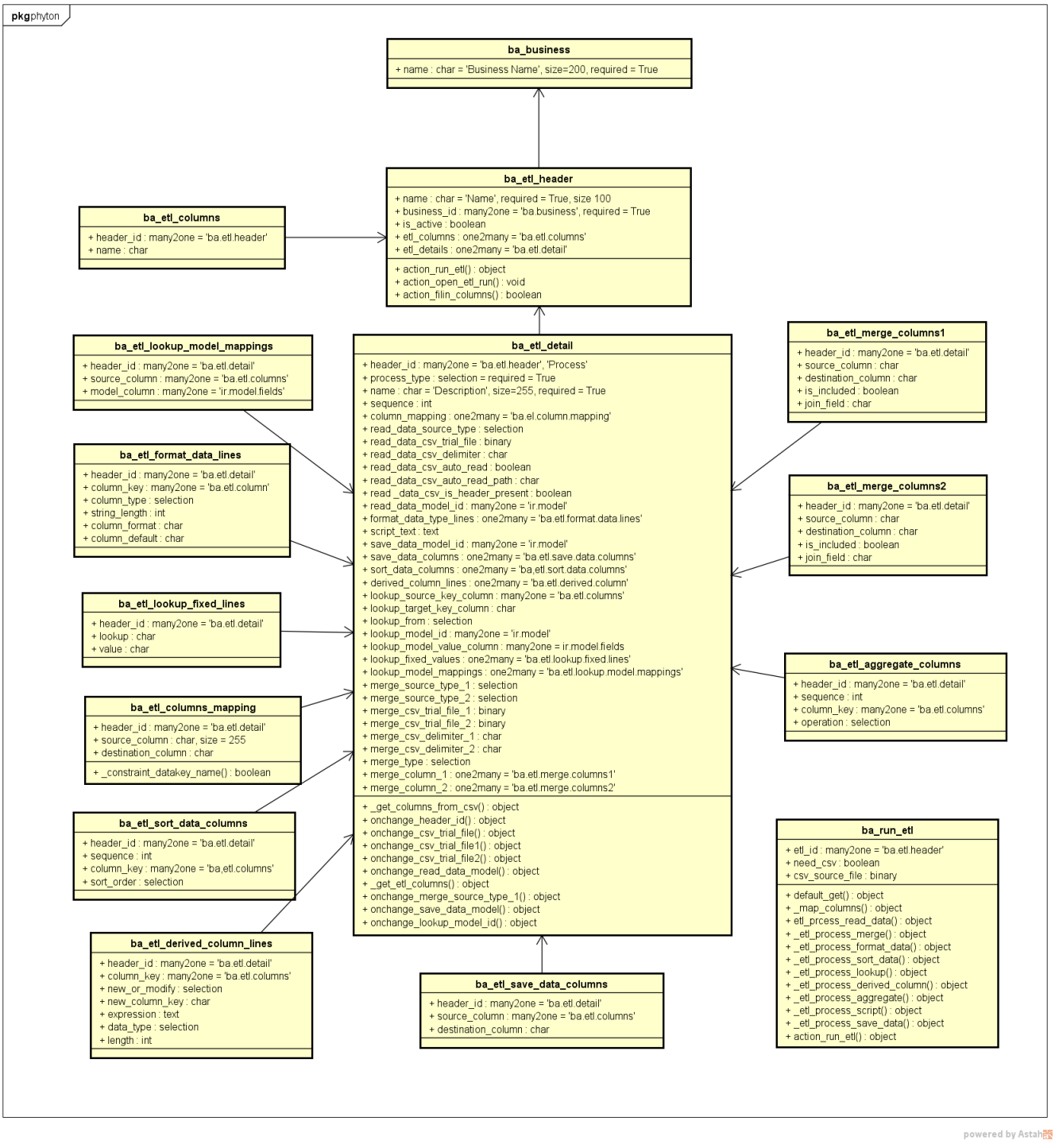
Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
source_column	CHAR(64)
destination_column	CHAR(64)
is_included	BOOLEAN
join_field	CHAR

Tabel 4.28: Tabel Kelas *ba_etl_aggregate_columns*

Atribut	
Nama atribut	Tipe Data
header_id	MANY2ONE('ba.etl.detail')
sequence	INT
column_key	MANY2ONE('ba.etl.columns')
operation	SELECTION

Tabel 4.29: Tabel Kelas *ba_run_etl*

Atribut	
Nama atribut	Tipe Data
etl_id	MANY2ONE('ba.etl.header')
need_csv	BOOLEAN
csv_source_file	BINARY
Method	
default_get(fields)	
map_columns(source_data, columns_mapping)	
etl_process_read_data(step_info, raw_data)	
etl_process_merge(step_info, raw_data)	
etl_process_format_data(step_info, raw_data)	
etl_process_sort_data(step_info, raw_data)	
etl_process_lookup(step_info, raw_data)	
etl_process_derived_column(step_info, raw_data)	
etl_process_aggregate(step_info, raw_data)	
etl_process_script(step_info, raw_data)	
etl_process_save_data(step_info, raw_data)	
action_run_etl(etl_id)	



Gambar 4.2: Struktur kelas diagram BI *tool*

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Pada bagian ini akan dijelaskan mengenai lingkungan implementasi perangkat keras maupun perangkat lunak. Serta implementasi basisdata BI *tool* beserta tampilan antar muka. Terakhir akan dibahas mengenai pengujian pada perangkat lunak ini.

5.1 Lingkungan Implementasi

5.1.1 Lingkungan Perangkat Keras

Dalam mengembangakan perangkat ini, digunakan spesifikasi perangkat keras sebagai berikut:

- Processor : Intel Core i7 2.4 Ghz
- *Memory* : 8 GB
- *Hardisk* : 640 GB
- VGA : Nvidia GeForce 540M
- *keyboard* dan *mouse standard*

5.1.2 Lingkungan Perangkat Lunak

Untuk pengembangan perangkat lunak BI *tool*, digunakan spesifikasi sebagai berikut:

- *Tools:* ODOO 8.0-20141128
- Bahasa Pemograman Phyton 2.7
- *Database management system* PosgreSQL 9.3
- *Operating system*: windows 8

5.2 Implementasi Tabel Basisdata

5.2.1 Tabel Basisdata ETL dan Bisnis

Berikut implementasi tabel basisdata yang terlibat langsung dalam proses ETL beserta pendukungnnya.

- Tabel ba_business
  Tabel ini digunakan untuk mencatat semua bisnis yang menggunakan perangkat lunak ini. Selain itu , agar dapat *multicompany/multiclient*.

```
-- Table: ba_business

-- DROP TABLE ba_business;

CREATE TABLE ba_business
(
    id serial NOT NULL,
    create_uid integer, -- Created by
    create_date timestamp without time zone, -- Created on
    name character varying(200) NOT NULL, -- Business Name
    write_uid integer, -- Last Updated by
    write_date timestamp without time zone, -- Last Updated on
    CONSTRAINT ba_business_pkey PRIMARY KEY (id),
    CONSTRAINT ba_business_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_business_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_business
    OWNER TO openpg;
COMMENT ON TABLE ba_business
    IS 'Client Businesses';
COMMENT ON COLUMN ba_business.create_uid IS 'Created by';
COMMENT ON COLUMN ba_business.create_date IS 'Created on';
COMMENT ON COLUMN ba_business.name IS 'Business Name';
COMMENT ON COLUMN ba_business.write_uid IS 'Last Updated by';
COMMENT ON COLUMN ba_business.write_date IS 'Last Updated on';
```

Gambar 5.1: Implementasi tabel ba_business

- Tabel ba_etl_header
  Tabel ini mencatat proses ETL yang ada di bawah perusahaan/bisnis yang ada.

```
CREATE TABLE ba_etl_header
(
    id serial NOT NULL,
    create_date timestamp without time zone, -- Created on
    name character varying(100) NOT NULL, -- Name
    create_uid integer, -- Created by
    is_active boolean, -- Active?
    business_id integer NOT NULL, -- Business
    write_uid integer, -- Last Updated by
    write_date timestamp without time zone, -- Last Updated on
    CONSTRAINT ba_etl_header_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_header_business_id_fkey FOREIGN KEY (business_id)
        REFERENCES ba_business (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_header_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_header_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_etl_header
    OWNER TO openpg;
COMMENT ON TABLE ba_etl_header
    IS 'Store ETL process settings';
COMMENT ON COLUMN ba_etl_header.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_header.name IS 'Name';
COMMENT ON COLUMN ba_etl_header.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_header.is_active IS 'Active?';
COMMENT ON COLUMN ba_etl_header.business_id IS 'Business';
COMMENT ON COLUMN ba_etl_header.write_uid IS 'Last Updated by';
COMMENT ON COLUMN ba_etl_header.write_date IS 'Last Updated on';
```

