

**SKRIPSI**

**STUDI DAN INTEGRASI *WORKFLOW* MENGGUNAKAN  
BPMS DAN SISTEM EMAIL**



**LUCKY SENJAYA DARMAWAN**

**NPM: 2012730009**

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2017**



**UNDERGRADUATE THESIS**

**STUDY AND WORKFLOW INTEGRATION USING BPMS  
AND EMAIL SYSTEM**



**LUCKY SENJAYA DARMAWAN**

**NPM: 2012730009**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND  
SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
2017**



## ABSTRAK

*Workflow* merupakan pemodelan proses bisnis yang dapat digambarkan sebagai *flow map* atau BPMN (*Business Process Model and Notation*). *Workflow* ini dapat diotomasi menggunakan BPMS (*Business Process Management System*), seperti Camunda. Agar eksekusi *workflow* lebih alamiah dengan model komunikasi organisasi saat ini, maka *event* dapat dipropagasi dan diintegrasikan dengan sistem email.

Dalam skripsi ini, akan dibuat suatu integrasi antara *user task* dan sistem email. *User task* adalah suatu tugas yang perlu dilakukan oleh pengguna. Ketika ada suatu *user task*, sistem email akan mengirimkan email ke pengguna yang akan mengerjakan task tersebut. Email tersebut berisi tautan yang mengarah ke tugas yang perlu dikerjakan tersebut.

**Kata-kata kunci:** Alur Kerja, Proses Bisnis, BPMN, BPMS, Camunda, Email



## **ABSTRACT**

Workflow is business process model that can be described as a flow map or BPMN (Business Process Model and Notation). Workflow can be automated using BPMS (Business Process Management System), such as Camunda. Workflow execution will be more natural with current organizational communication models, event can be propagated and integrated with email system.

This thesis will develop integration between user task and email system. User task is task that need to be done by the user. When there is a user task, email system will send email to user. The email contains link to the task that needs to be done.

**Keywords:** Workflow, Business Process, BPMN, BPMS, Camunda, Email





# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>ix</b>
<b>DAFTAR GAMBAR</b>	<b>xi</b>
<b>DAFTAR TABEL</b>	<b>xii</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	1
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi . . . . .	2
1.6 Sistematika Pembahasan . . . . .	3
<b>2 DASAR TEORI</b>	<b>5</b>
2.1 <i>Business Process</i> (BP) . . . . .	5
2.1.1 <i>Komponen Business Process</i> . . . . .	5
2.2 <i>Business Process Management</i> (BPM) . . . . .	6
2.2.1 <i>Siklus Business Process Management</i> . . . . .	6
2.3 <i>Business Process Model and Notation</i> . . . . .	7
2.3.1 <i>Event</i> . . . . .	7
2.3.2 <i>Activity</i> . . . . .	8
2.3.3 <i>Gateway</i> . . . . .	8
2.3.4 <i>Data</i> . . . . .	9
2.3.5 <i>Artifact</i> . . . . .	9
2.3.6 <i>Pools dan Lanes</i> . . . . .	9
2.4 <i>Business Process Management System (BPMS)</i> . . . . .	9
2.5 BPMS Camunda . . . . .	10
2.5.1 <i>Arsitektur BPMS Camunda</i> . . . . .	10
2.6 Forms SDK . . . . .	12
2.6.1 <i>Controls</i> . . . . .	12
2.7 JavaMail . . . . .	14
2.7.1 <i>Java EE</i> . . . . .	14
<b>3 HASIL STUDI</b>	<b>15</b>
3.1 Hasil Studi BPMN . . . . .	15
3.1.1 <i>Masalah Proses Bisnis</i> . . . . .	15
3.1.2 <i>Memodelkan Workflow</i> . . . . .	16
3.2 Menyiapkan BPMS Camunda . . . . .	20
3.2.1 <i>Instalasi Camunda</i> . . . . .	20
3.2.2 <i>Kasus 1 - Pengajuan Proposal</i> . . . . .	21
<b>4 ANALISIS DAN PERANCANGAN</b>	<b>25</b>

4.1	Analisis Hasil Studi . . . . .	25
4.1.1	<i>Event</i> yang Terkait dengan Integrasi Sistem Email . . . . .	25
4.1.2	Mekanisme Integrasi Sistem Email . . . . .	25
4.1.3	Analisis Kebutuhan . . . . .	26
4.1.4	Email . . . . .	26
4.1.5	Algoritma Pengiriman Email . . . . .	27
4.2	Peran Partisipan . . . . .	27
4.2.1	Tugas Desainer . . . . .	27
4.2.2	Tugas Admin . . . . .	27
4.2.3	Tugas Aktor . . . . .	28
4.2.4	Perancangan Aktor . . . . .	28
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>29</b>
5.1	Lingkungan Implementasi . . . . .	29
5.2	Implementasi Kode Program . . . . .	29
5.2.1	Implementasi Algoritma Pengiriman Email . . . . .	29
5.2.2	Implementasi Skenario . . . . .	30
5.3	Pengujian . . . . .	31
5.3.1	Pengujian Skenario 1 . . . . .	31
5.3.2	Pengujian Skenario 2 . . . . .	33
5.4	Hasil Pengujian . . . . .	37
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>39</b>
6.1	Kesimpulan . . . . .	39
6.2	Saran . . . . .	39
	<b>DAFTAR REFERENSI</b>	<b>41</b>
	<b>A KODE PROGRAM PENGIRIMAN EMAIL</b>	<b>43</b>
	<b>B KODE POM.XML</b>	<b>45</b>

## DAFTAR GAMBAR

2.1	Komponen BPM	6
2.2	Siklus BPM	7
2.3	Notasi <i>Event</i>	8
2.4	Notasi <i>Task</i>	8
2.5	Notasi <i>Gateway</i>	8
2.6	Notasi <i>Data</i>	9
2.7	Notasi <i>Artifact</i>	9
2.8	Notasi <i>Lanes dan Pools</i>	9
2.9	Arsitektur BPMS	10
2.10	Arsitektur BPMS Camunda	10
2.11	Camunda Modeler	11
2.12	Camunda Tasklist	11
2.13	Camunda Cockpit	12
2.14	Camunda Admin	12
3.1	Mengunggah Proposal	16
3.2	Mengunggah Proposal	17
3.3	Memeriksa Proposal	17
3.4	Pendaftaran BPJS	19
3.5	Mengunggah Proposal	21
3.6	Memeriksa Proposal	22
3.7	Proposal Layak	22
3.8	Ekspresi Proposal Layak	23
3.9	Proposal tidak Layak	23
3.10	Ekspresi Proposal tidak Layak	23
4.1	Event Task Listener	25
5.1	Mengunggah Proposal	32
5.2	Mengunggah Proposal	32
5.3	Menerima Email	33
5.4	Tasklist Peter	33
5.5	Mengunggah Proposal Group	34
5.6	John Mengunggah Proposal	34
5.7	Mary Mendapat Email	34
5.8	John Mendapat Email	35
5.9	John Mengunggah Proposal	35
5.10	Mary Mengunggah Proposal	35
5.11	Mary mengklaim Task	36
5.12	Mary mengklaim Task	36
5.13	Peter mengklaim Task	36
5.14	John mendapatkan Email	36
5.15	Mary mendapatkan Email	37

## DAFTAR TABEL

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

*Workflow* merupakan pemodelan proses bisnis yang dapat digambarkan sebagai *flow map* atau BPMN (*Business Process Model and Notation*). *Workflow* ini dapat diotomasi menggunakan BPMS (*Business Process Management System*), yaitu sistem yang dapat mengeksekusi dan mengotomasi proses bisnis yang berbentuk *workflow*. Salah satu BPMS yang digunakan di skripsi ini adalah Camunda yang berbasis Java. Agar eksekusi *workflow* lebih alamiah dengan model komunikasi organisasi saat ini, maka *event* yang ada pada *workflow* dapat dipropagasi dan diintegrasikan dengan sistem email. Dengan model komunikasi ini, aktor dapat segera melakukan pekerjaan dari mana dan kapan saja. Hal ini meningkatkan efektifitas dan efisiensi komunikasi pada organisasi.

Dalam skripsi ini, dibuat suatu integrasi antara *user task* dan sistem email. *User task* adalah suatu tugas yang perlu dilakukan oleh pengguna. Ketika ada suatu *user task*, sistem akan mengirimkan email ke pengguna yang akan mengerjakan task tersebut. Email tersebut akan berisi tautan yang mengarah ke tugas yang perlu dikerjakan. Untuk mencapainya, dibuat sebuah *listener* yang dikaitkan di *event* pada *workflow*. *Listener* ini dapat dibuat dengan berbagai bahasa (misalnya Java).

### 1.2 Rumusan Masalah

Berdasarkan latar belakang yang dipaparkan sebelumnya, maka rumusan masalah dalam skripsi ini adalah sebagai berikut :

1. Bagaimana cara kerja BPMN dan BPMS?
2. Bagaimana memodelkan *workflow* dengan BPMN?
3. Event-event *workflow* apa saja yang dapat dipropagasi ke sistem email?
4. Bagaimana mekanisme propagasi dan integrasi *workflow* dengan sistem email?
5. Bagaimana mengimplementasikan dan menguji integrasi *workflow* dengan sistem email?

### 1.3 Tujuan

Berdasarkan rumusan masalah yang dipaparkan sebelumnya, tujuan dari penelitian ini adalah :

1. Mempelajari BPMN dan BPMS.
2. Memodelkan *workflow* dengan BPMN.
3. Mengidentifikasi event-event *workflow* yang dapat dipropagasi ke sistem email.
4. Menentukan mekanisme propagasi dan mengintegrasikan *workflow* dengan sistem email.
5. Menguji integrasi *workflow* dengan sistem email.

### 1.4 Batasan Masalah

1. Pemodelan BPMN menggunakan versi 2.0 dan menggunakan editor Camunda Modeler versi 1.7.2, yaitu versi terbaru untuk pada bulan Mei 2017.
2. Perangkat lunak BPMS Camunda yang digunakan merupakan versi 7.6.0 dan berjalan pada tomcat versi 8.0.24, yaitu versi terbaru pada bulan Mei 2017.
3. Semua uji kasus berada di lingkungan Camunda. Hal ini dilakukan agar skripsi ini lebih fokus kepada integrasi email.
4. Sistem email yang digunakan adalah Google Mail.
5. Menggunakan dua kasus uji.

### 1.5 Metodologi

Metodologi yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Melakukan studi mengenai proses bisnis, *workflow*, *Business Process Model and Notation (BPMN)*, *Business Process Management System (BPMS)*, dan sistem e-mail.
2. Memodelkan proses bisnis tertentu menggunakan BPMN.
3. Mengidentifikasi *event-event* dari *workflow* yang dapat diintegrasikan dengan sistem email.
4. Merancang integrasi sistem email.
5. Mengimplementasikan sistem email ke BPMS.
6. Melakukan pengujian fungsionalitas.

## 1.6 Sistematika Pembahasan

1. Bab 1 Pendahuluan, berisi latar belakang masalah, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian, dan sistematika penulisan.
2. Bab 2 Dasar Teori, berisi dasar teori yang mencakup *Business Process Management*, *Business Process Model and Notation (BPMN)*, *Business Process Management System (BPMS)*, dan sistem e-mail.
3. Bab 3 Analisis, Berisi analisis BPMN dengan menggunakan skenario, analisis event yang terkait dengan sistem email dan mekanisme integrasi sistem email.
4. Bab 4 Perancangan, Berisi rancangan sistem dan rancangan partisipan dalam otomasi BPMS Camunda, .
5. Bab 5 Implementasi, dan Pengujian Berisi implementasi dari program yang dibuat dan pengujian aplikasi berdasarkan contoh kasus pada bab tiga.
6. Bab 6 Penutup, Berisi kesimpulan dan saran-saran untuk pengembangan selanjutnya.





## BAB 2

### DASAR TEORI

Bab dua ini berisi dasar-dasar teori yang terkait dengan BPM, BPMN, BPMS, dan sistem email

#### 2.1 *Business Process* (BP)

*Business Process* adalah kumpulan dari *event*/kejadian, *activity*/kegiatan, dan *decision point*/keputusan serta melibatkan sejumlah aktor dan objek yang bertujuan untuk menghasilkan nilai dalam bentuk produk/jasa yang berguna bagi konsumen<sup>[1]</sup>.

##### 2.1.1 Komponen *Business Process*

*Business Process Management* memiliki komponen-komponen sebagai berikut :

###### *Event*

*Event* adalah kejadian yang terjadi saat proses bisnis berjalan.

###### *Activity*

*Activity* adalah kumpulan kegiatan yang dapat dikerjakan. Ketika suatu *Activity* berupa sebuah kegiatan yang sederhana, *activity* disebut dengan *task*.

###### *Decision Point*

*Decision point* adalah keputusan yang mempengaruhi proses selanjutnya.

###### *Actor*

*Actor* berupa individu, organisasi, maupun sistem yang mempengaruhi proses bisnis.

