

**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 perangkat lunak untuk otomatisasi proses bisnis, yaitu BPMS (*Business Process Management System*), seperti Camunda. Agar eksekusi *workflow* lebih alami dengan model komunikasi organisasi saat ini, maka *event* dapat dipropagasi dan diintegrasikan dengan sistem email.

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 email akan mengirim email ke pengguna yang akan mengerjakan task tersebut. Email tersebut berisi tautan yang mengarah ke tugas yang perlu dikerjakan tersebut.

Berdasarkan pengujian, sistem dapat mengirim email ke masing-masing pemilik *task*. Email langsung dikirim setelah *user task* siap untuk dikerjakan. Dengan ini dapat disimpulkan proses integrasi *user task* dan sistem email dapat dilakukan.

**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.

According to experiment, system can send email to task owner. System send the email instantly after task ready. With this, we can conclude that user task can be integrated with email system.

**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>xiii</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 . . . . .	14
2.7 Email . . . . .	15
2.7.1 <i>Mail Server</i> . . . . .	15
2.7.2 <i>JavaMail</i> . . . . .	16
<b>3 HASIL STUDI</b>	<b>17</b>
3.1 Hasil Studi BPMN . . . . .	17
3.1.1 <i>Masalah Proses Bisnis</i> . . . . .	17
3.1.2 <i>Memodelkan Workflow</i> . . . . .	18
3.2 Menyiapkan BPMS Camunda . . . . .	22
3.2.1 <i>Instalasi Camunda</i> . . . . .	22
3.2.2 <i>Kasus 1 - Pengajuan Proposal</i> . . . . .	23
3.2.3 <i>Kasus 2 - Pendaftaran BPJS</i> . . . . .	26
3.3 Menjalankan Camunda . . . . .	28

3.3.1	Otomasi Kasus 1 - Pengajuan Proposal . . . . .	28
3.3.2	Otomasi Kasus 2 - Pendaftaran BPJS . . . . .	29
<b>4</b>	<b>ANALISIS DAN PERANCANGAN</b>	<b>33</b>
4.1	Analisis Hasil Studi . . . . .	33
4.1.1	<i>Event</i> yang Terkait dengan Integrasi Sistem Email . . . . .	33
4.1.2	Mekanisme Integrasi Sistem Email . . . . .	33
4.2	Analisis Kebutuhan . . . . .	34
4.3	Perancangan Sistem . . . . .	35
4.3.1	Perancangan Aktor . . . . .	35
4.3.2	Perancangan Email . . . . .	35
4.3.3	Algoritma Pengiriman Email . . . . .	36
4.3.4	<i>Task Event Listener</i> . . . . .	36
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>37</b>
5.1	Lingkungan Implementasi . . . . .	37
5.2	Implementasi Algoritma Pengiriman Email . . . . .	37
5.3	Pengujian . . . . .	39
5.3.1	Pengujian Kasus Pengajuan Proposal . . . . .	39
5.3.2	Pengujian Kasus Pendaftaran BPJS . . . . .	42
5.4	Hasil Pengujian . . . . .	49
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>51</b>
6.1	Kesimpulan . . . . .	51
6.2	Saran . . . . .	51
	<b>DAFTAR REFERENSI</b>	<b>53</b>
	<b>A KODE PROGRAM PENGIRIMAN EMAIL</b>	<b>55</b>
	<b>B KODE POM.XML</b>	<b>57</b>
	<b>C KODE SKENARIO</b>	<b>59</b>
C.1	Kasus 1 - Pengajuan Proposal . . . . .	59
C.2	Kasus 2 - Pendaftaran BPJS . . . . .	60