Gambar 5.2: Implementasi tabel ba_etl_header

- Tabel ba_etl_detail
  Tabel ini mencata detail dari setiap proses ETL dibawah perusahaan/bisnis.

```
CREATE TABLE ba_etl_detail
(
    id serial NOT NULL,
    read_data_csv_auto_read boolean, -- Auto-read from a folder?
    sequence integer, -- Step
    write_uid integer, -- Last Updated by
    save_data_model_id integer, -- Target Data Model
    write_date timestamp without time zone, -- Last Updated on
    header_id integer, -- Process
    create_date timestamp without time zone, -- Created on
    create_uid integer, -- Created by
    name character varying(255) NOT NULL, -- Description
    read_data_csv_trial_file bytea, -- Upload Sample File
    read_data_csv_delimiter character varying, -- Delimiter
    process_type character varying NOT NULL, -- Process Type
    read_data_source_type character varying, -- Source Type
    read_data_csv_auto_read_path character varying, -- Auto-read Folder Path
    read_data_csv_is_header_present boolean, -- CSV Has Header Row?
    script_text text, -- Enter Script
    lookup_from character varying, -- Lookup From
    lookup_source_key_column_moved0 character varying, -- Source Data Key
    lookup_model_key_column_moved0 character varying, -- Table Lookup Column
    lookup_target_key_column character varying, -- Target Data Key
    lookup_model_value_column_moved0 character varying, -- Table Value Column
    lookup_model_id integer, -- Table
    merge_csv_delimiter_2 character varying, -- Source 2 Delimiter
    merge_csv_delimiter_1 character varying, -- Source 1 Delimiter
    merge_csv_trial_file_1 bytea, -- Source 1 Sample
    lookup_model_value_column integer, -- Table Value Column
    merge_csv_trial_file_2 bytea, -- Source 2 Sample
    lookup_model_key_column integer, -- Table Lookup Column
    merge_csv_is_header_present_1 boolean, -- Source 1 Has Header Row?
    merge_csv_is_header_present_2 boolean, -- Source 2 Has Header Row?
    merge_source_type_1 character varying, -- Source 1 Type
    merge_type character varying, -- Merge Type
    merge_source_type_2 character varying, -- Source 2 Type
```

Gambar 5.3: Implementasi tabel ba_etl_detail

```
    merge_source_type_2 character varying, -- Source 2 Type
    lookup_source_key_column integer, -- Source Data Key
    read_data_model_id integer, -- Existing Model
    CONSTRAINT ba_etl_detail_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_detail_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_header_id_fkey FOREIGN KEY (header_id)
        REFERENCES ba_etl_header (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_lookup_model_id_fkey FOREIGN KEY (lookup_model_id)
        REFERENCES ir_model (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_lookup_model_key_column_fkey FOREIGN KEY (lookup_model_key_column)
        REFERENCES ir_model_fields (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_lookup_model_value_column_fkey FOREIGN KEY (lookup_model_value_column)
        REFERENCES ir_model_fields (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_lookup_source_key_column_fkey FOREIGN KEY (lookup_source_key_column)
        REFERENCES ba_etl_columns (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_read_data_model_id_fkey FOREIGN KEY (read_data_model_id)
        REFERENCES ir_model (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_save_data_model_id_fkey FOREIGN KEY (save_data_model_id)
        REFERENCES ir_model (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_detail_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
```

Gambar 5.4: Implementasi tabel ba_etl_detail2

```
,,
ALTER TABLE ba_etl_detail
  OWNER TO openpg;
COMMENT ON TABLE ba_etl_detail
  IS 'Store ETL method process details, based on process type';
COMMENT ON COLUMN ba_etl_detail.read_data_csv_auto_read IS 'Auto-read from a folder?';
COMMENT ON COLUMN ba_etl_detail.sequence IS 'Step';
COMMENT ON COLUMN ba_etl_detail.write_uid IS 'Last Updated by';
COMMENT ON COLUMN ba_etl_detail.save_data_model_id IS 'Target Data Model';
COMMENT ON COLUMN ba_etl_detail.write_date IS 'Last Updated on';
COMMENT ON COLUMN ba_etl_detail.header_id IS 'Process';
COMMENT ON COLUMN ba_etl_detail.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_detail.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_detail.name IS 'Description';
COMMENT ON COLUMN ba_etl_detail.read_data_csv_trial_file IS 'Upload Sample File';
COMMENT ON COLUMN ba_etl_detail.read_data_csv_delimiter IS 'Delimiter';
COMMENT ON COLUMN ba_etl_detail.process_type IS 'Process Type';
COMMENT ON COLUMN ba_etl_detail.read_data_source_type IS 'Source Type';
COMMENT ON COLUMN ba_etl_detail.read_data_csv_auto_read_path IS 'Auto-read Folder Path';
COMMENT ON COLUMN ba_etl_detail.read_data_csv_is_header_present IS 'CSV Has Header Row?';
COMMENT ON COLUMN ba_etl_detail.script_text IS 'Enter Script';
COMMENT ON COLUMN ba_etl_detail.lookup_from IS 'Lookup From';
COMMENT ON COLUMN ba_etl_detail.lookup_source_key_column_moved0 IS 'Source Data Key';
COMMENT ON COLUMN ba_etl_detail.lookup_model_key_column_moved0 IS 'Table Lookup Column';
COMMENT ON COLUMN ba_etl_detail.lookup_target_key_column IS 'Target Data Key';
COMMENT ON COLUMN ba_etl_detail.lookup_model_value_column_moved0 IS 'Table Value Column';
COMMENT ON COLUMN ba_etl_detail.lookup_model_id IS 'Table';
COMMENT ON COLUMN ba_etl_detail.merge_csv_delimiter_2 IS 'Source 2 Delimiter';
COMMENT ON COLUMN ba_etl_detail.merge_csv_delimiter_1 IS 'Source 1 Delimiter';
COMMENT ON COLUMN ba_etl_detail.merge_csv_trial_file_1 IS 'Source 1 Sample';
COMMENT ON COLUMN ba_etl_detail.lookup_model_value_column IS 'Table Value Column';
COMMENT ON COLUMN ba_etl_detail.merge_csv_trial_file_2 IS 'Source 2 Sample';
COMMENT ON COLUMN ba_etl_detail.lookup_model_key_column IS 'Table Lookup Column';
COMMENT ON COLUMN ba_etl_detail.merge_csv_is_header_present_1 IS 'Source 1 Has Header Row?';
COMMENT ON COLUMN ba_etl_detail.merge_csv_is_header_present_2 IS 'Source 2 Has Header Row?';
COMMENT ON COLUMN ba_etl_detail.merge_source_type_1 IS 'Source 1 Type';
```

Gambar 5.5: Implementasi tabel ba_etl_detail3

- Tabel basisdata ba_etl_columns
  Tabel ini berfungsi menyimpan kolom-kolom yang terlibat dalam proses ETL, baik dari sumber data maupun yang ditambahkan di tengah proses

```
CREATE TABLE ba_etl_columns
(
    id serial NOT NULL,
    create_uid integer, -- Created by
    create_date timestamp without time zone, -- Created on
    name character varying, -- Column Key
    write_uid integer, -- Last Updated by
    write_date timestamp without time zone, -- Last Updated on
    header_id integer, -- Process
    CONSTRAINT ba_etl_columns_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_columns_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_columns_header_id_fkey FOREIGN KEY (header_id)
        REFERENCES ba_etl_header (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_columns_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_etl_columns
    OWNER TO openpg;
COMMENT ON TABLE ba_etl_columns
    IS 'Columns of source data tp be used in ETL process';
COMMENT ON COLUMN ba_etl_columns.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_columns.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_columns.name IS 'Column Key';
COMMENT ON COLUMN ba_etl_columns.write_uid IS 'Last Updated by';
COMMENT ON COLUMN ba_etl_columns.write_date IS 'Last Updated on';
COMMENT ON COLUMN ba_etl_columns.header_id IS 'Process';
```

Gambar 5.6: Implementasi tabel ba_etl_columns

- Tabel basisdata ba_etl_columns_mapping
  Tabel ini menyimpan detail *mapping* kolom dari data *input* ke *output*.

```
CREATE TABLE ba_etl_columns_mapping
(
    id serial NOT NULL,
    source_column character varying(255), -- Source Column
    create_date timestamp without time zone, -- Created on
    create_uid integer, -- Created by
    destination_column character varying, -- Destination Column
    write_uid integer, -- Last Updated by
    write_date timestamp without time zone, -- Last Updated on
    header_id integer, -- Header
    CONSTRAINT ba_etl_columns_mapping_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_columns_mapping_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_columns_mapping_header_id_fkey FOREIGN KEY (header_id)
        REFERENCES ba_etl_detail (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_columns_mapping_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_etl_columns_mapping
    OWNER TO openpg;
COMMENT ON TABLE ba_etl_columns_mapping
    IS 'Columns mapping between process steps';
COMMENT ON COLUMN ba_etl_columns_mapping.source_column IS 'Source Column';
COMMENT ON COLUMN ba_etl_columns_mapping.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_columns_mapping.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_columns_mapping.destination_column IS 'Destination Column';
COMMENT ON COLUMN ba_etl_columns_mapping.write_uid IS 'Last Updated by';
COMMENT ON COLUMN ba_etl_columns_mapping.write_date IS 'Last Updated on';
COMMENT ON COLUMN ba_etl_columns_mapping.header_id IS 'Header';
```