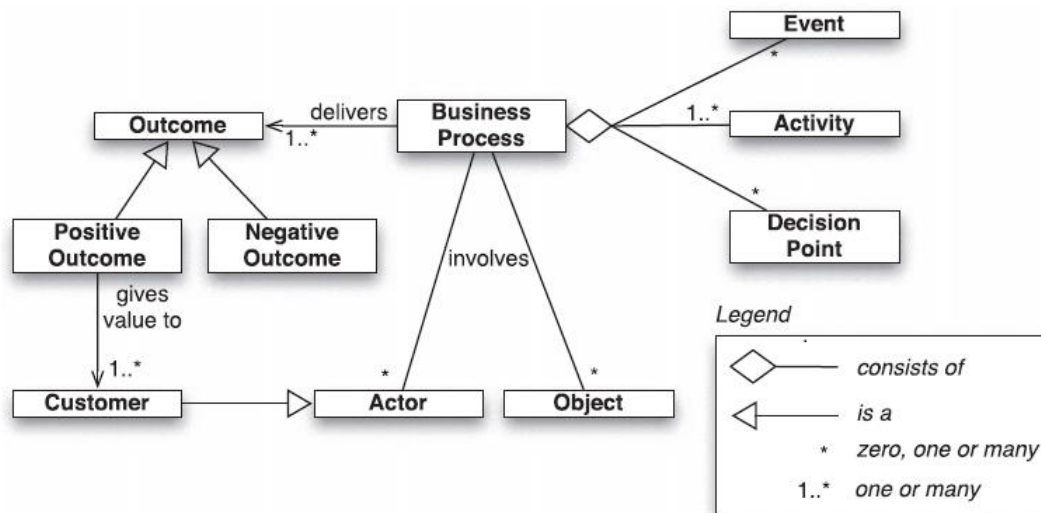
###### *Object*

*Object* dapat berupa objek fisik (peralatan, bahan baku, produk, dokumen) maupun non fisik (dokumen elektronik, basis data elektronik).

###### *Positive/Negative Outcome*

Hasil dari bisnis proses dapat menghasilkan nilai bagi konsumen (positif) atau tidak menghasilkan nilai (negatif).

Komponen-komponen penyusun proses bisnis dapat dilihat pada Gambar [2.1](#).



Gambar 2.1: Komponen BPM

## 2.2 Business Process Management(BPM)

*Business Process Management* merupakan kumpulan metode, teknik, dan alat untuk menemukan, menganalisa, mendesain kembali, menjalankan, dan mengawasi proses bisnis.

### 2.2.1 Siklus *Business Process Management*

Suatu proses bisnis tidak selalu berjalan dengan baik. Banyak hal yang tidak diantisipasi sebelumnya dapat mengganggu proses bisnis. Untuk menjaga kualitas dari sebuah proses bisnis diperlukan pengawasan dan kontrol pada suatu fase tertentu serta perbaikan apabila diperlukan. Maka dari itu, suatu bisnis proses dapat dilihat sebagai suatu siklus yang terus menerus meningkatkan kualitasnya. Siklus dalam proses bisnis berupa :

#### *Process Identification*

Pada fase ini, suatu masalah bisnis ditemukan, kemudian proses-proses yang berhubungan dengan masalah bisnis tersebut diidentifikasi, dibatasi, dan dihubungkan satu sama lain. Proses ini terbagi menjadi dua tahap, yaitu *designation* dan *evaluation*. Tahap *designation* bertujuan untuk mengenali proses-proses yang ada dan hubungan antar proses tersebut. Sedangkan tahap *evaluation* memprioritaskan proses-proses yang menghasilkan nilai dan mempertimbangkan proses yang memiliki risiko atau tidak menghasilkan nilai. Fase ini menghasilkan arsitektur dari proses bisnis yang merepresentasikan proses bisnis dan relasi-relasinya.

#### *Process Discovery*

Setiap proses yang relevan dengan masalah bisnis didokumentasikan, umumnya dalam bentuk model proses. Fase ini menghasilkan *as-is process model*

#### *Process Analysis*

Pada fase ini, masalah pada model proses diidentifikasi, didokumentasikan, dan diukur kinerjanya dengan ukuran yang telah ditetapkan. Hasil dari fase ini adalah kumpulan masalah pada proses model.

### Process Redesign

Tujuan dari fase ini adalah membuat perubahan pada proses yang dapat mengatasi berbagai kumpulan masalah yang telah diidentifikasi pada fase sebelumnya. Proses ini menghasilkan *to-be process model*.

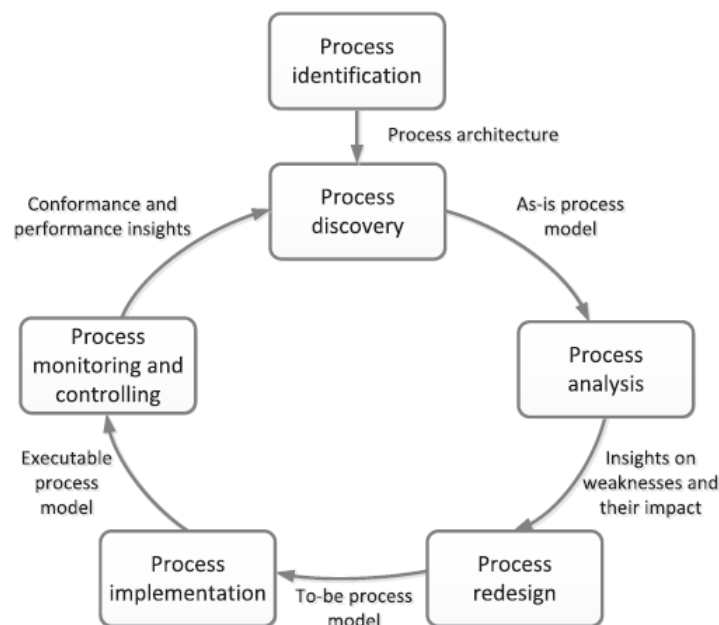
### Process Implementation

Pada fase ini, model proses diimplementasikan untuk dieksekusi menggunakan *Business Process Management System*.

### Process Monitoring and Controlling

Setelah proses bisnis berjalan pada BPMS, berbagai data yang relevan dikumpulkan dan dianalisa untuk menentukan kualitas dari proses. Apabila terdapat masalah baru yang ditemukan, maka proses diulangi.

Siklus BPM dapat dilihat pada Gambar 2.2.



Gambar 2.2: Siklus BPM

## 2.3 Business Process Model and Notation

Business Process Model Notation (BPMN) adalah notasi grafis yang menggambarkan langkah-langkah dalam proses bisnis[2]. Notasi-notasi tersebut terdiri dari *Event*, *Activity*, *Gateway*, *Data*, *Artifact*, *Pools*, dan *Lanes*.

### 2.3.1 Event

Event merupakan kejadian yang terjadi pada proses bisnis yang dilambangkan dengan bentuk lingkaran. Notasi event secara umum terbagi menjadi tiga, yaitu *start event*, *intermediate event*, dan *end event*. *Start event* menunjukkan dimulainya proses, *intermediate*

*event* dapat muncul ketika proses berjalan, sedangkan *end event* menunjukkan berakhirnya proses.

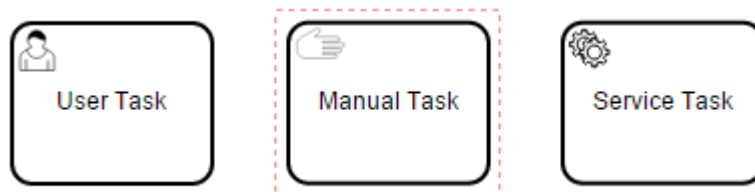


Gambar 2.3: Notasi *Event*

### 2.3.2 Activity

*Activity* merupakan kumpulan kegiatan yang dapat dikerjakan. Sebuah *task* merupakan bagian dari *Activity* yang tidak dapat dipecah lagi. Beberapa jenis dari *Task* adalah :

1. *User Task*, yaitu pekerjaan yang perlu dilakukan oleh manusia melalui sistem. Contohnya adalah mengisi formulir pada halaman web, mengganti password.
2. *Manual Task*, yaitu pekerjaan yang dilakukan manusia tanpa melalui sistem. Contohnya adalah mengirim barang, mengirim surat.
3. *Service Task*, yaitu pekerjaan yang dilakukan oleh sistem dengan mengeksekusi kode. Contohnya adalah notifikasi dari sistem, membangkitkan nomor token.



Gambar 2.4: Notasi *Task*

### 2.3.3 Gateway

*Gateway* merupakan simbol yang menentukan percabangan dan penggabungan jalur dalam proses. Gateway dilambangkan dengan belah ketupat. Beberapa macam adalah :

- *Exclusive Gateway* (XOR) berarti memilih salah satu dari cabang yang ada.
- *Inclusive Gateway* berarti memilih satu, beberapa, atau seluruh cabang yang ada.
- *Parallel Gateway* berarti mengerjakan proses pada seluruh cabang yang ada.
- *Event Based* berarti mengerjakan proses setelah suatu *event* selesai.



Gambar 2.5: Notasi *Gateway*

### 2.3.4 Data

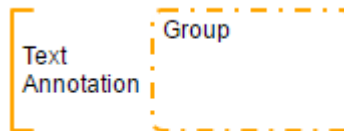
*Data Object* melambangkan informasi yang berjalan dalam proses seperti dokumen, e-mail, atau surat. Sedangkan *Data Store* merupakan tempat proses membaca atau menyimpan data seperti basis data atau rak.



Gambar 2.6: Notasi *Data*

### 2.3.5 Artifact

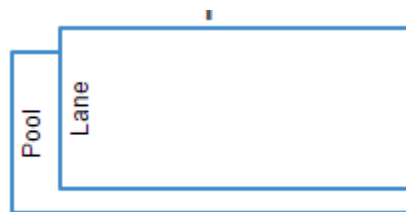
*Artifact* tidak mempengaruhi jalannya proses, tetapi hanya sebagai informasi tambahan agar proses lebih mudah dimengerti. Terdapat dua jenis, yaitu *Text Annotation* dan *Group*



Gambar 2.7: Notasi *Artifact*

### 2.3.6 Pools dan Lanes

*Lanes* digunakan untuk memberikan kumpulan *tasks* kepada yang bertanggung jawab untuk mengerjakannya. Sedangkan *Pools* merupakan kumpulan dari *Lanes*.



Gambar 2.8: Notasi *Lanes dan Pools*

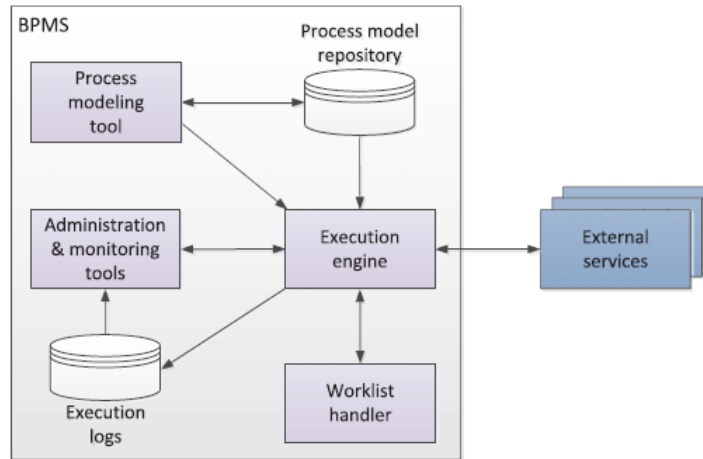
## 2.4 Business Process Management System (BPMS)

*Business Process Management System (BPMS)* adalah sistem yang mengkoordinasikan otomatisasi proses bisnis. Tujuan dari BPMS adalah menyelesaikan proses pada waktu yang ditentukan dan menggunakan sumber daya yang tepat.

Arsitektur BPMS Komponen-komponen BPMS beserta hubungannya yang ditunjukkan pada Gambar 2.9 terdiri dari :

- *Execution Engine*, menyediakan beberapa fungsi seperti mengeksekusi proses, mendistribusikan *task*, mengambil dan menyimpan data yang diperlukan.

- *Process Modeling Tool*, tool untuk membuat model proses.
- *Worklist Handler*, tool untuk mendistribusikan pekerjaan.
- *Administration dan Monitoring Tool tools* untuk administrasi dan memonitor proses.



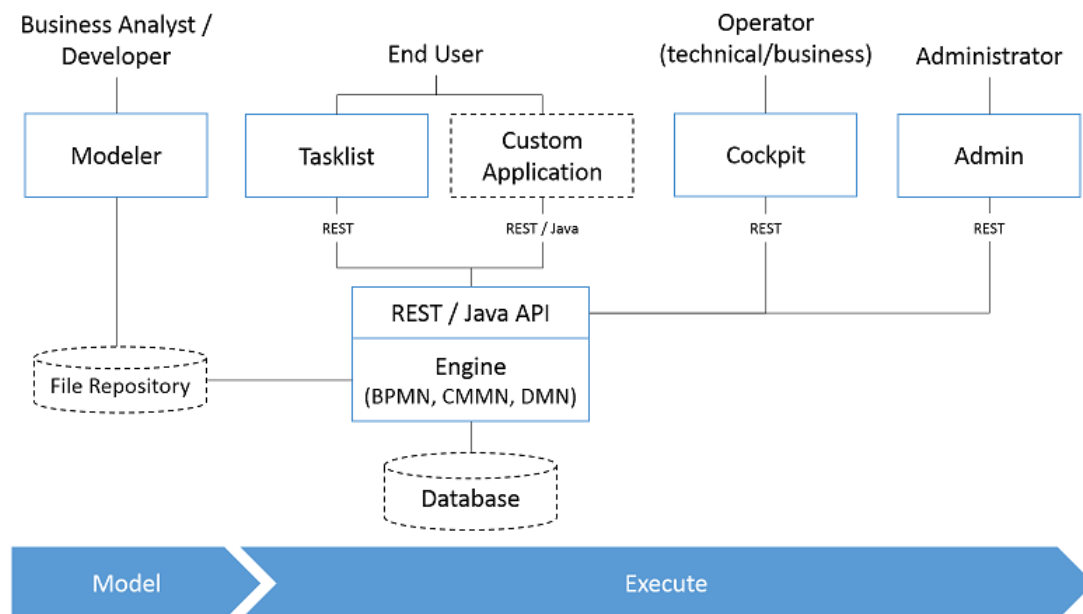
Gambar 2.9: Arsitektur BPMS

## 2.5 BPMS Camunda

Camunda adalah *framework* BPMS berbasis Java yang mendukung *workflow* BPMN dan otomatisasi proses bisnis[3].