## 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	Tampilan Pengaturan Camunda Modeler	12
2.13	Tampilan Pengaturan Camunda Modeler	12
2.14	Tampilan Pengaturan Camunda Modeler	12
2.15	Camunda Tasklist	13
2.16	Camunda Cockpit	13
2.17	Camunda Admin	14
3.1	Mengunggah Proposal	18
3.2	Mengunggah Proposal	19
3.3	Memeriksa Proposal	19
3.4	Pendaftaran BPJS	21
3.5	Atribut <i>assignee</i> dari Mengisi formulir pendaftaran BPJS	22
3.6	Mengunggah Proposal	24
3.7	Memeriksa Proposal	24
3.8	Proposal Layak	25
3.9	Ekspresi Proposal Layak	25
3.10	Proposal tidak Layak	26
3.11	Ekspresi Proposal tidak Layak	26
3.12	Menghubungkan <i>Service Task</i> dengan kode Java	28
3.13	Mengunggah Proposal	29
3.14	Memeriksa Proposal	29
3.15	Melihat Status Proposal	29
3.16	Mendaftar BPJS	30
3.17	Mengunggah Dokumen	30
3.18	Melihat Nomor dan Biaya Pendaftaran	31
3.19	Memilih Hari	31
3.20	Mencetak Jadwal	31
3.21	Verifikasi Pendaftaran BPJS	32
3.22	Mencetak Kartu BPJS	32
4.1	Event Task Listener	33

5.1	Task Listener pada BPMN . . . . .	39
5.2	Memulai Proses Pengajuan Proposal . . . . .	39
5.3	Email Mengunggah Proposal . . . . .	40
5.4	Email Mengunggah Proposal . . . . .	40
5.5	Mengunggah Proposal . . . . .	40
5.6	Email Memeriksa Proposal . . . . .	41
5.7	Peter Memeriksa Proposal . . . . .	41
5.8	Email Melihat Status Proposal . . . . .	41
5.9	John Melihat Status Proposal . . . . .	42
5.10	Memulai Proses Pendaftaran BPJS . . . . .	42
5.11	Email Mengisi Formulir Pendaftaran BPJS . . . . .	43
5.12	Mengisi Formulir Pendaftaran BPJS . . . . .	43
5.13	Email Mengunggah Dokumen Persyaratan . . . . .	44
5.14	Mengunggah Dokumen Persyaratan . . . . .	44
5.15	Email Nomor Pembayaran dan Uang Pendaftaran . . . . .	45
5.16	Melihat Nomor Pembayaran dan Uang Pendaftaran . . . . .	45
5.17	Email Memilih Jadwal Verifikasi Dokumen . . . . .	45
5.18	Memilih Jadwal Verifikasi Dokumen . . . . .	46
5.19	Email Mencetak Jadwal . . . . .	46
5.20	Mencetak Jadwal dan Nomor Antrian . . . . .	47
5.21	Email Verifikasi Pendaftaran . . . . .	47
5.22	Memverifikasi Pendaftaran dan Semua Persyaratan . . . . .	48
5.23	Email Mencetak Kartu BPJS . . . . .	48
5.24	Mencetak Kartu BPJS . . . . .	49

## 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, yaitu satu kasus sederhana dan satu kasus kompleks.

### 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)*, BPMS Camunda, Forms SDK dan sistem e-mail.
3. Bab 3 Hasil Studi, berisi masalah proses bisnis yang diselesaikan menggunakan otomatisasi BPMS Camunda. Mulai dari memodelkan *workflow*, instalasi Camunda, menghubungkan BPMN dan BPMS Camunda hingga otomatisasi menggunakan BPMS Camunda.
4. Bab 3 Analisis, Berisi analisis BPMN dengan menggunakan skenario, analisis event yang terkait dengan sistem email dan mekanisme integrasi sistem email.
5. Bab 4 Analisis dan Perancangan, berisi analisis hasil studi mengenai *event* yang terkait dengan integrasi sistem email beserta mekanisme integrasinya, analisis kebutuhan, dan rancangan sistem yang berupa rancangan alamat email dan algoritma pengiriman email.
6. Bab 5 Implementasi, dan Pengujian Berisi implementasi dari program yang dibuat dan pengujian aplikasi berdasarkan contoh kasus pada bab tiga.
7. 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.