Gambar 5.7: Implementasi tabel ba_etl_columns_mapping

- Tabel basisdata ba_etl_format_data_lines
  Tabel ini menyimpan detil informasi dari setiap kolom khusus tipe ETL Format Data.

```
CREATE TABLE ba_etl_columns_mapping
(
  id serial NOT NULL,
  source_column character varying(255), -- Source Column
  create_date timestamp without time zone, -- Created on
  create_uid integer, -- Created by
  destination_column character varying, -- Destination Column
  write_uid integer, -- Last Updated by
  write_date timestamp without time zone, -- Last Updated on
  header_id integer, -- Header
  CONSTRAINT ba_etl_columns_mapping_pkey PRIMARY KEY (id),
  CONSTRAINT ba_etl_columns_mapping_create_uid_fkey FOREIGN KEY (create_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_columns_mapping_header_id_fkey FOREIGN KEY (header_id)
      REFERENCES ba_etl_detail (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_columns_mapping_write_uid_fkey FOREIGN KEY (write_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ba_etl_columns_mapping
  OWNER TO openpg;
COMMENT ON TABLE ba_etl_columns_mapping
  IS 'Columns mapping between process steps';
COMMENT ON COLUMN ba_etl_columns_mapping.source_column IS 'Source Column';
COMMENT ON COLUMN ba_etl_columns_mapping.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_columns_mapping.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_columns_mapping.destination_column IS 'Destination Column';
COMMENT ON COLUMN ba_etl_columns_mapping.write_uid IS 'Last Updated by';
COMMENT ON COLUMN ba_etl_columns_mapping.write_date IS 'Last Updated on';
COMMENT ON COLUMN ba_etl_columns_mapping.header_id IS 'Header';
```

Gambar 5.8: Implementasi tabel ba_etl_columns_mapping

- Tabel basisdata ba_etl_sort_data_columns
  Tabel ini menyimpan detail kolom dan arah sorting khusus tipe ETL *Sort Data*

```
CREATE TABLE ba_etl_sort_data_columns
(
  id serial NOT NULL,
  create_uid integer, -- Created by
  create_date timestamp without time zone, -- Created on
  sequence integer, -- Sort Order
  write_uid integer, -- Last Updated by
  column_key_moved0 character varying(255), -- Column Key
  sort_order character varying, -- Direction
  write_date timestamp without time zone, -- Last Updated on
  header_id integer, -- Header
  column_key integer NOT NULL, -- Column Key
  CONSTRAINT ba_etl_sort_data_columns_pkey PRIMARY KEY (id),
  CONSTRAINT ba_etl_sort_data_columns_column_key_fkey FOREIGN KEY (column_key)
      REFERENCES ba_etl_columns (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_sort_data_columns_create_uid_fkey FOREIGN KEY (create_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_sort_data_columns_header_id_fkey FOREIGN KEY (header_id)
      REFERENCES ba_etl_detail (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_sort_data_columns_write_uid_fkey FOREIGN KEY (write_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ba_etl_sort_data_columns
  OWNER TO openpg;
COMMENT ON TABLE ba_etl_sort_data_columns
  IS 'Details for "sort_data" process type';
COMMENT ON COLUMN ba_etl_sort_data_columns.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_sort_data_columns.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_sort_data_columns.sequence IS 'Sort Order';
```

Gambar 5.9: Implementasi tabel ba_etl_sort_data_columns

- Tabel basisdata ba_etl_derived_column_lines
  Tabel ini menyimpan detail formula untuk menghasilkan *derived column* khusus tipe ETL *Derived Column*.

```
CREATE TABLE ba_etl_derived_column_lines
(
  id serial NOT NULL,
  function character varying, -- Formula
  create_date timestamp without time zone, -- Created on
  data_type character varying, -- Data Type
  write_uid integer, -- Last Updated by
  write_date timestamp without time zone, -- Last Updated on
  header_id integer, -- Header
  create_uid integer, -- Created by
  new_or_modify character varying, -- New Column or Modify
  length integer, -- Length
  column_key_moved0 character varying(255), -- Column Key
  parameter character varying(1000), -- Parameter
  column_key integer, -- Replace
  expression text, -- Expression
  new_column_key character varying(64), -- New Column
  CONSTRAINT ba_etl_derived_column_lines_pkey PRIMARY KEY (id),
  CONSTRAINT ba_etl_derived_column_lines_column_key_fkey FOREIGN KEY (column_key)
      REFERENCES ba_etl_columns (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_derived_column_lines_create_uid_fkey FOREIGN KEY (create_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_derived_column_lines_header_id_fkey FOREIGN KEY (header_id)
      REFERENCES ba_etl_detail (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_derived_column_lines_write_uid_fkey FOREIGN KEY (write_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ba_etl_derived_column_lines
  OWNER TO openpg;
```

Gambar 5.10: Implementasi tabel ba_etl_derived_column_lines

- Tabel basisdata ba_etl_lookup_fixed_lines
  Tabel ini menyimpan pilihan *lookup* yang diambil dari himpan pilihan *fixed*. Khusus tipe ETL *lookup*.

```
CREATE TABLE ba_etl_lookup_fixed_lines
(
  id serial NOT NULL,
  create_uid integer, -- Created by
  create_date timestamp without time zone, -- Created on
  value character varying, -- Result Value
  write_uid integer, -- Last Updated by
  lookup character varying, -- Lookup Value
  write_date timestamp without time zone, -- Last Updated on
  header_id integer, -- Header
  CONSTRAINT ba_etl_lookup_fixed_lines_pkey PRIMARY KEY (id),
  CONSTRAINT ba_etl_lookup_fixed_lines_create_uid_fkey FOREIGN KEY (create_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_lookup_fixed_lines_header_id_fkey FOREIGN KEY (header_id)
      REFERENCES ba_etl_detail (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_lookup_fixed_lines_write_uid_fkey FOREIGN KEY (write_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ba_etl_lookup_fixed_lines
  OWNER TO openpg;
COMMENT ON TABLE ba_etl_lookup_fixed_lines
  IS 'Fixed lookup value set for lookup process';
COMMENT ON COLUMN ba_etl_lookup_fixed_lines.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_lookup_fixed_lines.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_lookup_fixed_lines.value IS 'Result Value';
COMMENT ON COLUMN ba_etl_lookup_fixed_lines.write_uid IS 'Last Updated by';
COMMENT ON COLUMN ba_etl_lookup_fixed_lines.lookup IS 'Lookup Value';
COMMENT ON COLUMN ba_etl_lookup_fixed_lines.write_date IS 'Last Updated on';
COMMENT ON COLUMN ba_etl_lookup_fixed_lines.header_id IS 'Header';
```

Gambar 5.11: Implementasi tabel ba_etl_lookup_fixed_lines

- Tabel basisdata ba_etl_save_data_columns
  Tabel ini menyimpan *mapping* kolom antara hasil proses dengan *field* di tabel. Khusus tipe ETL *Save Data*

```
CREATE TABLE ba_etl_save_data_columns
(
    id serial NOT NULL,
    source_column integer, -- Source Column
    create_date timestamp without time zone, -- Created on
    create_uid integer, -- Created by
    destination_column character varying(64), -- Destination Column
    write_uid integer, -- Last Updated by
    write_date timestamp without time zone, -- Last Updated on
    header_id integer, -- Header
    CONSTRAINT ba_etl_save_data_columns_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_save_data_columns_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_save_data_columns_header_id_fkey FOREIGN KEY (header_id)
        REFERENCES ba_etl_detail (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_save_data_columns_source_column_fkey FOREIGN KEY (source_column)
        REFERENCES ba_etl_columns (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_save_data_columns_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_etl_save_data_columns
    OWNER TO openpg;
COMMENT ON TABLE ba_etl_save_data_columns
    IS 'Save data columns mapping';
COMMENT ON COLUMN ba_etl_save_data_columns.source_column IS 'Source Column';
COMMENT ON COLUMN ba_etl_save_data_columns.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_save_data_columns.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_save_data_columns.destination_column IS 'Destination Column';
COMMENT ON COLUMN ba_etl_save_data_columns.write_uid IS 'Last Updated by';
```