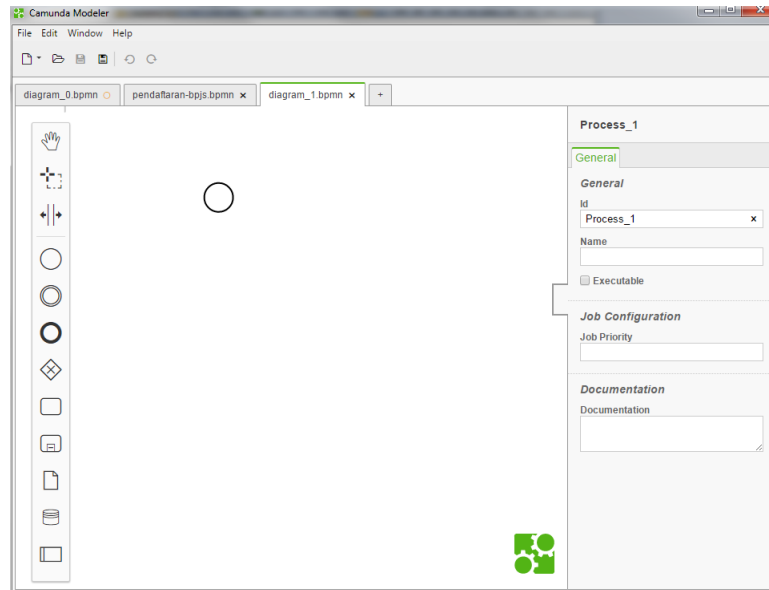
### 2.5.1 Arsitektur BPMS Camunda

Komponen-komponen pada BPMS Camunda adalah sebagai berikut :



Gambar 2.10: Arsitektur BPMS Camunda

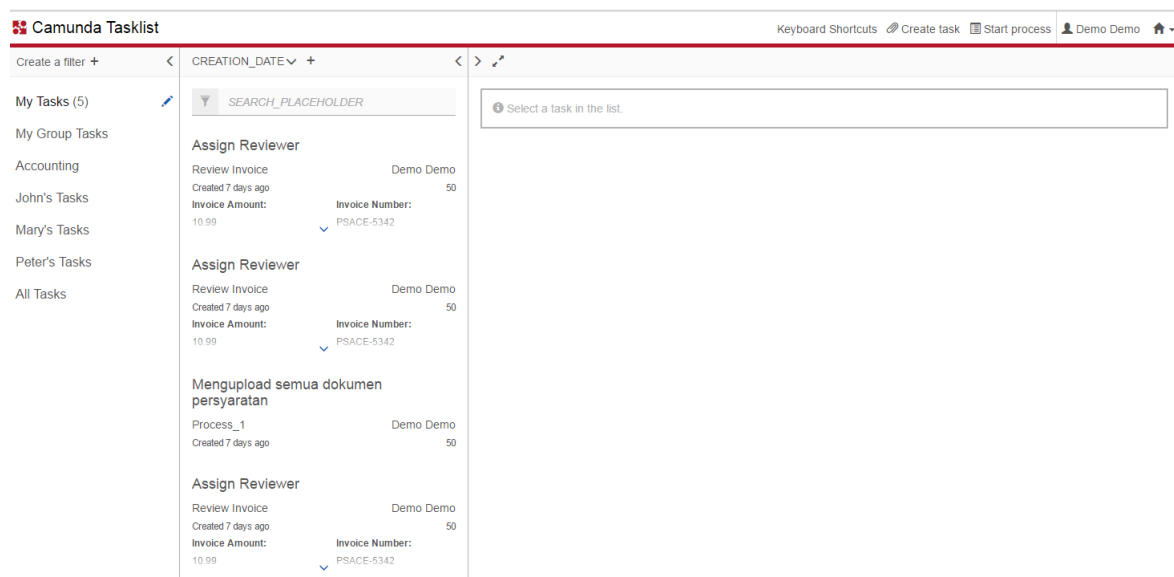
- *Modeler*, *tool* untuk membuat diagram BPMN yang dapat dieksekusi. Camunda Modeler menyediakan berbagai notasi yang diperlukan untuk membuat diagram BPMN. Terdapat pula beberapa pengaturan yang dapat dimasukkan ke dalam notasi.



Gambar 2.11: Camunda Modeler

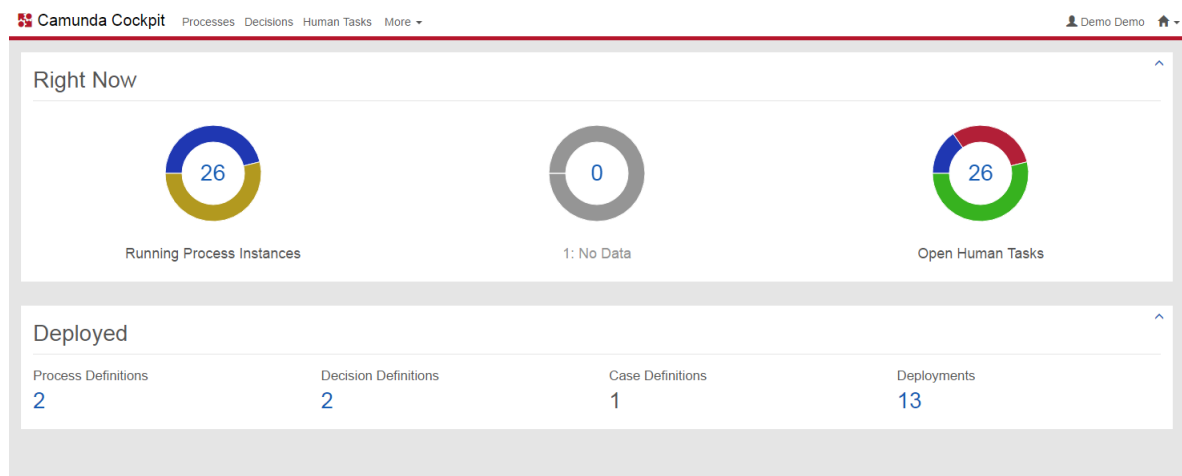
Terdapat tiga bagian utama pada Camunda Modeler, yaitu :

1. Bagian kiri merupakan kumpulan *tool* dan notasi untuk membuat diagram BPMN.
  2. Bagian kanan merupakan pengaturan untuk tiap *event*, *task*, maupun notasi lainnya.
  3. Bagian tengah merupakan tempat membuat diagram BPMN.
- *Tasklist*, tempat pengguna mengakses dan mengerjakan tugas. Tugas yang dikerjakan mengikuti alur model proses (BPMN) yang telah dibuat.



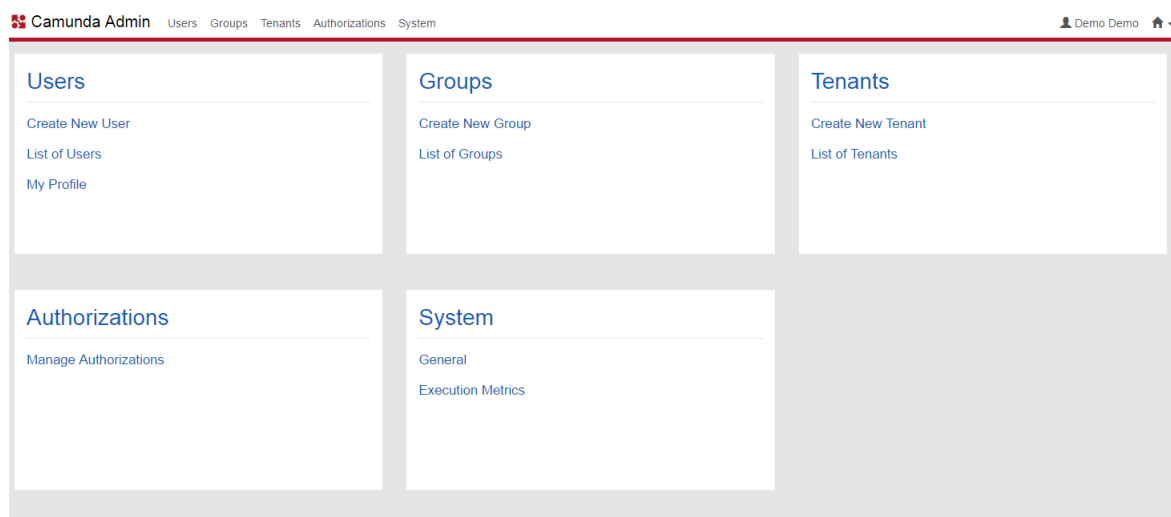
Gambar 2.12: Camunda Tasklist

- *Cockpit*, memeriksa proses yang sedang berjalan maupun proses yang sudah selesai.



Gambar 2.13: Camunda Cockpit

- *Admin*, memiliki tugas untuk mengatur, mengelompokkan, dan memberi izin kepada pengguna untuk melakukan tugas.



Gambar 2.14: Camunda Admin

- *Custom Application*, aplikasi lain yang diintegrasikan dengan Camunda menggunakan Java atau REST API.

## 2.6 Forms SDK

Camunda menggunakan Forms SDK untuk mengimplementasikan *user task* menggunakan aplikasi berbasis HTML5 / JavaScript.

### 2.6.1 Controls

Forms SDK menyediakan instruksi untuk mengakses variabel proses pada *form* HTML. Terdapat dua tipe instruksi yaitu *cam-variable-name* yang digunakan untuk memberi nama



proses / *task* / variabel dan `cam-variable-type` yang digunakan untuk menentukan tipe dari variabel. Elemen HTML yang didukung adalah :

1. *Text Inputs*, untuk memasukkan satu baris teks dan dapat diisi dengan berbagai tipe data seperti String, Integer, Long, Short, dan Double. Kodenya sebagai berikut:

Listing 2.1: Text Input

```
1 | <input type="text" cam-variable-name="CUSTOMER_ID" cam-variable-type="String" />
```

2. *Text Areas*, untuk memasukkan teks dan dapat diisi dengan berbagai tipe data seperti String, Integer, Long, Short, dan Double. Kodenya sebagai berikut:

Listing 2.2: Text Areas

```
1 | <textarea cam-variable-name="CUSTOMER_ADDRESS" cam-variable-type="String"></textarea>
```

3. *Date Inputs*, untuk memasukkan tanggal dengan format yyyy-MM-dd'T'HH:mm:ss (contoh:2017-05-30T11:00:00). Kodenya sebagai berikut:

Listing 2.3: Date Inputs

```
1 | <input type="text"
2 |   cam-variable-name="CONTRACT_START_DATE"
3 |   cam-variable-type="Date" />
```

4. *Boolean Inputs*, terdiri dari tiga tipe, yaitu Checkbox, Select Box, dan Text Inputs (pengguna harus menulis *true* atau *false*. Kodenya sebagai berikut :

Listing 2.4: Boolean Inputs

```
1 |
2 | <input type="checkbox" cam-variable-name="IS_VIP_CUSTOMER" cam-variable-type="Boolean" />
3 |
4 | <select cam-variable-name="APPROVED" cam-variable-type="Boolean">
5 |   <option value="true">Yes</option>
6 |   <option value="false">No</option>
7 | </select>
8 |
9 | <input type="text" cam-variable-name="IS_VIP_CUSTOMER" cam-variable-type="Boolean" />
```

5. *Selects*, untuk memilih salah satu pilihan. Kodenya sebagai berikut :

Listing 2.5: Selects

```
1 | <select cam-variable-name="foo"
2 |   cam-variable-type="String">
3 |   <option>bar</option>
4 |   <option>zar</option>
5 | </select>
```

6. *Hidden Input Fields*, untuk menyembunyikan *form*. Kodenya sebagai berikut :

Listing 2.6: Hidden Input Fields

```
1 | <input type="hidden"
2 |   cam-variable-name="CUSTOMER_ID"
3 |   cam-variable-type="String"
4 |   value="testuser" />
```

7. *Upload* dan *Download*, untuk mengunggah maupun mengunduh file. Kode untuk mengunggah file sebagai berikut :

Listing 2.7: Upload

```

1 | <input type="file"
2 |     cam-variable-name="INVOICE_DOCUMENT"
3 |     cam-variable-type="File"
4 |     cam-max-file-size="1000000" />

```

Sedangkan kode untuk mengunduh file sebagai berikut :

Listing 2.8: Download

```

1 | <a cam-file-download="INVOICE_DOCUMENT"></a>

```

Nama variabel pada `cam-variable-name` harus sama dengan nama variabel pada `cam-file-download`.