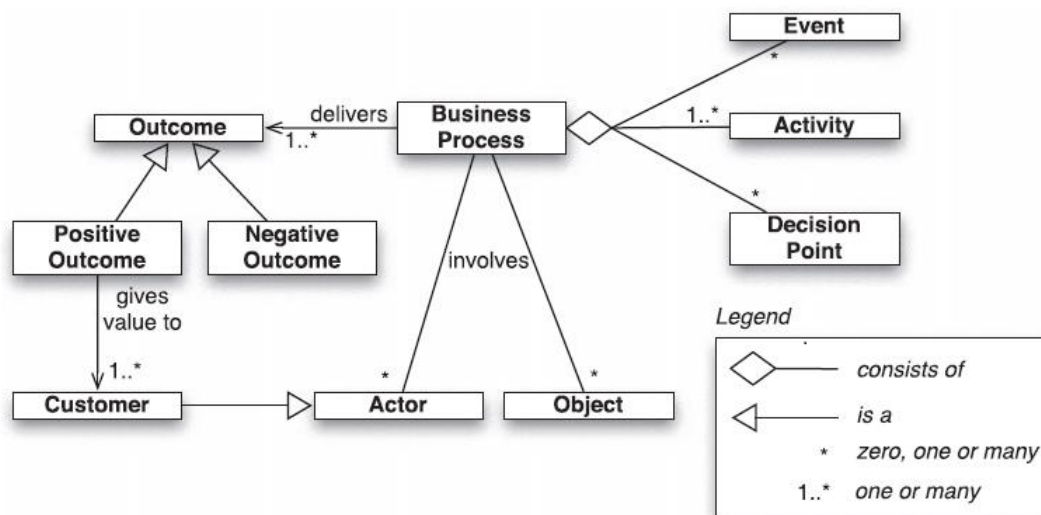
###### *Object*

*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*.

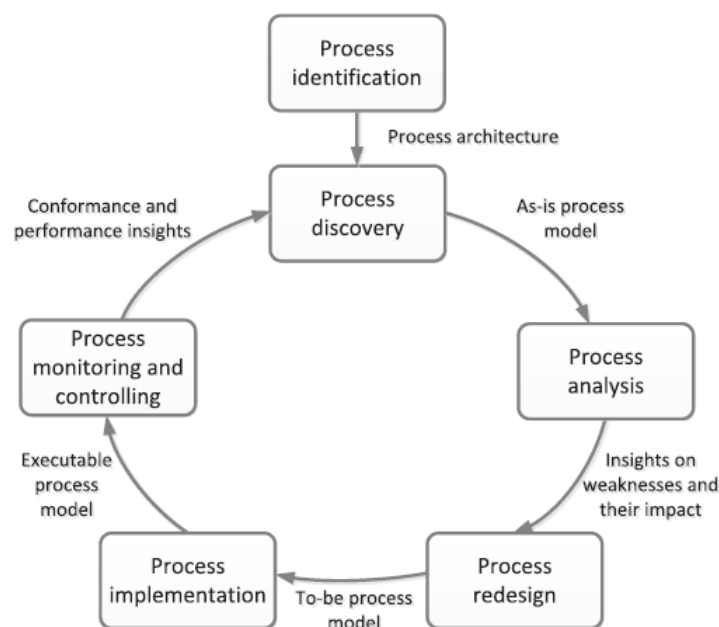
### Process Implementation

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.

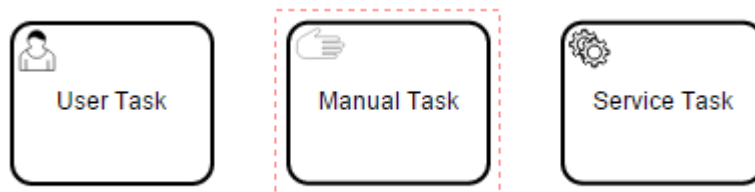


Gambar 2.3: Notasi *Event*

### 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

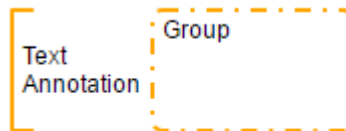
*Data Object* melambangkan informasi yang berjalan dalam proses seperti dokumen, e-mail, atau surat. Sedangkan *Data Store* merupakan tempat proses membaca atau menyimpan data seperti basis data atau rak.



Gambar 2.6: Notasi *Data*

### 2.3.5 Artifact

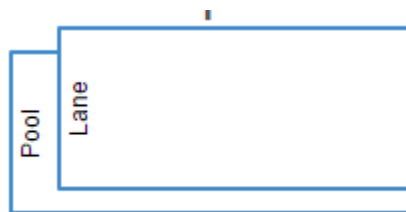
*Artifact* tidak mempengaruhi jalannya proses, tetapi hanya sebagai informasi tambahan agar proses lebih mudah dimengerti. Terdapat dua jenis, yaitu *Text Annotation* dan *Group*