Gambar 5.12: Implementasi tabel ba_etl_save_data_columns

- Tabel basisdata ba_etl_lookup_model_mappings
  Tabel ini menyimpan *lookup* dari tabel lain, tabel ini menampung pasangan *field* di data proses dengan yang di tabel lookup. Khusus tipe ETL *lookup*

```
CREATE TABLE ba_etl_lookup_model_mappings
(
    id serial NOT NULL,
    source_column integer, -- Lookup Source Column
    create_date timestamp without time zone, -- Created on
    create_uid integer, -- Created by
    write_uid integer, -- Last Updated by
    model_column integer, -- Lookup Table Column
    write_date timestamp without time zone, -- Last Updated on
    header_id integer, -- Header
    CONSTRAINT ba_etl_lookup_model_mappings_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_lookup_model_mappings_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_lookup_model_mappings_header_id_fkey FOREIGN KEY (header_id)
        REFERENCES ba_etl_detail (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_lookup_model_mappings_model_column_fkey FOREIGN KEY (model_column)
        REFERENCES ir_model_fields (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_lookup_model_mappings_source_column_fkey FOREIGN KEY (source_column)
        REFERENCES ba_etl_columns (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_lookup_model_mappings_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_etl_lookup_model_mappings
    OWNER TO openpg;
COMMENT ON TABLE ba_etl_lookup_model_mappings
    IS 'Columns mapping between data source and target model/table';
COMMENT ON COLUMN ba_etl_lookup_model_mappings.source_column IS 'Lookup Source Column';
COMMENT ON COLUMN ba_etl_lookup_model_mappings.create_date IS 'Created on';
```

Gambar 5.13: Implementasi tabel ba_etl_lookup_model_mappings

- Tabel basisdata ba_etl_merge_columns1
  Tabel ini menyimpan informasi *merge* dari sumber pertama. Khusus tipe ETL *Merge*

```
CREATE TABLE ba_etl_merge_columns1
(
    id serial NOT NULL,
    create_uid integer, -- Created by
    create_date timestamp without time zone, -- Created on
    is_included boolean, -- Output?
    join_field character varying, -- Join Field
    column_key character varying(64), -- Column Key
    write_date timestamp without time zone, -- Last Updated on
    header_id integer, -- Header
    write_uid integer, -- Last Updated by
    source_column character varying(64), -- Column Key
    destination_column character varying(64), -- Mapping
    CONSTRAINT ba_etl_merge_columns1_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_merge_columns1_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_merge_columns1_header_id_fkey FOREIGN KEY (header_id)
        REFERENCES ba_etl_detail (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_merge_columns1_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_etl_merge_columns1
    OWNER TO openpg;
COMMENT ON TABLE ba_etl_merge_columns1
    IS 'Merge column settings for first data source';
COMMENT ON COLUMN ba_etl_merge_columns1.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_merge_columns1.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_merge_columns1.is_included IS 'Output?';
COMMENT ON COLUMN ba_etl_merge_columns1.join_field IS 'Join Field';
COMMENT ON COLUMN ba_etl_merge_columns1.column_key IS 'Column Key';
```

Gambar 5.14: Implementasi tabel ba_etl_merge_columns1

- Tabel basisdata ba_etl_merge_columns2
  Tabel ini menyimpan informasi *merge* dari sumber kedua. Khusus tipe ETL *Merge*

```
CREATE TABLE ba_etl_merge_columns2
(
    id serial NOT NULL,
    create_uid integer, -- Created by
    create_date timestamp without time zone, -- Created on
    is_included boolean, -- Output?
    join_field character varying, -- Join Field
    column_key character varying(64), -- Column Key
    write_date timestamp without time zone, -- Last Updated on
    header_id integer, -- Header
    write_uid integer, -- Last Updated by
    source_column character varying(64), -- Column Key
    destination_column character varying(64), -- Mapping
    CONSTRAINT ba_etl_merge_columns2_pkey PRIMARY KEY (id),
    CONSTRAINT ba_etl_merge_columns2_create_uid_fkey FOREIGN KEY (create_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_merge_columns2_header_id_fkey FOREIGN KEY (header_id)
        REFERENCES ba_etl_detail (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL,
    CONSTRAINT ba_etl_merge_columns2_write_uid_fkey FOREIGN KEY (write_uid)
        REFERENCES res_users (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
    OIDS=FALSE
);
ALTER TABLE ba_etl_merge_columns2
    OWNER TO openpg;
COMMENT ON TABLE ba_etl_merge_columns2
    IS 'Merge column settings for second data source';
COMMENT ON COLUMN ba_etl_merge_columns2.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_merge_columns2.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_merge_columns2.is_included IS 'Output?';
COMMENT ON COLUMN ba_etl_merge_columns2.join_field IS 'Join Field';
COMMENT ON COLUMN ba_etl_merge_columns2.column_key IS 'Column Key';
```

Gambar 5.15: Implementasi tabel ba_etl_merge_columns2

- Tabel basisdata ba_etl_aggregate_columns
  Tabel ini menyimpan detail informasi proses aggregate khusus tipe ETL *aggregate*.

```
CREATE TABLE ba_etl_aggregate_columns
(
  id serial NOT NULL,
  create_uid integer, -- Created by
  create_date timestamp without time zone, -- Created on
  sequence integer, -- Sequence
  write_uid integer, -- Last Updated by
  column_key integer, -- Column Key
  write_date timestamp without time zone, -- Last Updated on
  header_id integer, -- Header
  operation character varying, -- Operation
  CONSTRAINT ba_etl_aggregate_columns_pkey PRIMARY KEY (id),
  CONSTRAINT ba_etl_aggregate_columns_column_key_fkey FOREIGN KEY (column_key)
      REFERENCES ba_etl_columns (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_aggregate_columns_create_uid_fkey FOREIGN KEY (create_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_aggregate_columns_header_id_fkey FOREIGN KEY (header_id)
      REFERENCES ba_etl_detail (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL,
  CONSTRAINT ba_etl_aggregate_columns_write_uid_fkey FOREIGN KEY (write_uid)
      REFERENCES res_users (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE SET NULL
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ba_etl_aggregate_columns
  OWNER TO openpg;
COMMENT ON TABLE ba_etl_aggregate_columns
  IS 'Definition details for aggregate process';
COMMENT ON COLUMN ba_etl_aggregate_columns.create_uid IS 'Created by';
COMMENT ON COLUMN ba_etl_aggregate_columns.create_date IS 'Created on';
COMMENT ON COLUMN ba_etl_aggregate_columns.sequence IS 'Sequence';
COMMENT ON COLUMN ba_etl_aggregate_columns.write_uid IS 'Last Updated by';
```

Gambar 5.16: Implementasi tabel ba_etl_aggregate_columns

5.3 Implementasi Kode Program Phyton

Berikut ini sisipan kode program perangkat lunak BI *tool*

1. Kode Program *load* kolom dari *source*

```
# buat load semua kolom dari source
def action_fillin_columns(self, cr, uid, ids, context={}):
    etl_id = ids[0]
    data = self.browse(cr, uid, etl_id, context=context)
# kudu udah ada step minimal 1
    if len(data.etl_details) == 0:
        raise osv.except_osv(_("ETL Error"), _('No load data process defined yet. Please define the process first.'))
# step pertama harus load data atau merge
    step_info = data.etl_details[0]
    if step_info.process_type not in ['read_data','merge']:
        raise osv.except_osv(_("ETL Error"), _('First process must be Read data or Merge.'))
# kalau processnya read_data, baca aja langsung dari columns_mapping
    if step_info.process_type == 'read_data':
        etl_columns = []
        for mapping in step_info.columns_mapping:
            etl_columns.append(mapping.destination_column)
        if len(etl_columns) == 0:
            raise osv.except_osv(_("ETL Error"), _('In the read data process, please define column mapping first.'))
# kalau processnya merge, baca dari source 1 dan source 2, menskip yang tidak included dan yang dijadikan join column
    elif step_info.process_type == 'merge':
        etl_columns = []
        left_join_columns = []
        right_join_columns = []
        for column in step_info.merge_columns_1:
            if column.is_included: etl_columns.append(column.output_key)
            if column.join_field: left_join_columns.append(column.join_field)
        for column in step_info.merge_columns_2:
            if column.is_included: etl_columns.append(column.output_key)
            if column.join_field: right_join_columns.append(column.join_field)
    # hapus kolom2 yang dipakai untuk join. kalau left atau inner join, hapus dari left_join_columnss,
    # kalau right join hapus dari yang right_join_columnss
        if step_info.merge_type in ['left','inner']:
            for column in left_join_columns:
                if column in etl_columns: etl_columns.remove(column)
        else:
            for column in right_join_columns:
```