## 2.7 JavaMail

JavaMail adalah Java API yang digunakan untuk mengirim dan menerima email melalui SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol 3), dan IMAP (Internet Message Access Protocol)[4]. JavaMail dibuat dalam lingkungan Java EE.

### 2.7.1 Java EE

Java Platform, Enterprise Edition (Java EE) adalah lingkungan komputasi untuk pengembangan perangkat lunak

Untuk menggunakan JavaMail diperlukan beberapa kelas, yaitu :

- Kelas Properties, kelas untuk mengatur standar email yang akan digunakan, seperti SMTP, IMAP, POP3, dan lainnya.
- Kelas MimeMessage, kelas untuk menulis email.
- Kelas Transport, kelas untuk membuat koneksi ke email *server* dan mengirim email.

Listing 2.9: Contoh Kode Pengiriman Email

```

1 | Properties props = new Properties();
2 | props.put("mail.smtp.host", "my-mail-server");
3 | Session session = Session.getInstance(props, null);
4 |
5 | try {
6 |     MimeMessage msg = new MimeMessage(session);
7 |     msg.setFrom("me@example.com");
8 |     msg.setRecipients(Message.RecipientType.TO,
9 |         "you@example.com");
10 |    msg.setSubject("JavaMail_hello_world_example");
11 |    msg.setSentDate(new Date());
12 |    msg.setText("Hello , world!\n");
13 |    Transport.send(msg, "me@example.com", "my-password");
14 | } catch (MessagingException mex) {
15 |     System.out.println("send_failed , exception:" + mex);
16 | }

```

## BAB 3

### HASIL STUDI

Bab ini berisi hasil studi terhadap *Business Process Model and Notation* dan *Business Process Management System* Camunda.

#### 3.1 Hasil Studi BPMN

Setiap bisnis memiliki alur kerja maupun proses yang perlu dilewati. Proses tersebut dapat digambarkan dalam bentuk *Business Process Model and Notation*. BPMN merupakan sebuah standar untuk menggambarkan langkah-langkah pada suatu proses bisnis. Dengan BPMN, suatu proses bisnis yang kompleks dapat digambarkan menjadi lebih sederhana sehingga lebih mudah dimengerti. BPMN memiliki berbagai notasi seperti *event*, *task*, *gateway*, *data*, *artifact*, *lanes*, dan *pool*.

##### 3.1.1 Masalah Proses Bisnis

Berikut ini berbagai masalah proses bisnis yang akan dimodelkan pada *workflow* :

##### Pengajuan Proposal

Pegawai di perusahaan X memiliki tiga divisi yaitu *accounting*, *sales*, dan *management*. Divisi *accounting* dan *sales* dapat mengajukan proposal bisnis ke divisi *management*. Divisi *management* harus memeriksa apakah proposalnya layak atau tidak. Jika proposalnya tidak layak, pembuat proposal harus memperbaiki dan mengunggahnya kembali. Workflow dari skenario ini sebagai berikut :

##### Proses Pendaftaran BPJS

1. Pemohon mengisi formulir pendaftaran BPJS di situs BPJS (termasuk jenis keanggotaan).
2. Pemohon mengupload semua dokumen persyaratan di situs BPJS.
3. Sistem BPJS membangkitkan nomor pembayaran uang pendaftaran/ iuran pertama (nomor pembayaran selanjutnya menjadi nomor keanggotaan/ kartu BPJS).
4. Pemohon melihat nomor pembayaran dan besarnya uang pendaftaran/ iuran pertama.
5. Pemohon membayar uang pendaftaran/ iuran pertama melalui bank sesuai nomor pembayaran (paling lambat 3 hari setelah pendaftaran, jika lebih maka pendaftaran hangus).

6. Pemohon memilih jadwal verifikasi dokumen asli yang tersedia.
7. Sistem BPJS membangkitkan jadwal kedatangan dan nomor antrian.
8. Pemohon mencetak jadwal kedatangan dan nomor antriannya.
9. Pemohon datang ke kantor BPJS membawa dokumen asli (sesuai jadwal, jika tidak maka pendaftaran hangus).
10. Petugas BPJS memverifikasi pendaftaran, dan attachment dokumen persyaratan dan keasliannya. Jika valid dan lengkap, proses dilanjutkan ke langkah 11, jika tidak lengkap maupun tidak valid, maka kembali ke langkah 1.
11. Sistem BPJS membangkitkan barcode untuk kartu BPJS.
12. Petugas BPJS mencetak kartu BPJS dan meyerahkannya ke Pemohon.

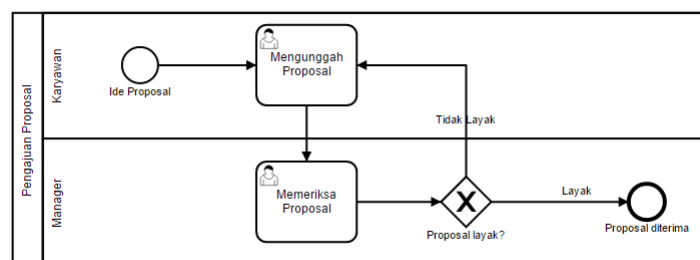
Workflow dari proses bisnis ini adalah sebagai berikut :

### 3.1.2 Memodelkan Workflow

#### Pengajuan Proposal

Pada kasus Pengajuan Proposal, terdapat beberapa elemen, yaitu :

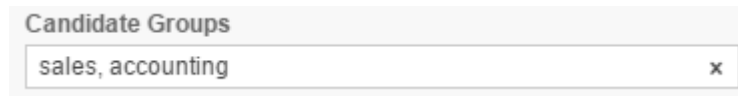
1. Satu *pool*, yaitu Pengajuan Proposal
2. Dua *lane*, yaitu lane untuk Pegawai dan Manajemen
3. Dua *event*, yaitu *start event* (Ide Proposal) dan *end event* Proposal Diterima
4. Dua *user task*, yaitu Mengunggah Proposal oleh pegawai dan Memeriksa Proposal oleh manajemen
5. Satu *decision point*, yaitu penentuan apakah proposal layak atau tidak



Gambar 3.1: Mengunggah Proposal

#### User task Mengunggah Proposal

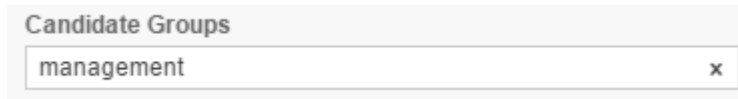
*Task* ini dapat dilakukan oleh pegawai (divisi *sales* dan *accounting*). Maka *Candidate Groups* diisi dengan *sales* dan *accounting*.


 A screenshot of a web form titled "Candidate Groups". Below the title is a text input field containing the text "sales, accounting". To the right of the input field is a small "x" icon for clearing the text.

Gambar 3.2: Mengunggah Proposal

*User task* Memeriksa Proposal

*Task* ini dilakukan oleh Manajemen. Maka *Candidate Groups* diisi dengan *management*.


 A screenshot of a web form titled "Candidate Groups". Below the title is a text input field containing the text "management". To the right of the input field is a small "x" icon for clearing the text.

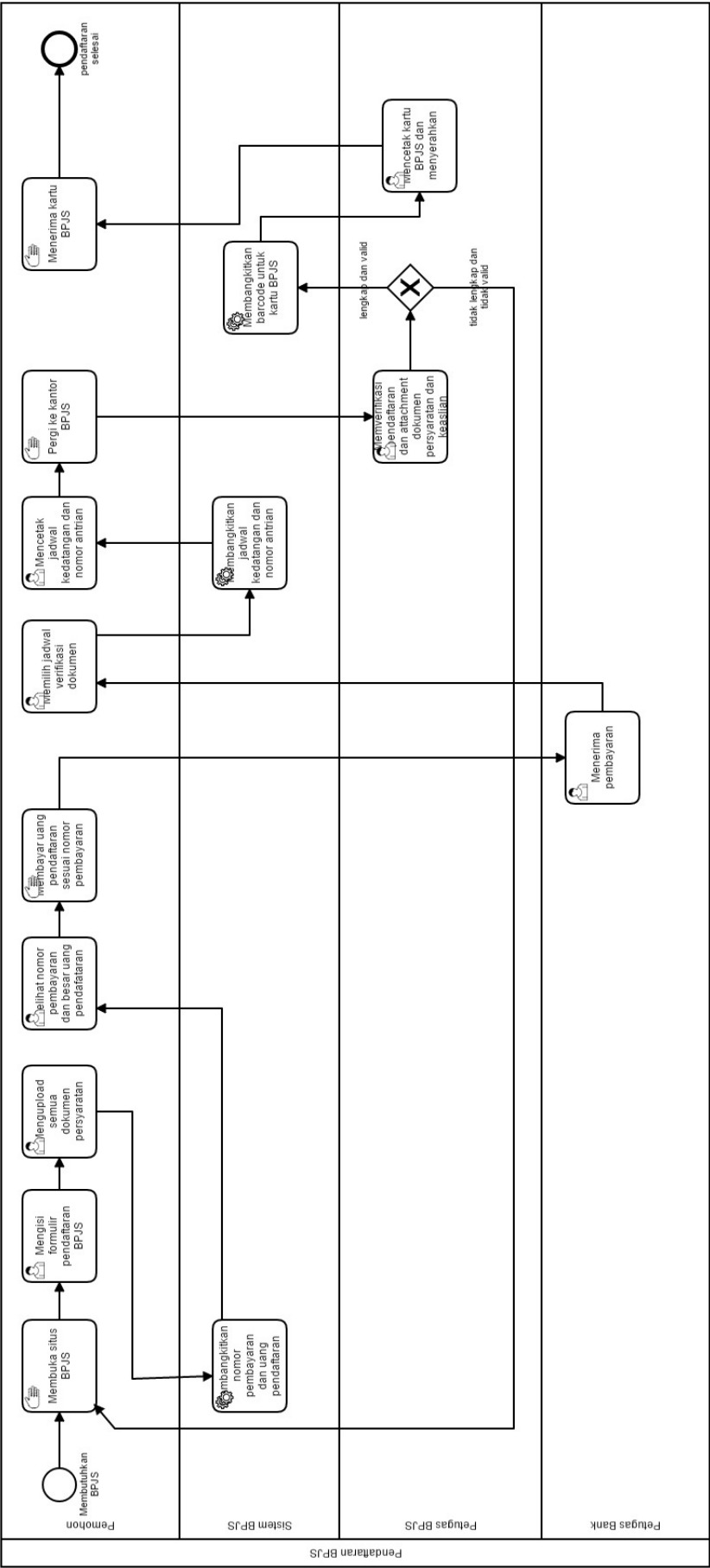
Gambar 3.3: Memeriksa Proposal

### Pendaftaran BPJS

Pada kasus Pendaftaran BPJS, terdapat beberapa elemen, yaitu :

1. Satu *pool*, yaitu Pendaftaran BPJS
2. Empat *lane*, yaitu lane untuk Pemohon (diwakilkan oleh John), Sistem BPJS, Petugas BPJS (diwakilkan oleh Mary) dan Petugas Bank (diwakilkan oleh Peter)
3. Dua *event*, yaitu *start event* Membutuhkan BPJS dan *end event* Pendaftaran Selesai
4. Empat *manual task* Pemohon, yaitu :
  - (a) Membuka situs BPJS
  - (b) Membayar uang pendaftaran sesuai nomor pembayaran
  - (c) Pergi ke kantor BPJS
  - (d) Menerima Kartu BPJS
5. Tiga *service task* Sistem BPJS, yaitu :
  - (a) Membangkitkan nomor pembayaran dan uang pendaftaran
  - (b) Membangkitkan jadwal kedatangan dan nomor antrian
  - (c) Membangkitkan barcode untuk kartu BPJS
6. Lima *User task* Pemohon, yaitu :
  - (a) Mengisi formulir pendaftaran BPJS
  - (b) Mengupload semua dokumen persyaratan
  - (c) Melihat nomor pembayaran dan besar uang pendaftaran
  - (d) Memilih jadwal verifikasi dokumen
  - (e) Mencetak jadwal kedatangan dan nomor antrian
7. Dua *User task* Petugas BPJS, yaitu :
  - (a) Memverifikasi pendaftaran dan *attachment* dokumen persyaratan dan keaslian

- (b) Mencetak kartu BPJS dan menyerahkannya
- 8. Satu *User task* Petugas Bank, yaitu menerima pembayaran
- 9. Satu *decision point*, yaitu penentuan apakah persyaratan pendaftaran lengkap dan valid



Gambar 3.4: Pendaftaran BPJS

Untuk setiap user task, ditentukan pemiliknya *task* masing-masing. Pemohon diwakilkan oleh John, Petugas BPJS diwakilkan oleh Mary, dan Petugas Bank diwakilkan oleh Peter. Sehingga atribut assignee di setiap *user task* akan diisi dengan john, mary, atau peter. Contohnya adalah *task* Mengisi formulir pendaftaran BPJS yang dimiliki oleh John memiliki atribut *assignee* john.