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*

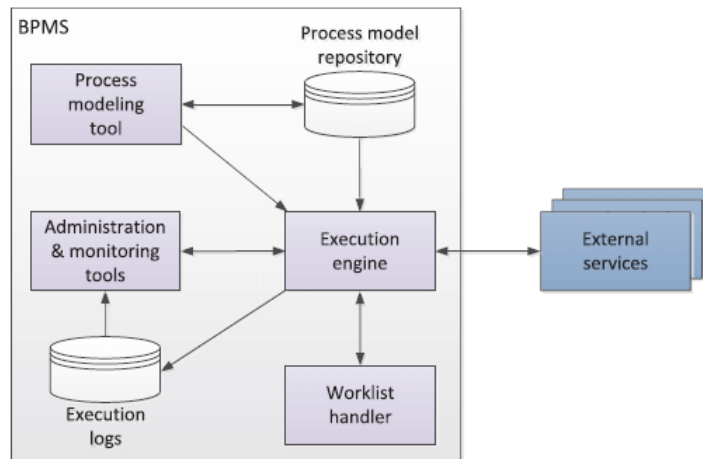
## 2.4 Business Process Management System (BPMS)

*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.



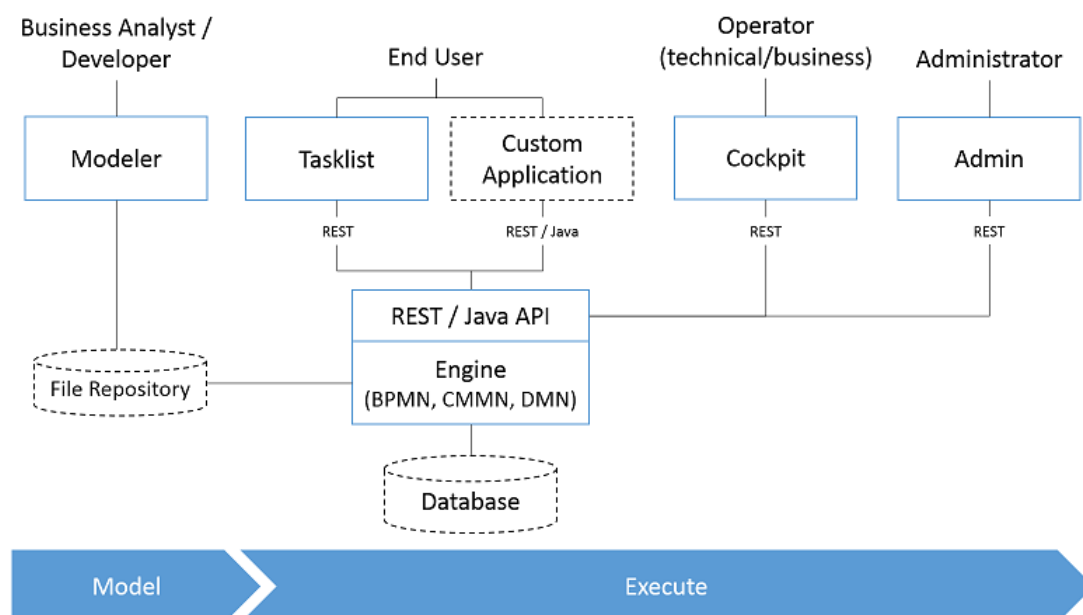
Gambar 2.9: Arsitektur BPMS

## 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

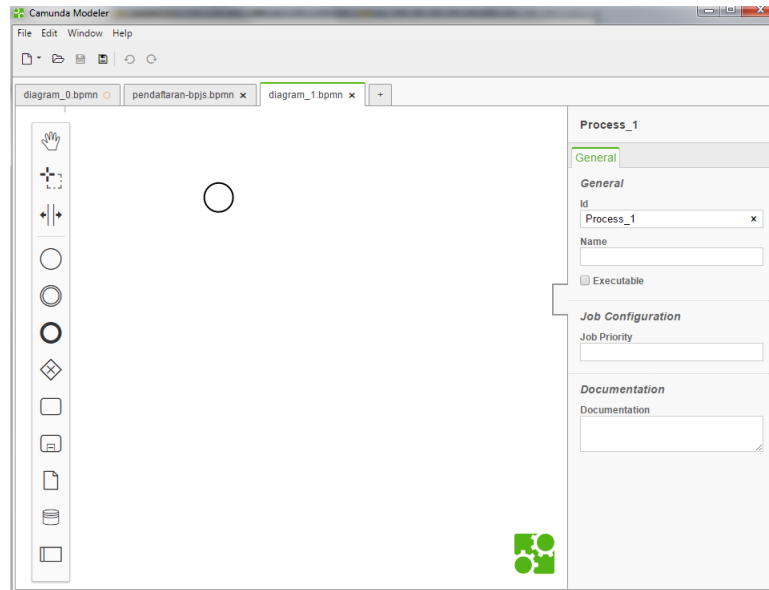
Komponen-komponen pada BPMS Camunda adalah sebagai berikut :