Gambar 5.17: kode program *load columns from source*

2. Kode Program untuk *mapping column*

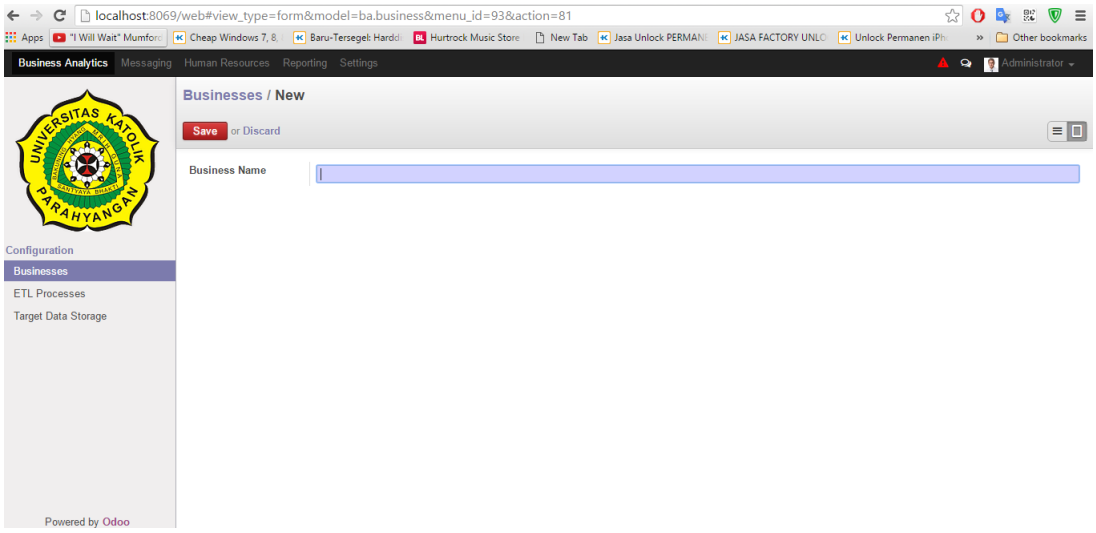
```
def _map_columns(self, source_data, columns_mapping, context={}):
# kalau tidak ada columns_mapping, artinya hasil map idem aslinya
    if len(columns_mapping) == 0: return source_data
    result = []
    for row in source_data:
        result_row = {}
        for mapping in columns_mapping:
            if not mapping.destination_column: continue # kalau destination_
        # kalau source_column dalam bentuk integer, maka diasumsikan row ada
        # bila bukan maka diasumsikan row adalah sebuah dictionary
            is_key_idx = False
            try:
                key = mapping.source_column.name
            except:
                try:
                    key = int(mapping.source_column)
                    is_key_idx = True
                except:
                    key = mapping.source_column
        # tentukan data yang akan dimasukkan ke hasil mapping
            try:
                if is_key_idx:
                    cell_data = row[key]
                else:
                    cell_data = row.get(key, None)
            except:
                cell_data = None
            result_row.update({mapping.destination_column: cell_data})
    # beres ngurusin sebaris. tambahkan ke baris hasil
        result.append(result_row)
# beres semua
    return result

def _etl_process_read_data(self, cr, uid, step_info, raw_data, context={}):
# untuk read dari csv
    if step_info.read_data_source_type == 'csv':
        csv_data = step_info.read_data_csv_trial_file.decode('base64')
```

Gambar 5.18: Kode Program untuk *mapping column*

5.4 Implementasi Antar Muka

5.4.1 Implementasi Antar Muka untuk *Business*

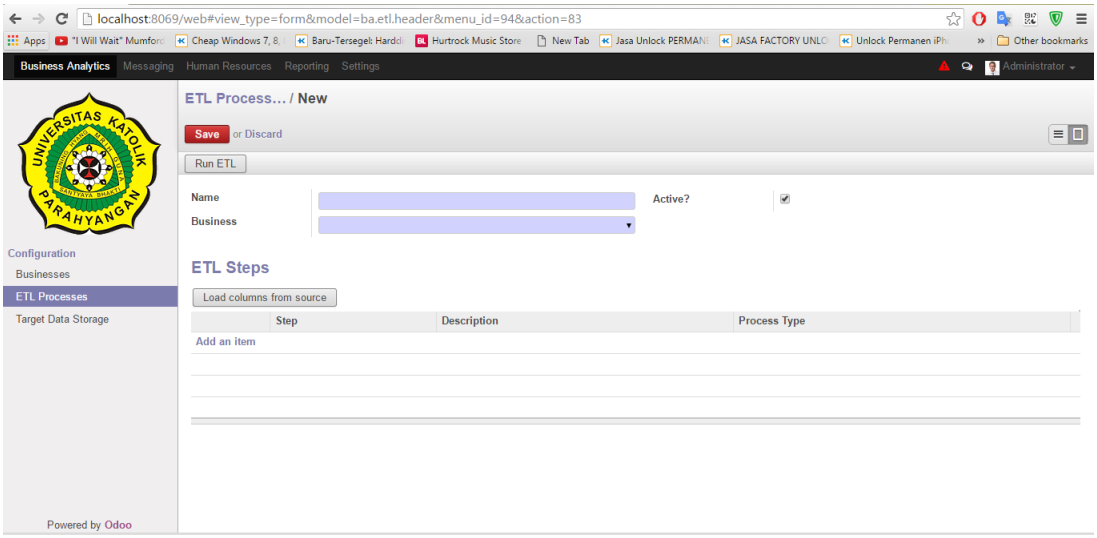
- Buat Bisnis Baru
  Halaman ini digunakan ketika pertama perusahaan akan membuat bisnis baru sebelum masuk pada proses ETL.



Gambar 5.19: Tampilan untuk membuat bisnis baru

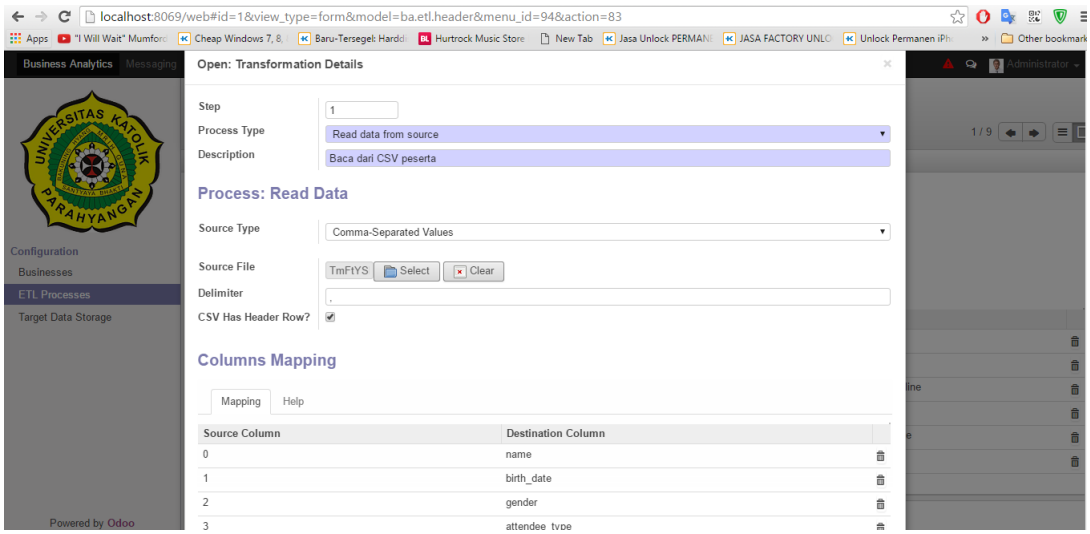
5.4.2 Implementasi Antar Muka untuk Proses ETL

- Membuat Proses ETL Baru
  Halaman ini digunakan ketika membuat proses ETL baru.



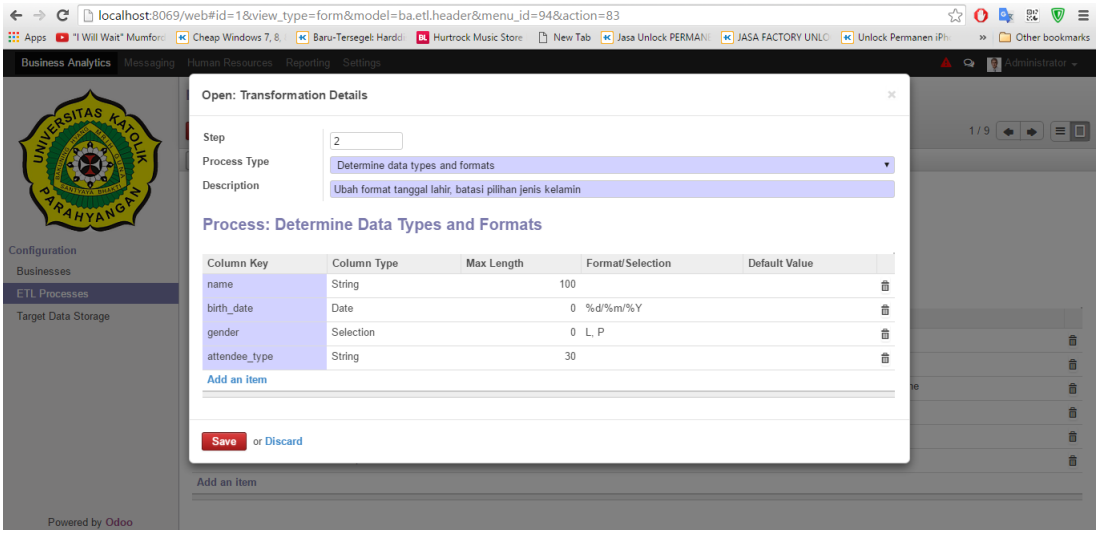
Gambar 5.20: Tampilan untuk membuat ETL baru

- *Upload Source*
  Halaman ini digunakan ketika pengguna meng-*upload data source*.