The image shows a web-based configuration interface for a Camunda UserTask. The title bar reads 'UserTask\_15dx9zp'. Below the title are five tabs: 'General' (highlighted with a green border), 'Forms', 'Listeners', 'Input/Output', and 'Extensions'. The 'General' tab is open, showing two sections. The 'General' section has two text input fields: 'Id' with the value 'UserTask\_15dx9zp' and 'Name' with the value 'Mengisi formulir pendaftaran BPJS'. The 'Details' section has one text input field: 'Assignee' with the value 'john'. Each input field has a small 'x' icon to its right for clearing the text.

Gambar 3.5: Atribut *assignee* dari Mengisi formulir pendaftaran BPJS

## 3.2 Menyiapkan BPMS Camunda

Bagian ini akan menjelaskan cara instalasi Camunda, menghubungkan *form* HTML maupun *script* Java dengan BPMN, dan menjalankan otomasi proses bisnis menggunakan BPMS Camunda.

### 3.2.1 Instalasi Camunda

Untuk menjalankan Camunda, diperlukan beberapa *tool*<sup>[5]</sup>, yaitu :

- Java JDK 1.7+.
- Apache Maven atau Maven yang sudah terpasang di Eclipse.
- Web browser.
- Camunda BPM Platform
- Camunda Modeler

### Mempersiapkan Proyek Java

Membuat Proyek Maven di Eclipse.

1. Pilih File / New / Other / Maven / Maven Project kemudian pilih *Next*.
2. Pilih Create a simple project (skip archetype selection) kemudian pilih *next*.
3. Pilih Packaging : war, kemudian pilih Finish.



Tambahkan *Camunda Maven Dependencies* ke file pom.xml (lihat Lampiran B).

Tambahkan sebuah kelas Process Application pada direktori src/main/java. Nama kelas dapat diganti dengan nama proses yang dibuat.

Listing 3.1: Kelas Process Application

```

1 | package org.camunda.bpm.getstarted.loanapproval;
2 |
3 | import org.camunda.bpm.application.ProcessApplication;
4 | import org.camunda.bpm.application.impl.ServletProcessApplication;
5 |
6 | @ProcessApplication("LoanApprovalApp")
7 | public class LoanApprovalApplication extends ServletProcessApplication {
8 |     // empty implementation
9 | }
```

Tambahkan *Deployment Descriptor* di META-INF/processes.xml.

Listing 3.2: processes.xml

```

1 | <?xml version="1.0" encoding="UTF-8" ?>
2 |
3 | <process-application
4 |     xmlns="http://www.camunda.org/schema/1.0/ProcessApplication"
5 |     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
6 |
7 |     <process-archive name="loan-approval">
8 |         <process-engine>default</process-engine>
9 |         <properties>
10 |             <property name="isDeleteUponUndeploy">false</property>
11 |             <property name="isScanForProcessDefinitions">true</property>
12 |         </properties>
13 |     </process-archive>
14 |
15 | </process-application>
```

### 3.2.2 Kasus 1 - Pengajuan Proposal

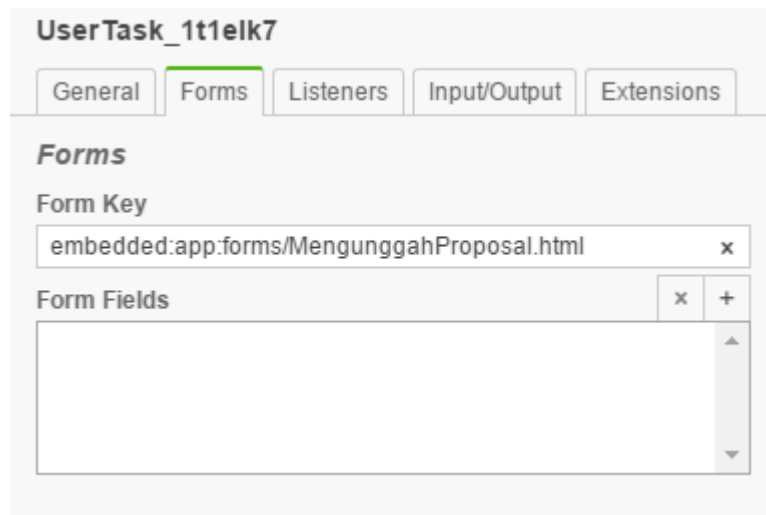
Untuk menjalankan proses bisnis pengajuan proposal menggunakan BPMS Camunda, langkah-langkah yang perlu dilakukan adalah :

1. Menambah Form HTML untuk setiap *user task* dan menghubungkannya dengan BPMN. File HTML yang dibuat disimpan di direktori src/main/webapp/forms. Proses bisnis ini memiliki dua *user task*, yaitu mengunggah proposal dan memeriksa proposal.
  - (a) *User task* Mengunggah Proposal Task ini merupakan *user task* sehingga membutuhkan suatu *form* HTML untuk mengunggah proposal. Pada *form* MengunggahProposal, isi dari variabel cam-variable-name adalah proposal yang akan digunakan pada *form* tempat proposal diunduh.

Listing 3.3: MengunggahProposal.html

```

1 | <html>
2 | <head></head>
3 | <body>
4 |     <form method="post" name="upload-dokumen">
5 |         <input type="file"
6 |             cam-variable-name="proposal"
7 |             cam-variable-type="File"
8 |             cam-max-file-size="10000000"/>
9 |     </form>
10 | </body>
11 | </html>
```



Gambar 3.6: Mengunggah Proposal

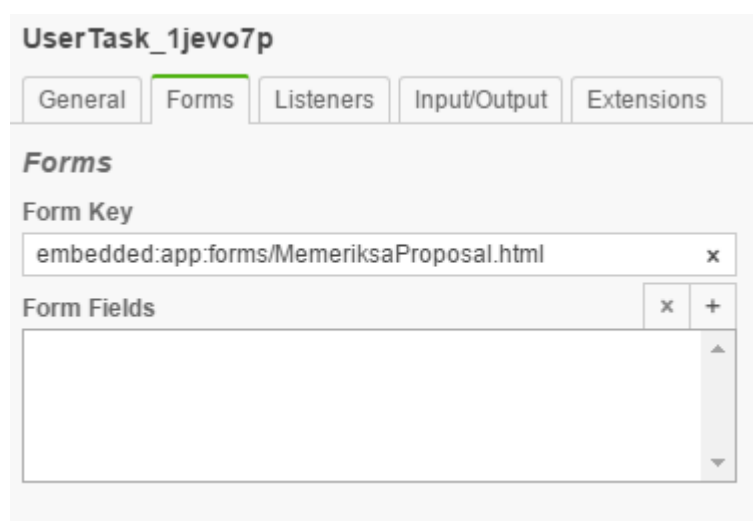
- (b) Task ini merupakan *user task* sehingga membutuhkan suatu *form* HTML untuk memeriksa proposal. Isi dari variabel *cam-file-download* adalah "proposal" (sama dengan *cam-variable-name* pada *form* mengunggah proposal, sehingga file yang diunduh sama dengan file yang diunggah. Kemudian terdapat *checkbox* untuk menentukan apakah proposal sudah layak atau belum. *Checkbox* ini memiliki *cam-variable-name* dengan nama "valid" yang nantinya akan digunakan pada *gateway*.

Listing 3.4: MemeriksaProposal.html

```

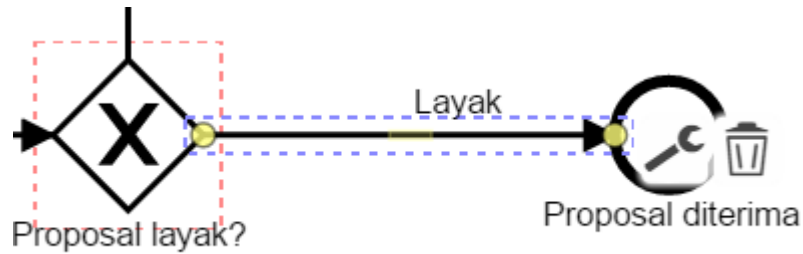
1 | <html>
2 | <head></head>
3 | <body>
4 | <form role="form" name="form">
5 |   <a cam-file-download="proposal">Download Dokumen</a>
6 |   <p>Apakah Proposal layak?</p>
7 |   <input cam-variable-name="valid"
8 |         cam-variable-type="Boolean"
9 |         type="checkbox"
10 |        name="valid"
11 |        class="form-control" />
12 | </form>
13 | </body>
14 | </html>

```



Gambar 3.7: Memeriksa Proposal

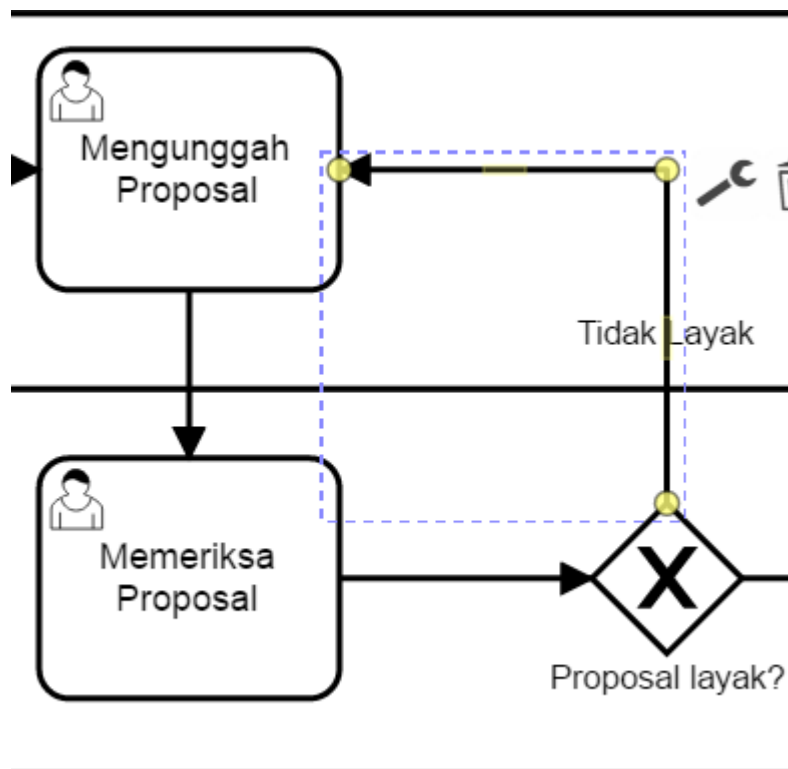
2. Mengatur *Gateway*, untuk mengatur keluaran dari *gateway* pengaturan dapat dilakukan pada modeler dengan menggunakan `cam-variable-name = valid` pada checkbox di form `MemeriksaProposal`. Apabila proposal layak, maka *expression* yang digunakan adalah `$(valid)`. Jika proposal tidak layak, *expression* yang digunakan adalah `$(!valid)`.



Gambar 3.8: Proposal Layak

Details	
Condition Type	Expression ▼
Expression	<code>\$(valid)</code> x

Gambar 3.9: Ekspresi Proposal Layak



Gambar 3.10: Proposal tidak Layak



**Details**

Condition Type

Expression ▼

Expression

`${!valid}` ✕

Gambar 3.11: Ekspresi Proposal tidak Layak

3. Menyimpan file BPMN ke direktori `src/main/resources` pada proyek pengajuan proposal

### Menjalankan Camunda

1. Klik kanan `pom.xml` dan pilih `Run As / Maven Install`. Langkah ini akan menghasilkan file WAR di folder `target`.
2. *Copy paste* file WAR ke `CAMUNDA_HOME / server / apache-tomcat / webapps` folder.
3. Jalankan `start-camunda.bat`

## BAB 4

### ANALISIS DAN PERANCANGAN

#### 4.1 Analisis Hasil Studi

Berdasarkan hasil studi di bab sebelumnya, beberapa hal yang perlu dianalisis adalah *event* apa yang diintegrasikan dengan sistem email dan bagaimana mekanisme integrasi sistem email

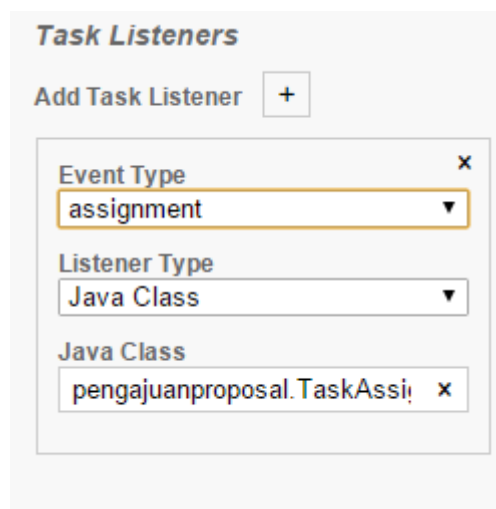
##### 4.1.1 *Event* yang Terkait dengan Integrasi Sistem Email

Integrasi Camunda dengan sistem email pada skripsi ini bertujuan untuk memberi tahu aktor Camunda apabila ada *tasks* yang perlu dikerjakan oleh aktor. Ketika aktor menerima email mengenai *tasks* yang perlu dikejakan, aktor dapat langsung mengerjakannya.

Camunda memiliki berbagai jenis *tasks* seperti *user tasks*, *manual tasks*, *service task*, dan lainnya. Karena proses integrasi email dengan Camunda melibatkan aktor (aktor menerima pemberitahuan pekerjaannya melalui email), *task* yang akan diintegrasikan dengan sistem email adalah *user tasks*.