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 tengah merupakan tempat membuat diagram BPMN.
3. Bagian kanan merupakan pengaturan untuk tiap *event*, *task*, maupun notasi lainnya. Berikut ini adalah penjelasan untuk beberapa kolom pengaturan *user task* :

(a) Tab General

- Id, yaitu id dari *task*
- Name, yaitu nama dari *task*
- Assignee, yaitu nama pemilik *task*
- Candidate Users, yaitu kandidat pemilik dari *task*
- Candidate Group, yaitu kandidat grup dari *task*
- Due Date, yaitu pengaturan batas waktu
- Follow Up Date, yaitu pengaturan waktu *follow up*

**UserTask\_1t1elk7**

General Forms Listeners Input/Output Extensions

**General**

Id  
UserTask\_1t1elk7 x

Name  
Mengunggah Proposal

**Details**

Assignee

Candidate Users

Candidate Groups  
accounting, sales x

Due Date

The due date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

Follow Up Date

The follow up date as an EL expression (e.g. \${someDate}) or an ISO date (e.g. 2015-06-26T09:54:00)

Priority

Gambar 2.12: Tampilan Pengaturan Camunda Modeler

- (b) Tab Forms, tempat untuk mengisi lokasi halaman web yang diacu oleh *user task*

**UserTask\_1t1elk7**

General Forms Listeners Input/Output Extensions

**Forms**

Form Key  
embedded:app:forms/MengunggahProposal.html x

Form Fields x +

Gambar 2.13: Tampilan Pengaturan Camunda Modeler

- (c) Listener, terbagi menjadi dua, yaitu *Execution Listener* dan *Task Listener*. Listener berfungsi untuk menjalankan perintah ketika ada suatu *event* yang terjadi. Misalnya *Task Listener* dengan event *create* akan dijalankan ketika *task* dibuat.