Gambar 5.21: Tampilan untuk *upload source*

- *Determine Data Types and Format*
  Halaman ini memudahkan pengguna untuk menentukan format dan tipe data pada *source*.

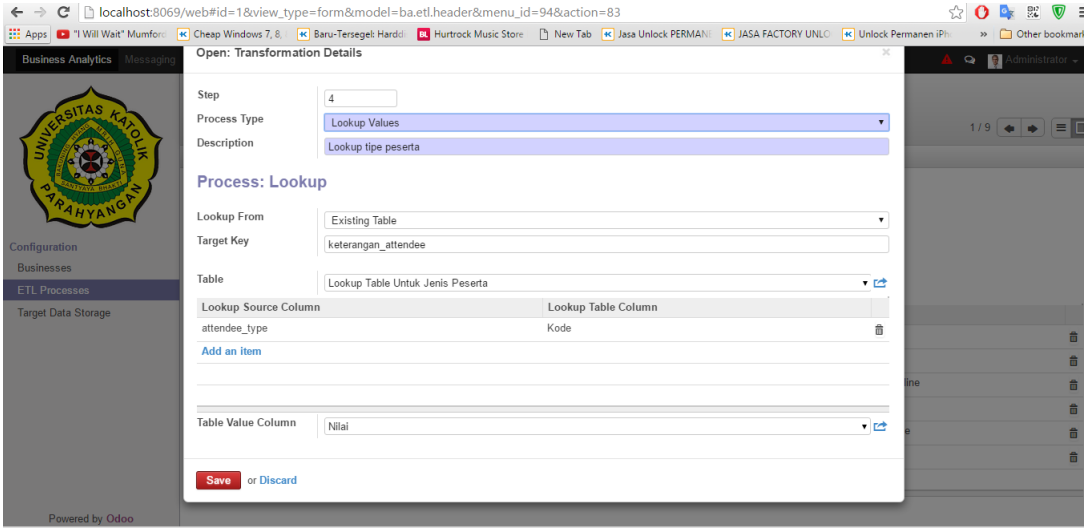


Gambar 5.22: Tampilan untuk ubah data dan format data

- Menerapkan Skrip Phyton
  Halaman ini dapat digunakan pengguna ketika akan menerapkan kode phyton untuk memodifikasi *source*.

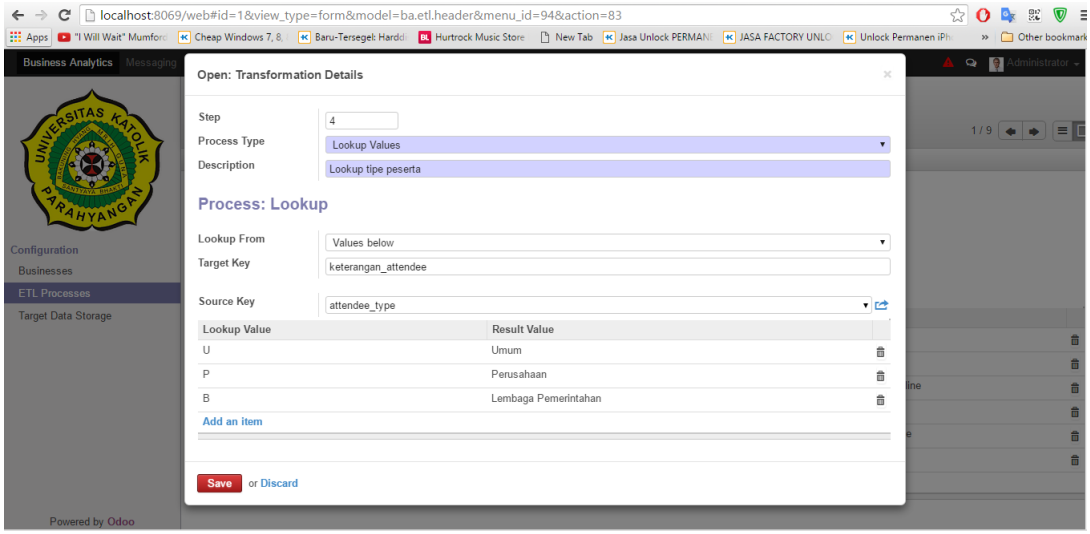
Gambar 5.23: Tampilan untuk menerapkan skrip phyton

- Tampilan *Lookup* dengan *Existing Table* Halaman ini digunakan ketika pengguna ingin melakukan lookup dengan tabel lain yang sudah dimasukan terlebih dahulu dalam *database*



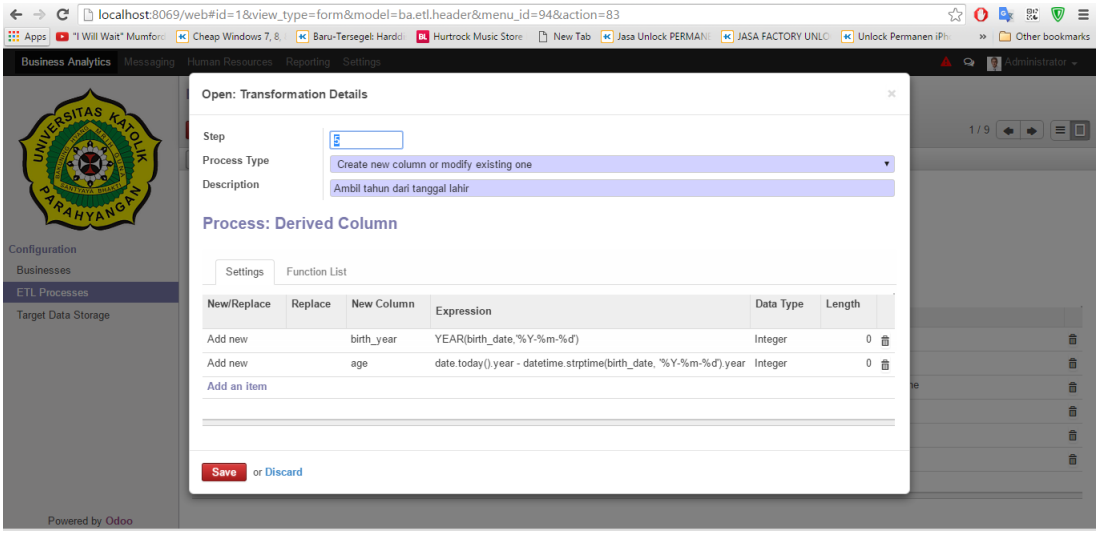
Gambar 5.24: Tampilan untuk *lookup* dari *existing table*

- Tampilan *Lookup* dengan *Values Below* Halaman ini digunakan ketika pengguna ingin melakukan lookup dengan nilai *lookup* dan hasil yang dimasukan langsung oleh pengguna



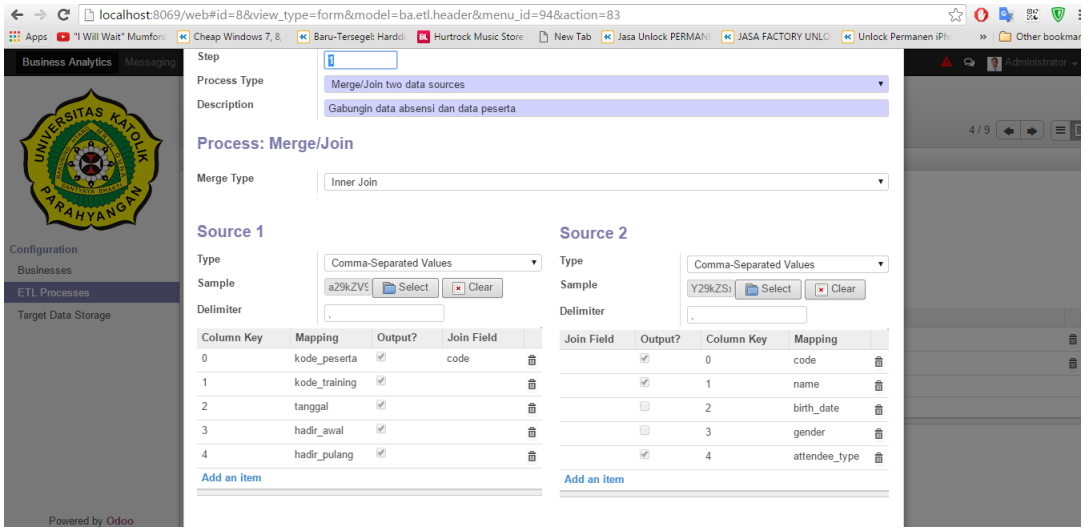
Gambar 5.25: Tampilan untuk *lookup* dari *value* yang dituliskan pengguna

- Menambah atau memodifikasi kolom yang tersedia
  Halaman ini digunakan ketika pengguna ingin menambahkan kolom atau memodifikasi kolom yang tersedia.