##### 4.1.2 Mekanisme Integrasi Sistem Email

*User tasks* memiliki atribut *Task Listener* yang dapat mengeksekusi perintah. *Task Listener* memiliki dua atribut, yaitu *Event Type* dan *Listener Type*. Terdapat empat pilihan dari *Event Type*, yaitu *create*, *assignment*, *complete*, *delete*.



The screenshot shows the 'Task Listeners' configuration window in Camunda. At the top, there is a title 'Task Listeners' and a button 'Add Task Listener' with a plus icon. Below this, there is a form with three fields: 'Event Type' with a dropdown menu showing 'assignment', 'Listener Type' with a dropdown menu showing 'Java Class', and 'Java Class' with a text input field containing 'pengajuanproposal.TaskAssi'. Each field has a close button (X) in the top right corner.

Gambar 4.1: Event Task Listener

- Create, perintah dieksekusi ketika *task* telah dibuat dan siap untuk dikerjakan.
- Assignment, perintah dieksekusi ketika aktor yang akan mengerjakan *task* sudah ditentukan.
- Complete, perintah dieksekusi ketika *task* sudah dikerjakan dan sebelum *task* dihapus.
- Delete, perintah dieksekusi setelah *task* dihapus.

Untuk mengintegrasikan *user tasks* dengan email, *event type* yang dapat digunakan adalah *create* dan *assignment*. *Event complete* dan *delete* tidak dapat digunakan untuk memberi tahu aktor karena setelah *task* selesai dan dihapus, alamat email untuk *Task* selanjutnya belum diambil sementara *event* sudah selesai dipanggil.

Apabila menggunakan *event create*, *task* harus memiliki pemiliknya masing-masing ketika BPMN dibuat atau memiliki *candidate user/group*. Bila pemilik *task* belum ditentukan, email tidak akan terkirim, karena *event create* sudah selesai dipanggil sebelum *task* memiliki pemilik. Pengiriman email untuk *task* yang belum memiliki aktor dapat menggunakan *event create*. Sedangkan pada *event assignment*, pengiriman email dilakukan setelah *task* didelegasikan ke masing-masing user.

### 4.1.3 Analisis Kebutuhan

Berdasarkan analisis di bagian sebelumnya, maka untuk mempropagasi email diperlukan beberapa persyaratan, yaitu :

1. Model proses menggunakan BPMN yang sudah dilengkapi form HTML untuk *user task*, implementasi untuk *service task* dan atribut lain yang diperlukan.
2. Kumpulan *user/group* yang akan mengerjakan tugas.
3. Alamat email yang merepresentasikan sistem.
4. Algoritma untuk mengirim email.
5. Business Process Management System (BPMS), yaitu tools untuk mengotomasi jalannya proses.

### 4.1.4 Email

Alamat email yang digunakan untuk merepresentasikan sistem berbasis Gmail SMTP. Gmail SMTP yang akan digunakan memiliki konfigurasi sebagai berikut [6] :

- Alamat server = smtp.gmail.com.
- Port = 587.
- Username Gmail.
- Password Gmail.

Email yang akan dikirimkan ke aktor memiliki format :

1. Subjek :

2. Nama aktor.
3. Nama *task*.
4. Link ke *task*, yaitu [http://localhost/camunda/app/tasklist/default/#/?task=\(id task\)](http://localhost/camunda/app/tasklist/default/#/?task=(id task)).

#### 4.1.5 Algoritma Pengiriman Email

Berikut adalah algoritma untuk mengirimkan email.

1. Mengambil id dari *task*.
2. Mengambil email aktor yang akan mengerjakan *task*.
3. Membangkitkan subjek dan isi email yang berisi tautan ke task yang akan dikerjakan. Tautan didapatkan dari id *task*.
4. Membuat koneksi ke email server dengan *username* dan *password*
5. Mengirim email.

## 4.2 Peran Partisipan

Setiap partisipan memiliki perannya masing-masing. Desainer bertugas merancang BPMN, admin bertugas mengatur jalannya otomasi proses bisnis, sedangkan aktor bertugas mengerjakan *tasks*

### 4.2.1 Tugas Desainer

Berdasarkan perancangan sistem di atas, seorang desainer model proses memiliki beberapa tugas, yaitu :

1. Merancang model proses.
2. Menambahkan form HTML pada *user task*, *implementasi service task*, *task listener* untuk propagasi email, dan berbagai atribut lainnya sesuai kebutuhan.
3. Mendelegasikan task kepada user/group yang akan mengerjakan.

### 4.2.2 Tugas Admin

1. Membuat alamat email yang merepresentasikan sistem.
2. Menambahkan *username*, *password*, dan *host* email pada kode task listener yang berhubungan dengan propagasi email.
3. Menambahkan user/group yang akan mengerjakan *tasks* pada Camunda Admin.
4. Menjalankan dan memulai proses.

### 4.2.3 Tugas Aktor

1. Memberitahu alamat email kepada admin.
2. Mengerjakan *task*.

### 4.2.4 Perancangan Aktor

Untuk pengujian skenario, ada beberapa aktor yang dibuat, yaitu :

1. John, dengan alamat email johncamunda@gmail.com dan bagian dari grup *sales*.
2. Mary, dengan alamat email marycamunda@gmail.com dan bagian dari grup *accounting*.
3. Peter, dengan alamat email petercamunda@gmail.com dan bagian dari grup *management*.



## BAB 5

### IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan diimplementasikan kode program untuk propagasi email dan pengujian dua skenario yang ada pada Bab ??.

#### 5.1 Lingkungan Implementasi

Implementasi dilakukan pada lingkungan :

1. Eclipse 4.5 Mars
2. BPMN versi 2.0 dan Camunda Modeler versi 1.7.2.
3. BPMS Camunda versi 7.6.0 dan berjalan pada tomcat versi 8.0.24.

#### 5.2 Implementasi Kode Program

##### 5.2.1 Implementasi Algoritma Pengiriman Email

Beberapa potongan kode di bawah ini adalah kode untuk pengiriman email. Kode secara keseluruhan dapat dilihat pada Lampiran [A](#)

- Konfigurasi email admin.

Listing 5.1: TaskAssignmentListener.java

```
1 | private static final String HOST = "smtp.gmail.com";  
2 | private static final String USER = "camundasys@gmail.com";  
3 | private static final String PWD = "epW3S4KN";
```

- Kode untuk mengambil assignee (aktor dari *task*, mengambil id *task*, dan mengambil alamat email aktor.

Listing 5.2: TaskAssignmentListener.java

```
1 |  
2 | public void notify(DelegateTask delegateTask) {  
3 |     String assignee = delegateTask.getAssignee();  
4 |     String taskId = delegateTask.getId();
```

- Konfigurasi SMTP Gmail.

Listing 5.3: TaskAssignmentListener.java

```
1 |  
2 | props = System.getProperties();  
3 | props.put("mail.smtp.port", "587");  
4 | props.put("mail.smtp.auth", "true");  
5 | props.put("mail.smtp.starttls.enable", "true");
```

- Kode untuk mendapatkan aktor apabila atribut assignee memiliki nilai.

Listing 5.4: TaskAssignmentListener.java

```

1  if (assignee != null) {
2      IdentityService identityService = Context.getProcessEngineConfiguration().
        getIdentityService();
3      User user = identityService.createUserQuery().userId(assignee).singleResult();
4      if (user != null) {
5          this.sendEmail(user);
6      }
7  }

```

- Kode untuk mendapatkan aktor apabila atribut assignee tidak memiliki nilai.

Listing 5.5: TaskAssignmentListener.java

```

1      TaskEntity task = (TaskEntity)delegateTask;
2      List<IdentityLinkEntity> identityLinks = task.getIdentityLinks();
3
4      for (IdentityLinkEntity link : identityLinks) {
5          if (link.getType().equals(IdentityLinkType.CANDIDATE)) {
6              if (link.isUser()) {
7                  User user = Context.getProcessEngineConfiguration().getIdentityService
                        ().createUserQuery().userId(link.getUserId()).singleResult();
8                  sendEmail(user);
9              }
10             if (link.isGroup()) {
11                 List<User> users = Context.getProcessEngineConfiguration().
                        getIdentityService().createUserQuery().memberOfGroup(link.
                        getGroupId()).list();
12                 for (User user : users) {
13                     sendEmail(user);
14                 }
15             }
16         }
17     }

```

- Kode untuk membangkitkan subjek dan isi email

Listing 5.6: TaskAssignmentListener.java

```

1  session = Session.getDefaultInstance(props, null);
2      message = new MimeMessage(session);
3      message.addRecipient(Message.RecipientType.TO, new InternetAddress(recipient)
4      );
5      message.setSubject("Task" + delegateTask.getName());
6
7      String emailBody = user.getFirstName() + "<br>";
8      emailBody += "Tolong Selesaikan Task" + taskName + " dibawah ini.<br>";
9      emailBody += "http://localhost:1234/camunda/app/tasklist/default/#/?task="+taskId;
10     message.setContent(emailBody, "text/html");

```

- Kode untuk mengirimkan email.

Listing 5.7: TaskAssignmentListener.java

```

1
2  Transport transport = session.getTransport("smtp");
3      transport.connect(HOST, USER, PWD);
4      transport.sendMessage(message, message.getAllRecipients());
5      transport.close();

```

## 5.2.2 Implementasi Skenario

### Pengajuan Proposal Bisnis dan Pengajuan Proposal Bisnis dari Grup

Kode ini adalah kode file HTML untuk Skenario ?? dan Skenario ?. Terdapat dua form HTML yaitu MengunggahDokumen.html dan MemeriksaDokumen.html.

Listing 5.8: MengunggahDokumen.html

```

1| <html>
2| <head>
3| <body>
4|     <form method="post" name="upload-dokumen">
5|         <input type="file"
6|             cam-variable-name="proposal"
7|             cam-variable-type="File"
8|             cam-max-filesize="10000000" />
9|     </form>
10| </body>
11| </html>

```

Listing 5.9: MemeriksaDokumen.html

```

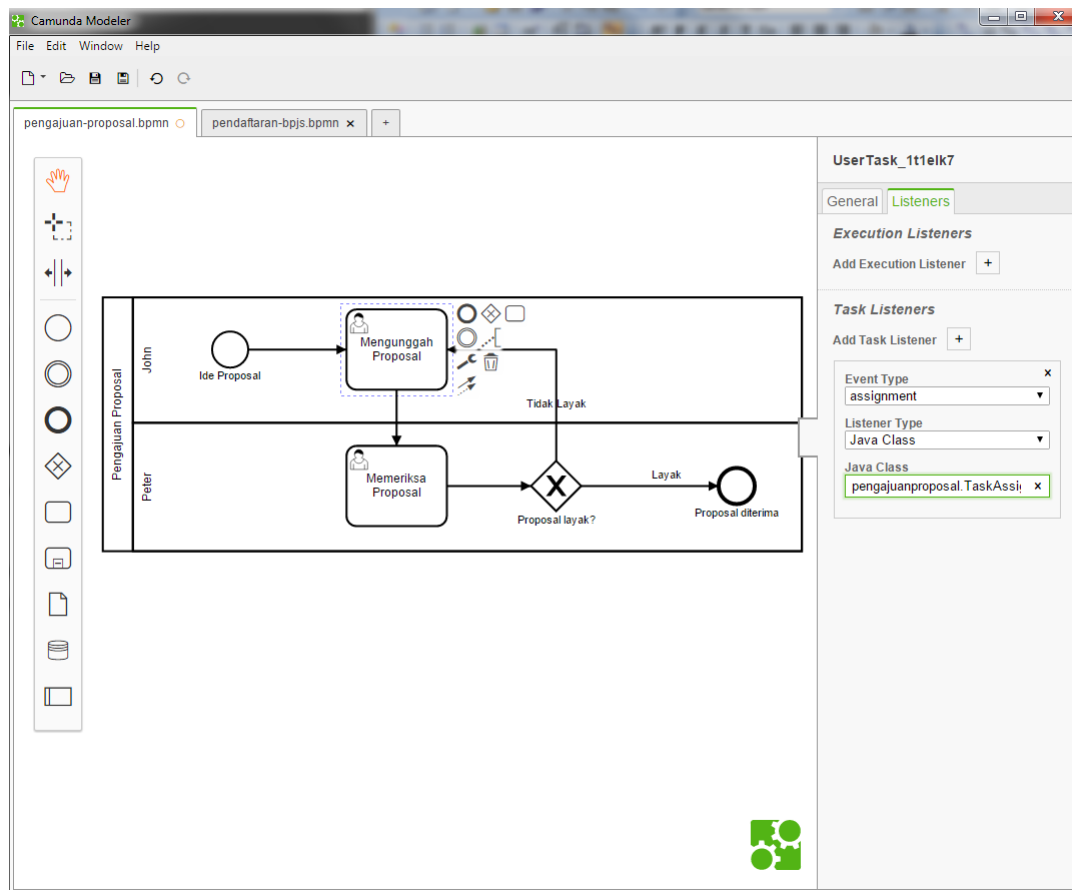
1|
2|
3| <html>
4| <head></head>
5|
6| <body>
7| <form role="form" name="form">
8|     <a cam-file-download="proposal">Download Dokumen</a>
9|     <p>Apakah Proposal layak?</p>
10|     <input cam-variable-name="valid"
11|         cam-variable-type="Boolean"
12|         type="checkbox"
13|         name="valid"
14|         class="form-control" />
15| </form>
16| </body>
17| </html>

```

## 5.3 Pengujian

### 5.3.1 Pengujian Skenario 1

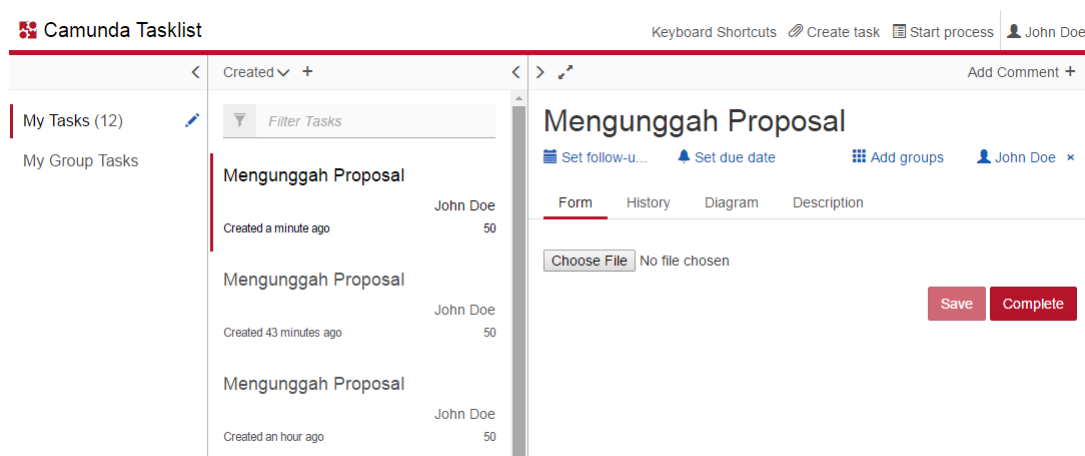
labelujiskenario1 Pada Skenario ??, ditambahkan form HTML dan Task Listener untuk mengirimkan email.