**UserTask\_1t1elk7**

General Forms Listeners Input/Output Extensions

**Listeners**

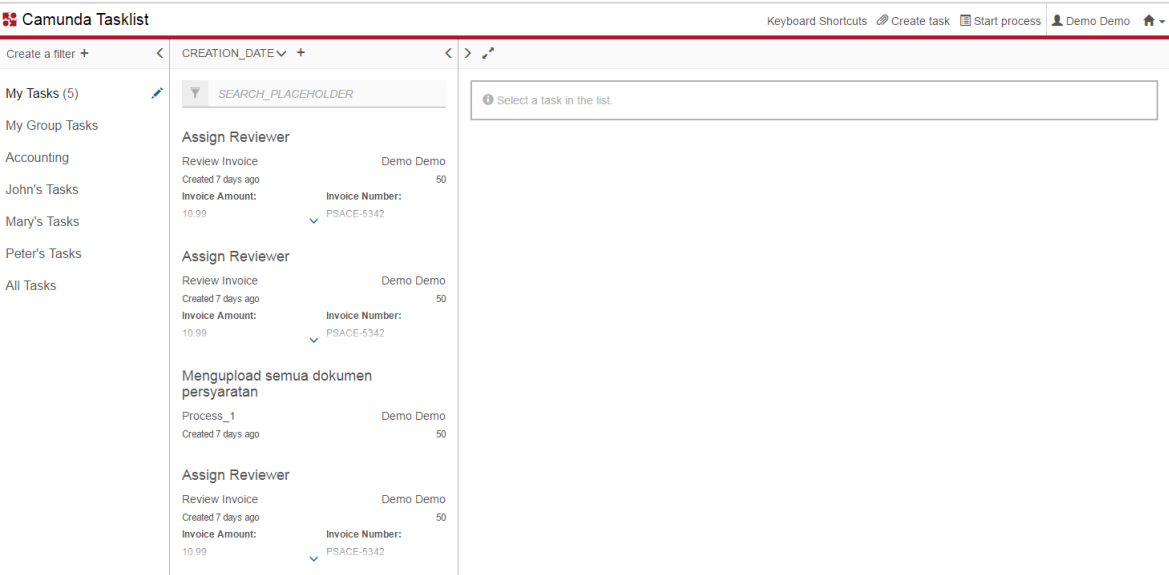
Execution Listener x +

Task Listener x +

create : Java Class

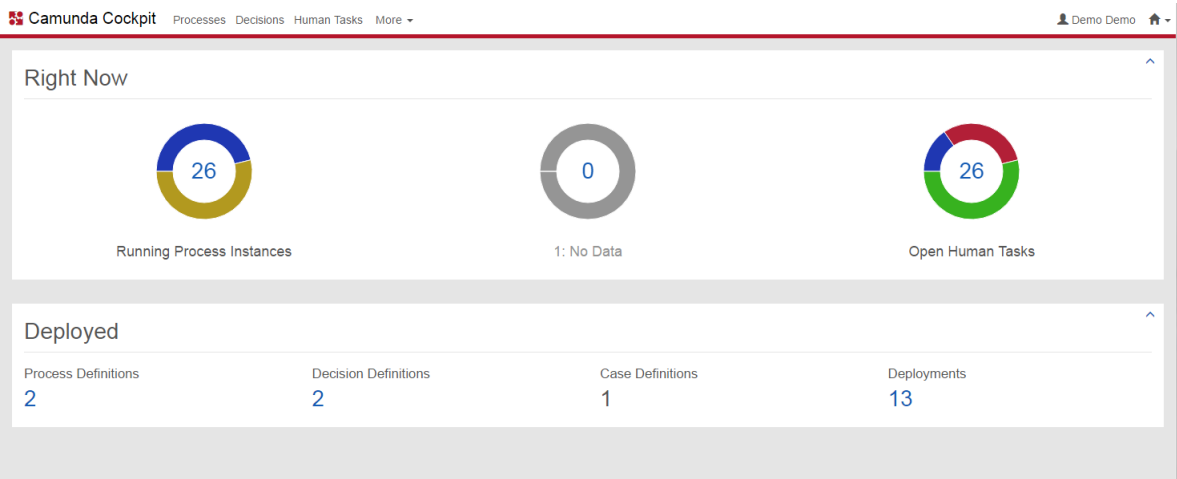
Gambar 2.14: Tampilan Pengaturan Camunda Modeler

- *Tasklist*, tempat pengguna mengakses dan mengerjakan tugas. Tugas yang dikerjakan mengikuti alur model proses (BPMN) yang telah dibuat.