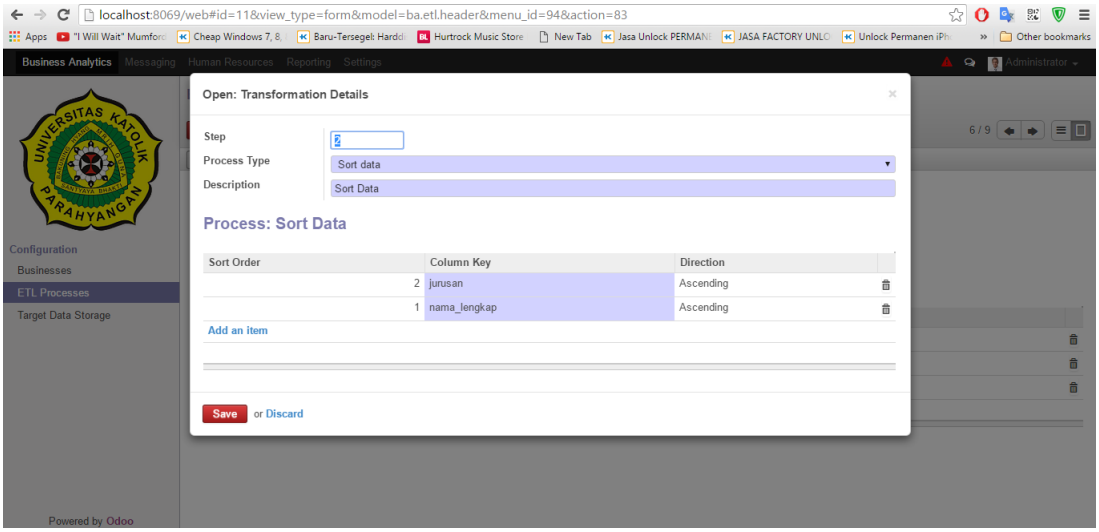
Gambar 5.26: Tampilan untuk menambah atau memodifikasi kolom

- *Merging* dua *data source*
  Halaman ini digunakan ketika pengguna ingin menggabungkan dua *data source* menjadi satu.



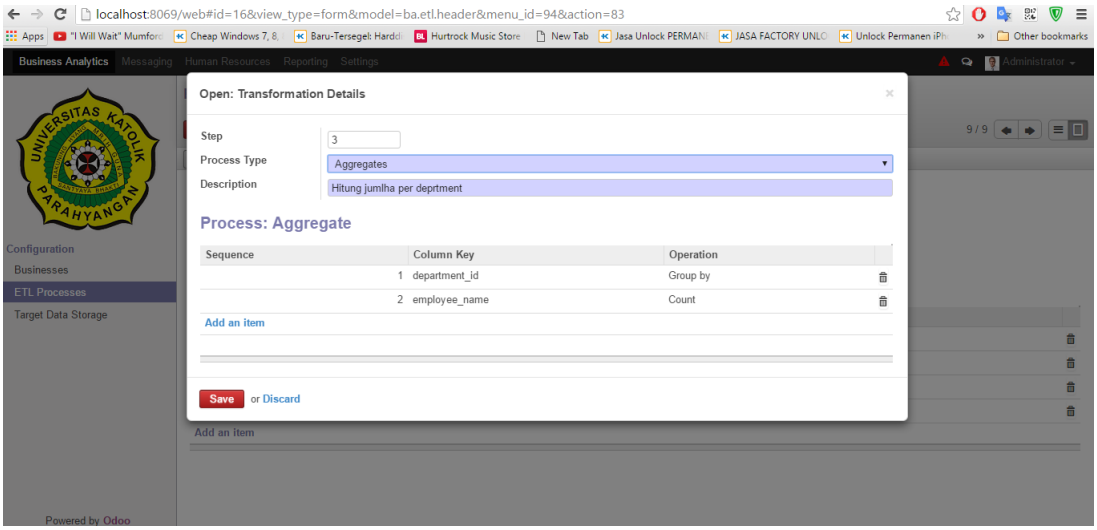
Gambar 5.27: Tampilan untuk *merging* dua *source*

- Melakukan *Sorting data*
  Halaman ini digunakan ketika pengguna ingin melakukan *sorting data* .



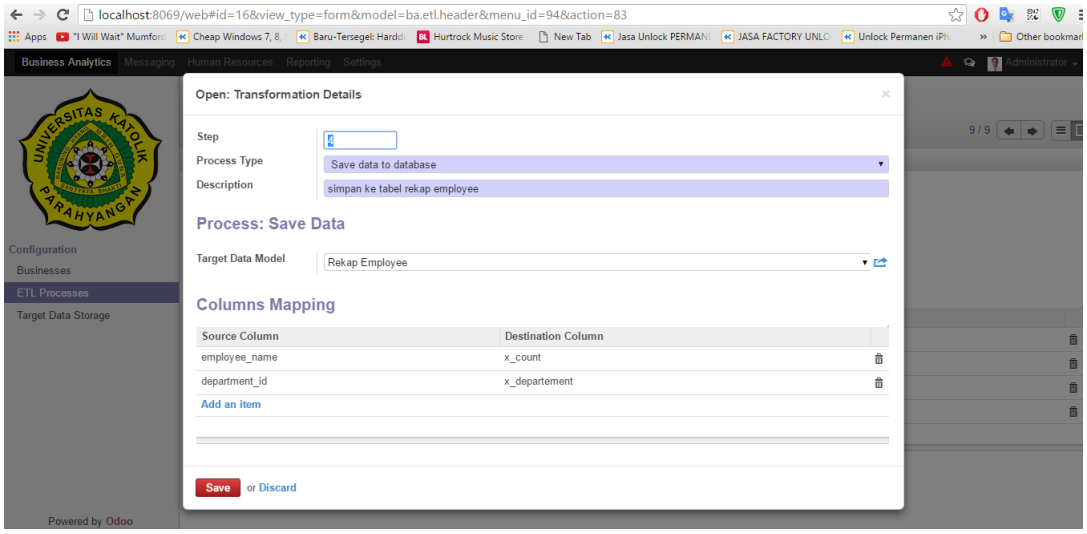
Gambar 5.28: Tampilan untuk melakukan *sort*

- Melakukan Agregat
  Tampilan ini digunakan pengguna ketika akan melakukan agregat terhadap *source*



Gambar 5.29: Tampilan untuk melakukan agregat

- Menyimpan ke *database*
  Halaman ini digunakan ketika pengguna ingin menyimpan hasil ETL kedalam *database*.



Gambar 5.30: Tampilan untuk menyimpan kedalam *database*

5.5 Pengujian

Pada subab ini akan dibahas mengenai pengujian terhadap perangkat lunak ini sehingga dapat mengukur seberapa baik performansinya. berikut ini merupakan *sample data customer* yang akan digunakan.

Tabel 5.1: Contoh Tabel *customer*

CIF	BRANCHCODE	GENDER	AGE	MARITAL STATUS	INCOME LEVEL	TENURE
1000000	ID0010096	2	15131	0	1	1175
1000043	ID0010075	2	14972	1	1	1175
1000044	ID0010049	2	13401	1	1	1175
1000056	ID0010049	1	18992	1	1	1175
1000059	ID0010038	2	20871	1	1	1175
1000064	ID0010049	2	20852	1	1	1175
1000067	ID0010038	1	15728	1	1	1175
1000068	ID0010049	1	16176	1	1	1175
1000069	ID0010026	1	19069	1	1	1175
1000075	ID0010038	1	11279	1	1	1175
1000076	ID0010090	1	18562	1	1	1175
1000084	ID0010090	2	14490	1	1	1175
1000140	ID0010129	1	15216	1	1	1174
1000149	ID0010026	1	9566	0	1	1174
1000166	ID0010108	2	14100	1	6	1174
1000195	ID0010014	1	12617	0	3	1174
1000202	ID0010132	1	23236	1	1	1174
1000244	ID0010096	2	19106	1	2	1174
1000246	ID0010038	2	27964	1	1	1174

Hasil Pengujian fitur adalah sebagai berikut.

1. *Sort, Aggregate, Modify Existing one, Read Source, Save to Database*
   Hasil pengujian proses ETL dapat dilihat pada tabel dibawah ini.