Gambar 5.1: Mengunggah Proposal

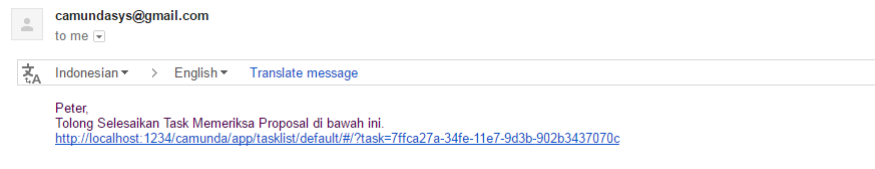
Proses pengujian :

- John mengunggah dokumen proposal



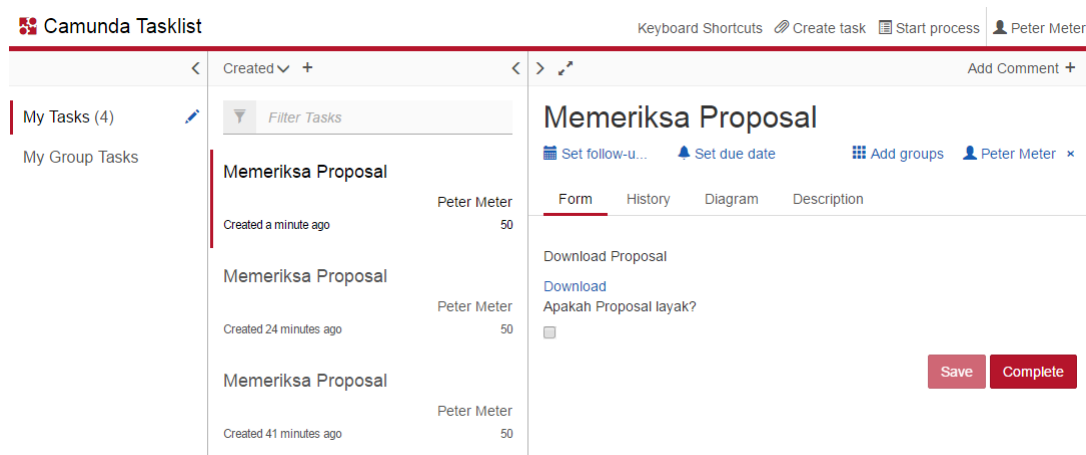
Gambar 5.2: Mengunggah Proposal

- Peter mendapatkan email dari sistem Camunda yang memberitahukan *task* terbaru



Gambar 5.3: Menerima Email

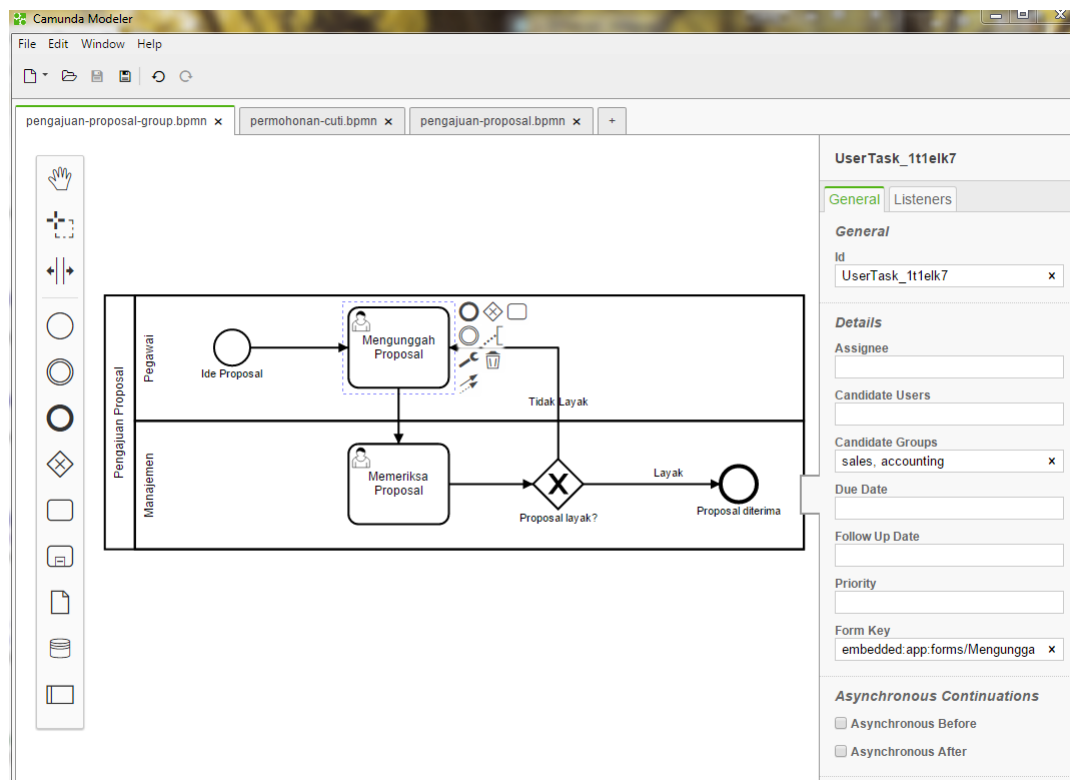
- Tautan email membawa Peter ke *task* yang harus dikerjakan.



Gambar 5.4: Tasklist Peter

### 5.3.2 Pengujian Skenario 2

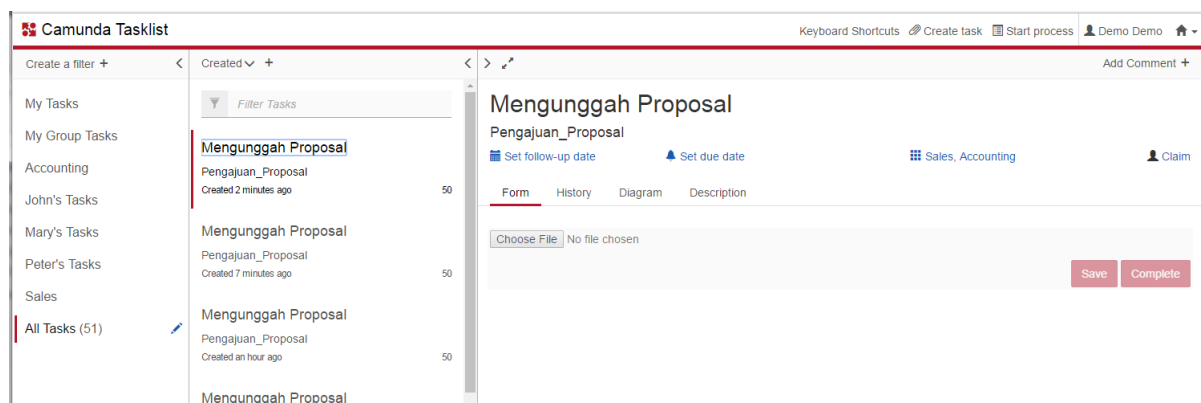
Pada Skenario ??, ditambahkan form HTML dan Task Listener untuk mengirimkan email. Perbedaan dengan Skenario ?? adalah *task* didelegasikan ke grup *accounting*, *sales*, dan *management*. John adalah bagian dari divisi *sales*, Mary adalah bagian dari divisi *accounting*, sedangkan Peter adalah bagian dari divisi *management*.



Gambar 5.5: Mengunggah Proposal Group

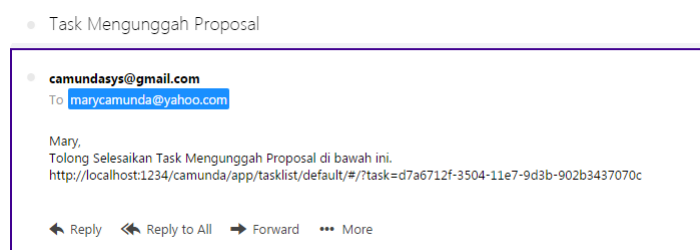
Proses pengujian :

- Admin memulai proses

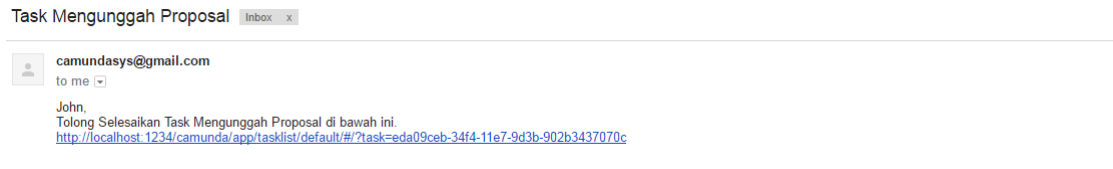


Gambar 5.6: John Mengunggah Proposal

- John dan Mary mendapatkan email untuk mengerjakan *task*.

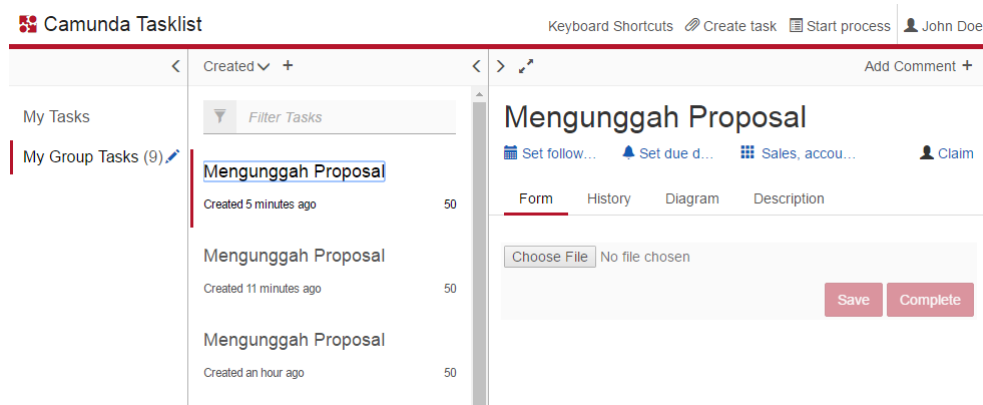


Gambar 5.7: Mary Mendapat Email

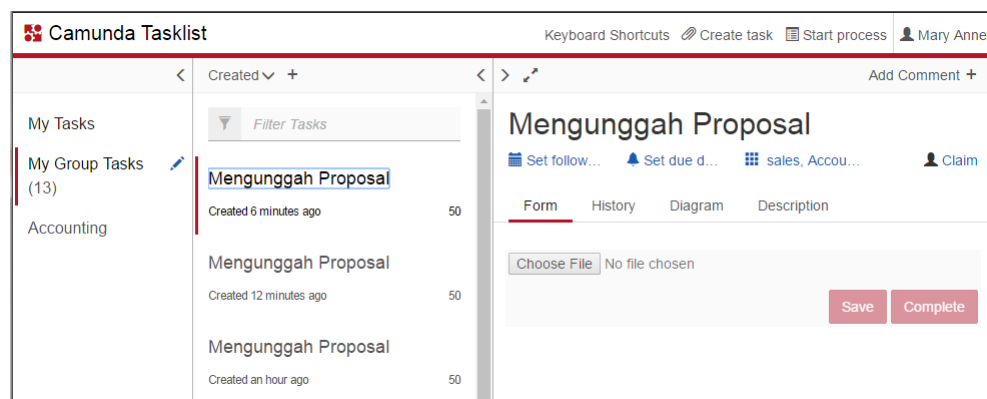


Gambar 5.8: John Mendapat Email

- John dan Mary dapat mengklaim *task*. Apabila John mengklaim *task*, maka Mary tidak bisa mengklaim *task*.