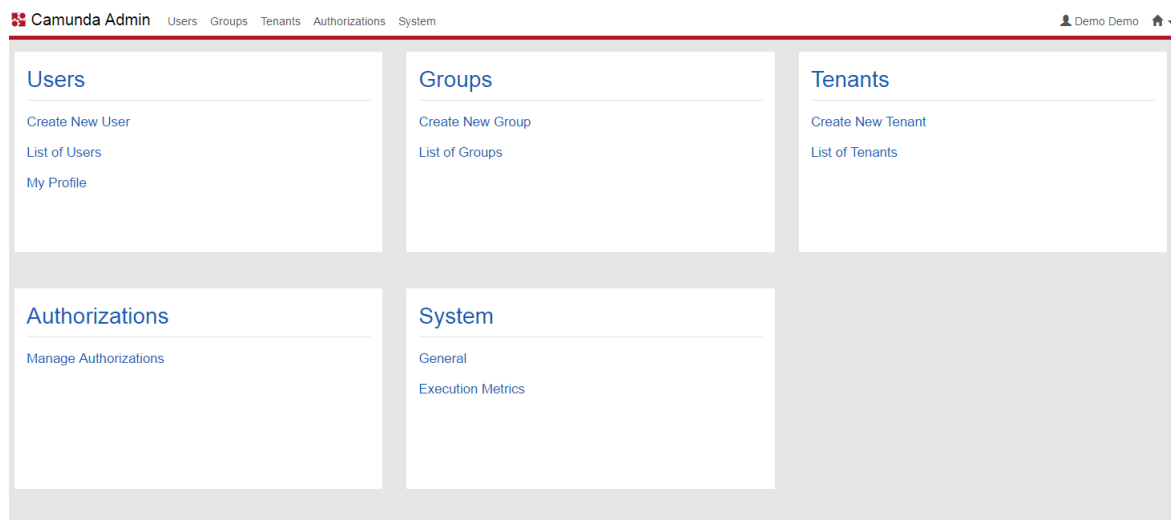
Gambar 2.15: Camunda Tasklist

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



Gambar 2.16: Camunda Cockpit

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



Gambar 2.17: 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. 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="10000000" />

```

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 Email

### 2.7.1 Mail Server

*Mail server* adalah mekanisme pengiriman email yang menyediakan berbagai standar sehingga email bisa dikirim dari satu domain ke domain lainnya. *Mail server* dapat dikategorikan menjadi dua, yaitu :

1. *Mail server* yang menuju ke dalam, yaitu server SMTP *Simple Mail Transfer Protocol*. Protokol SMTP digunakan untuk mengirim email dari sebuah klien ke alamat yang dituju.
2. *Mail server* yang menuju ke luar, yaitu server POP3 *Post Office Protocol* dan server IMAP *Internet Message Access Protocol*.

### 2.7.2 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 Enterprise Edition (Java EE) yang merupakan lingkungan komputasi untuk pengembangan perangkat lunak skala besar dan menggunakan jaringan yang aman.

Untuk menggunakan JavaMail diperlukan beberapa kelas, yaitu :

- Kelas Session, kelas untuk mengumpulkan atribut email yang akan digunakan. Kelas Session mengambil berbagai atribut dari kelas Properties, seperti informasi server email yang akan digunakan.
- Kelas MimeMessage, kelas untuk menulis email. Kelas ini menyediakan berbagai atribut untuk menulis email. Di antaranya adalah alamat email asal, alamat email penerima, subjek, isi email, dll.
- Kelas Transport, kelas untuk membuat koneksi ke email *server* dan mengirim email.

Di bawah ini adalah contoh kode pengiriman email menggunakan JavaMail API :

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. Apabila proposalnya layak, pegawai dapat melihat setelah status proposal tersebut. 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.
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.

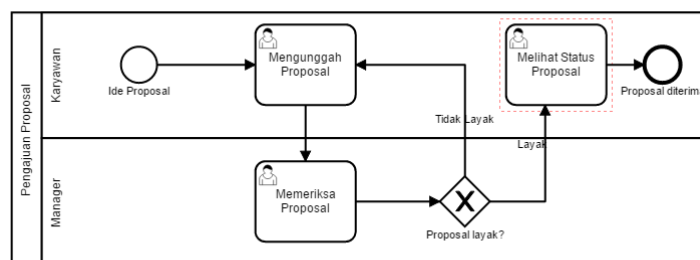