Tabel 5.2: Tabel *ba_ bussiness*

Hal yang diuji	Hasil yang diharapkan	Hasil Pengujian	Status
Upload CSV	Berhasil Melakukan *upload* CSV	Berhasil Melakukan *upload* CSV	OK
Melakukan Sorting	Data tersorting berdasarkan *branchcode* dan *gender*.	Data tersorting berdasarkan *branchcode* dan *gender*	OK
Melakukan Agregat	Hitung rata-rata umur per cabang per jenis kelamin	Data telah terhitung per rata-rata umur per cabang per jenis kelamin	OK
Memodifikasi data	Ubah umur ke dalam satuan tahun	Umur telah berhasil diubah kedalam satuan tahun	OK
Memodifikasi data	Ubah jenis kelamin menjadi P/L	Jenis kelamin menjadi P/L	OK
Menyimpan ke *database*	Simpan hasil ke *database*	Data berhasil tersimpan ke *database*	OK

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Setelah melakukan penelitian maka dapat disimpulkan beberapa hal, yakni:

1. *Framework* ODOO mendukung untuk pembuatan ETL *tool* yang berperan dalam proses BI.
2. ODOO bersifat modular sehingga fleksibilitasnya tinggi.
3. Jika SSIS mebutuhkan 2 *tool* untuk ETL yaitu visual studio dan management studio, ODOO hanya membutuhkan 1 tool saja.
4. Pengguna tanpa pengetahuan SQL dapat menggunakan *tool* ini.

6.2 Saran

Berikut ini saran yang diharapkan dapat membantu pengembangan penelitian ini lebih lanujut, yakni:

1. Menambahkan *input data* XML-RPC
2. Menambahkan OLAP dan Cube
3. *Merge* yang pada saat ini baru bisa di jalankan apabila ditempatkan paling pertama, diharapkan pada penelitian selanjutnya dapat ditempatkan dimana saja.

DAFTAR REFERENSI

[1] H. Lauri, pp. 37–39. 2014.

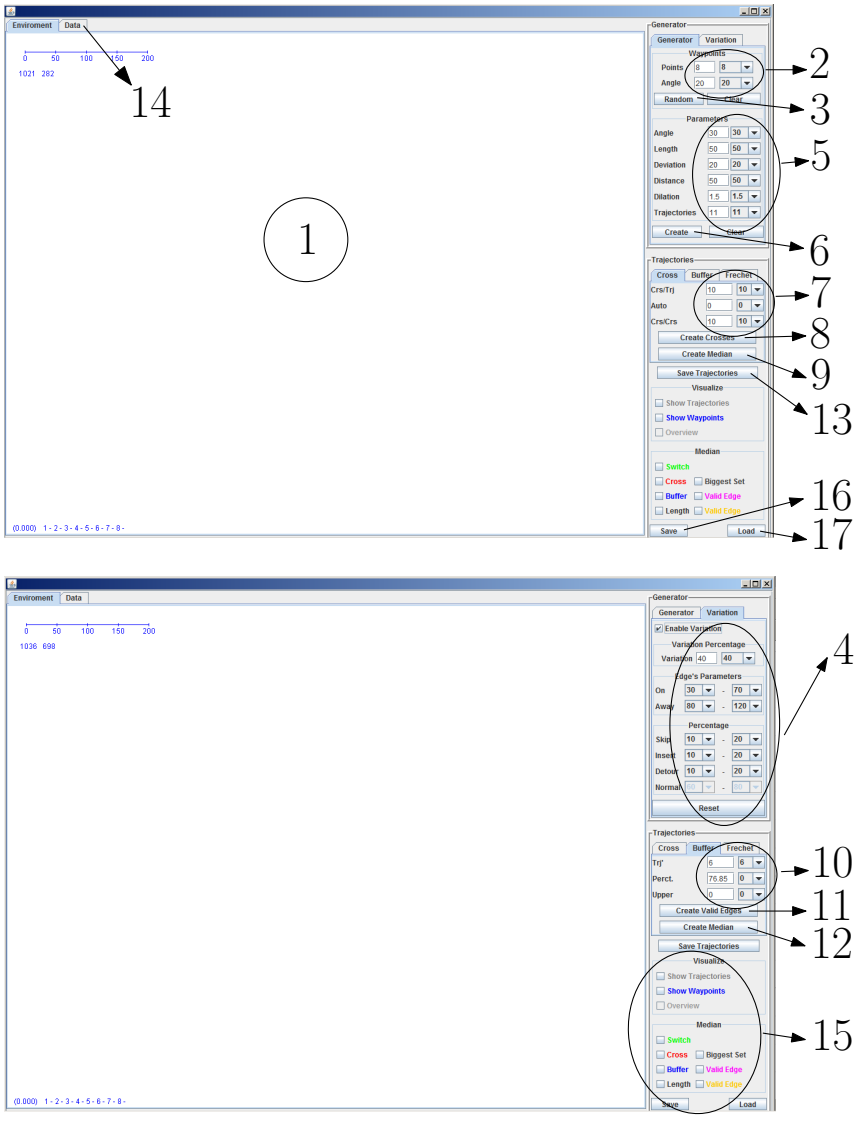
[2] T. Point, "Apache poi." https://www.tutorialspoint.com/apache_poi. [Online; accessed 13/09/2016].

[3] B. Fortune, "ical4j." http://ical4j.github.io/docs/ical4j/api/2.0-beta1. [Online; accessed 14/09/2016].

LAMPIRAN A

THE PROGRAM

The interface of the program is shown in Figure A.1:



Gambar A.1: Interface of the program

Step by step to compute the median trajectory using the program:

1. Create several waypoints. Click anywhere in the "Environment" area(1) or create them automatically by setting the parameters for waypoint(2) or clicking the button "Random"(3).

2. The "Variation" tab could be used to create variations by providing values needed to make them(4).

3. Create a set of trajectories by setting all parameters(5) and clicking the button "Create"(6).
4. Compute the median using the homotopic algorithm:
   - Define all parameters needed for the homotopic algorithm(7).
   - Create crosses by clicking the "Create Crosses" button(8).
   - Compute the median by clicking the "Compute Median" button(9).
5. Compute the median using the switching method and the buffer algorithm:
   - Define all parameters needed for the buffer algorithm(10).
   - Create valid edges by clicking the "Create Valid Edges"button(11).
   - Compute the median by clicking the "Compute Median"button(12).
6. Save the resulting median by clicking the "Save Trajectories" button(13). The result is saved in the computer memory and can be seen in "Data" tab(14)
7. The set of trajectories and its median trajectories will appear in the "Environment" area(1) and the user can change what to display by selecting various choices in "Visualize" and "Median" area(15).
8. To save all data to the disk, click the "Save"(16) button. A file dialog menu will appear.
9. To load data from the disk, click the "Load"(17) button.

LAMPIRAN B

THE SOURCE CODE

Listing B.1: MyFurSet.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

/**
 *
 * @author Lionov
 */

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                       //id of the set
    protected MyEdge FurthestEdge;                          //the furthest edge
    protected HashSet<MyVertex> set;                        //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;        //list of all vertices in the set for each
        trajectory
    protected ArrayList<Integer> closeID;                   //store the ID of all vertices
    protected ArrayList<Double> closeDist;                  //store the distance of all vertices
    protected int totaltrj;                                 //total trajectories in the set

    /**
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

    /**
     * set a vertex into the set
     * @param v : vertex to be added to the set
     */
    public void add(MyVertex v) {
        set.add(v);
    }

    /**
     * check whether vertex v is a member of the set
     * @param v : vertex to be checked
     * @return true if v is a member of the set, false otherwise
     */
    public boolean contains(MyVertex v) {
        return this.set.contains(v);
    }
}
```

LAMPIRAN C

THE SOURCE CODE

Listing C.1: MyFurSet.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

/**
 *
 * @author Lionov
 */

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                       //id of the set
    protected MyEdge FurthestEdge;                          //the furthest edge
    protected HashSet<MyVertex> set;                        //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;        //list of all vertices in the set for each
        trajectory
    protected ArrayList<Integer> closeID;                   //store the ID of all vertices
    protected ArrayList<Double> closeDist;                  //store the distance of all vertices
    protected int totaltrj;                                 //total trajectories in the set

    /**
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

    /**
     * set a vertex into the set
     * @param v : vertex to be added to the set
     */
    public void add(MyVertex v) {
        set.add(v);
    }

    /**
     * check whether vertex v is a member of the set
     * @param v : vertex to be checked
     * @return true if v is a member of the set, false otherwise
     */
    public boolean contains(MyVertex v) {
        return this.set.contains(v);
    }

    /**
     *  create a column for table Gamma, sorted for each row
     */
    public void createColumn() {
        for (MyVertex v : set) {
            for (Integer key : v.vertexnum.keySet()) {
                for (Integer values : v.vertexnum.get(key)) {
                    ordered.get(key).add(values);
                }
            }
        }
        for (ArrayList<Integer> al : ordered) Collections.sort(al);
    }


}
```