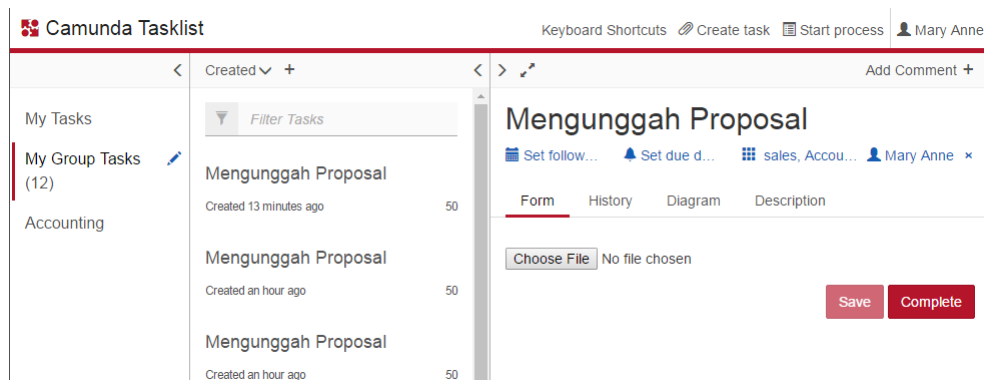


Gambar 5.9: John Mengunggah Proposal



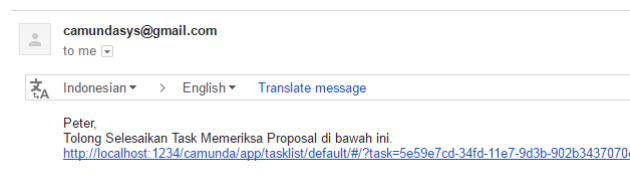
Gambar 5.10: Mary Mengunggah Proposal

- Mary mengklaim task dan mengerjakan *task*



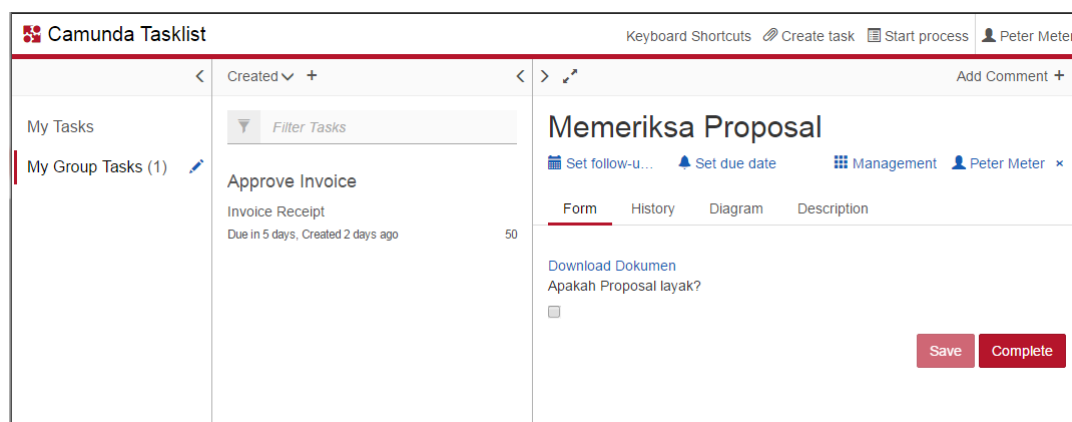
Gambar 5.11: Mary mengklaim Task

- Peter mendapatkan email untuk mengerjakan *task*.



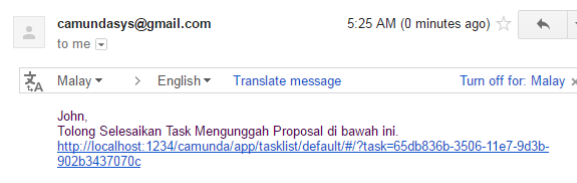
Gambar 5.12: Mary mengklaim Task

- Peter mengklaim task dan menolak proposal



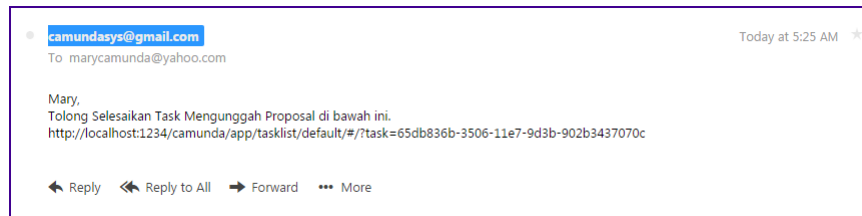
Gambar 5.13: Peter mengklaim Task

- John dan Mary mendapatkan email sistem Camunda yang memberitahukan *task* terbaru



Gambar 5.14: John mendapatkan Email





Gambar 5.15: Mary mendapatkan Email

## 5.4 Hasil Pengujian

Pengujian sudah berhasil untuk semua skenario. Pada skenario 1, John dan Peter masing-masing mendapatkan email ketika *task* siap dikerjakan. Pada skenario 2, Setiap karyawan (John dan Mary) mendapatkan email untuk membuat proposal. Peter juga mendapat email setelah John atau Mary mengunggah proposal.



## BAB 6

### KESIMPULAN DAN SARAN

Pada bab enam ini akan dijelaskan mengenai kesimpulan dan saran yang didapat dari propagasi sistem email dengan Camunda

#### 6.1 Kesimpulan

Berdasarkan hasil pengembangan propagasi sistem email dengan Camunda, didapatkan beberapa kesimpulan sebagai berikut :

1. *Workflow* dapat dimodelkan sebagai BPMN yang dapat divisualisasikan oleh BPMS.
2. *Event-event* dapat dipropagasi via email sehingga aktor dapat mengetahui apabila ada *task* yang harus dikerjakan. Dengan demikian akan meningkatkan efektifitas dan efisiensi proses bisnis.
3. Propagasi email dapat dilakukan dengan cara menyisipkan *Task Listener* di event yang akan dipropagasi. Selain itu dibutuhkan peran admin untuk mendaftarkan alamat email aktor.
4. Pengujian telah dilakukan dengan dua skenario dan dapat berjalan dengan baik.

#### 6.2 Saran

Berdasarkan kesimpulan yang didapat, ada beberapa saran untuk penelitian dan pengembangan lebih lanjut, antara lain :

1. Mekanisme propagasi email dapat dibuat dalam bentuk *library* sehingga tidak perlu membuka kode dan cukup memasukkan alamat email dan password.
2. Aspek integrasi bisa ditambahkan dengan *external tasks*, yaitu sistem di luar Camunda dengan memanfaatkan *web service*.



## DAFTAR REFERENSI

- [1] Dumas, M., Rosa, M. L., Mendling, J., dan Reijers, H. A. (2013) *Fundamentals of Business Process Management*. Springer-Verlag, Berlin.
- [2] Camunda (2015) Bpmn modeling reference. <https://camunda.org/bpmn/reference/>.
- [3] Version 7.6 (2015) *The Camunda BPM Manual*. Camunda BPM. Berlin, Germany.
- [4] Oracle Javamail. <http://www.oracle.com/technetwork/java/javamail/index.html/>.
- [5] Camunda (2015) Get started with camunda and bpmn 2.0. <https://docs.camunda.org/get-started/bpmn20/>.
- [6] Google Use smtp settings to send mail from a printer, scanner, or app. <https://support.google.com/a/answer/176600?hl=en>.



# LAMPIRAN A

## KODE PROGRAM PENGIRIMAN EMAIL

Listing A.1: TaskAssignmentListener.java

```

1
2 package pengajuanproposal;
3
4
5 import java.util.List;
6 import java.util.Properties;
7 import java.util.Set;
8 import java.util.logging.Level;
9 import java.util.logging.Logger;
10
11 import javax.mail.Address;
12 import javax.mail.Message;
13 import javax.mail.MessagingException;
14 import javax.mail.NoSuchProviderException;
15 import javax.mail.Session;
16 import javax.mail.Transport;
17 import javax.mail.internet.MimeMessage;
18 import javax.mail.internet.InternetAddress;
19
20 import org.camunda.bpm.engine.IdentityService;
21 import org.camunda.bpm.engine.delegate.DelegateTask;
22 import org.camunda.bpm.engine.delegate.TaskListener;
23 import org.camunda.bpm.engine.identity.User;
24 import org.camunda.bpm.engine.impl.context.Context;
25 import org.camunda.bpm.engine.impl.persistence.entity.IdentityLinkEntity;
26 import org.camunda.bpm.engine.impl.persistence.entity.TaskEntity;
27 import org.camunda.bpm.engine.task.IdentityLinkType;
28
29
30 public class TaskAssignmentListener implements TaskListener {
31     private static final String HOST = "smtp.gmail.com";
32     private static final String USER = "camundasys@gmail.com";
33     private static final String PWD = "epW3S4KN";
34
35     String assignee;
36     String taskId;
37     String taskName;
38
39     String[] recipient;
40
41     static Properties props;
42     static Session session;
43     static MimeMessage message;
44
45
46     public void notify(DelegateTask delegateTask) {
47         assignee = delegateTask.getAssignee();
48         taskId = delegateTask.getId();
49         taskName = delegateTask.getName();
50         delegateTask.getCandidates();
51
52         if (assignee != null) {
53             IdentityService identityService = Context.getProcessEngineConfiguration().getIdentityService();
54             User user = identityService.createUserQuery().userId(assignee).singleResult();
55             if (user != null) {
56                 this.sendEmail(user);
57             }
58         }
59         else {
60             TaskEntity task = (TaskEntity) delegateTask;
61             List<IdentityLinkEntity> identityLinks = task.getIdentityLinks();
62
63             for (IdentityLinkEntity link : identityLinks) {

```

```

64         if(link.getType().equals(IdentityLinkType.CANDIDATE)) {
65             if(link.isUser()) {
66                 User user = Context.getProcessEngineConfiguration().getIdentityService().
67                     createUserQuery().userId(link.getUserId()).singleResult();
68                 sendEmail(user);
69             }
70             if(link.isGroup()) {
71                 List<User> users = Context.getProcessEngineConfiguration().getIdentityService
72                     ().createUserQuery().memberOfGroup(link.getGroupId()).list();
73                 for(User user : users) {
74                     sendEmail(user);
75                 }
76             }
77         }
78     }
79
80     public void sendEmail(User user){
81         try {
82             props = System.getProperties();
83             props.put("mail.smtp.port", "587");
84             props.put("mail.smtp.auth", "true");
85             props.put("mail.smtp.starttls.enable", "true");
86
87             session = Session.getDefaultInstance(props, null);
88             message = new MimeMessage(session);
89             message.addRecipient(Message.RecipientType.TO, new InternetAddress(user.getEmail()));
90             message.setSubject("Task" + taskName);
91
92             String emailBody = user.getFirstName() + "<br>";
93             emailBody += "Tolong Selesaikan Task" + taskName + " dibawah ini.<br>";
94             emailBody += "http://localhost:1234/camunda/app/tasklist/default/#/?task="+taskId;
95             message.setContent(emailBody, "text/html");
96
97             Transport transport = session.getTransport("smtp");
98             transport.connect(HOST, USER, PWD);
99             transport.sendMessage(message, message.getAllRecipients());
100            transport.close();
101        } catch (NoSuchProviderException ex) {
102            Logger.getLogger(TaskAssignmentListener.class.getName()).log(Level.SEVERE, null, ex);
103        } catch (MessagingException ex) {
104            Logger.getLogger(TaskAssignmentListener.class.getName()).log(Level.SEVERE, null, ex);
105        }
106    }
107 }
108 }
109 }
110 }

```



## LAMPIRAN B

### KODE POM.XML

Listing B.1: pom.xml

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance"
2     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven
   -4.0.0.xsd">
3     <modelVersion>4.0.0</modelVersion>
4     <groupId>org.camunda.bpm.getstarted</groupId>
5     <artifactId>loan-approval</artifactId>
6     <version>0.1.0-SNAPSHOT</version>
7     <packaging>war</packaging>
8
9     <dependencyManagement>
10        <dependencies>
11            <dependency>
12                <groupId>org.camunda.bpm</groupId>
13                <artifactId>camunda-bom</artifactId>
14                <version>7.6.0</version>
15                <scope>import</scope>
16                <type>pom</type>
17            </dependency>
18        </dependencies>
19    </dependencyManagement>
20
21    <dependencies>
22        <dependency>
23            <groupId>org.camunda.bpm</groupId>
24            <artifactId>camunda-engine</artifactId>
25            <scope>provided</scope>
26        </dependency>
27
28        <dependency>
29            <groupId>javax.servlet</groupId>
30            <artifactId>javax.servlet-api</artifactId>
31            <version>3.0.1</version>
32            <scope>provided</scope>
33        </dependency>
34    </dependencies>
35
36    <build>
37        <plugins>
38            <plugin>
39                <groupId>org.apache.maven.plugins</groupId>
40                <artifactId>maven-war-plugin</artifactId>
41                <version>2.3</version>
42                <configuration>
43                    <failOnMissingWebXml>>false</failOnMissingWebXml>
44                </configuration>
45            </plugin>
46        </plugins>
47    </build>
48
49 </project>
```