Workflow dari proses bisnis ini adalah sebagai berikut :

### 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. Tiga *user task*, yaitu Mengunggah Proposal oleh pegawai, Memeriksa Proposal oleh manajemen, dan Melihat Status Proposal oleh pegawai.
5. Satu *decision point*, yaitu penentuan apakah proposal layak atau tidak

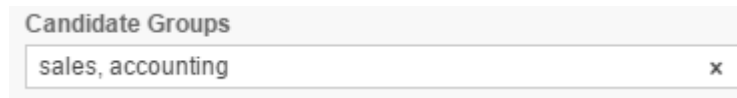


Gambar 3.1: Mengunggah Proposal



*User task* Mengunggah Proposal dan Melihat Status Proposal

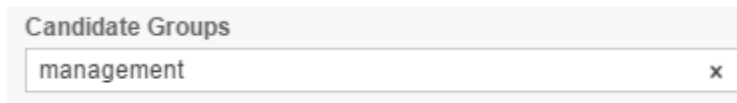
*Task* ini dapat dilakukan oleh pegawai (divisi *sales* dan *accounting*). Maka *Candidate Groups* diisi dengan *sales* dan *accounting*.

A screenshot of a web form element titled "Candidate Groups". It features a text input field with the value "sales, accounting" and a small "x" icon on the right side of the field, likely for clearing the input.

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 element titled "Candidate Groups". It features a text input field with the value "management" and a small "x" icon on the right side of the field, likely for clearing the input.

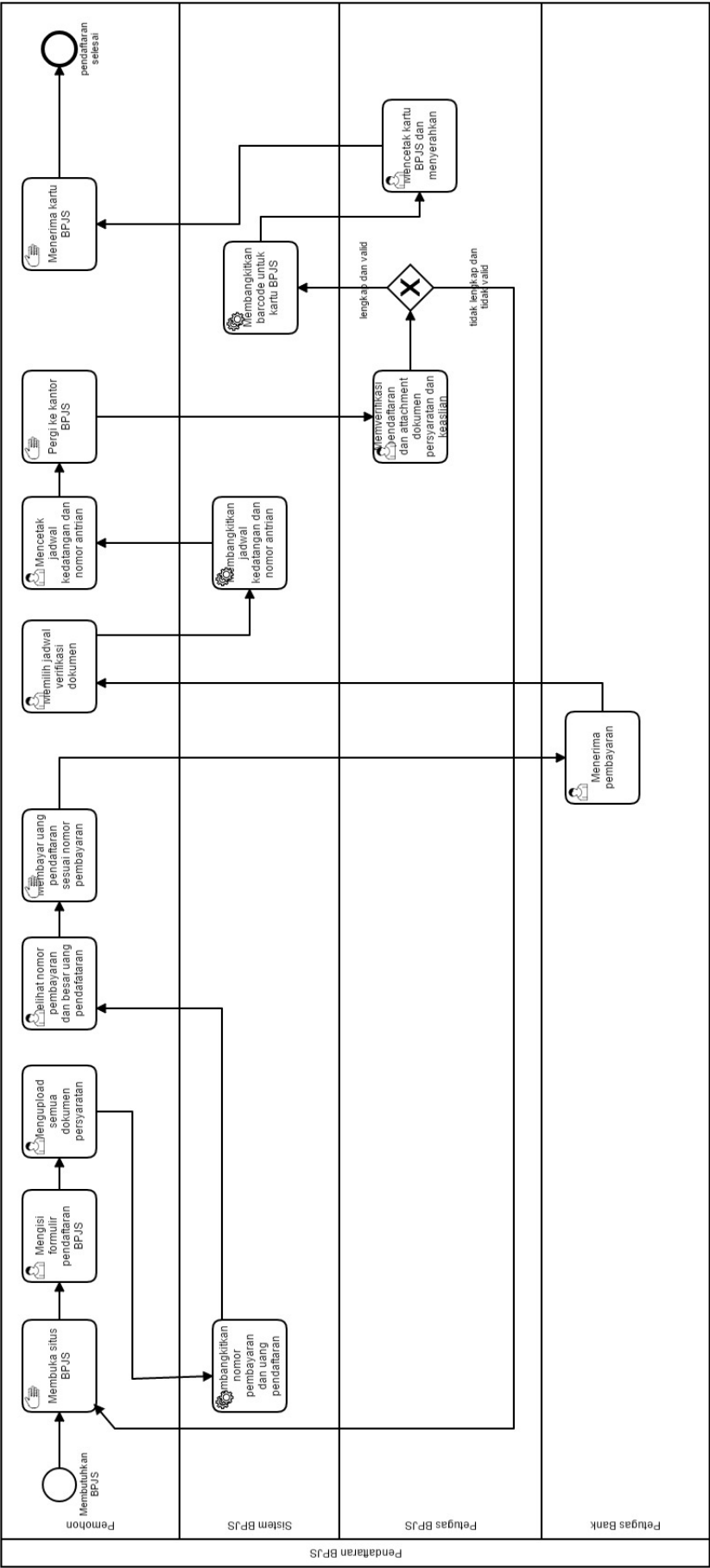
Gambar 3.3: Memeriksa Proposal

### 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 *Manual 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. At the top, the title is 'UserTask\_15dx9zp'. Below the title are five tabs: 'General', 'Forms', 'Listeners', 'Input/Output', and 'Extensions'. The 'General' tab is selected and highlighted. Under the 'General' tab, there are two sections. The first section, 'General', contains two text input fields: 'Id' with the value 'UserTask\_15dx9zp' and 'Name' with the value 'Mengisi formulir pendaftaran BPJS'. The second section, 'Details', contains one text input field: 'Assignee' with the value 'john'. Each input field has a small 'x' icon in the top right corner, likely for clearing the field.

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 menyiapkan proses bisnis pengajuan proposal sehingga dapat dijalankan oleh BPMS Camunda, langkah-langkah yang perlu dilakukan adalah :

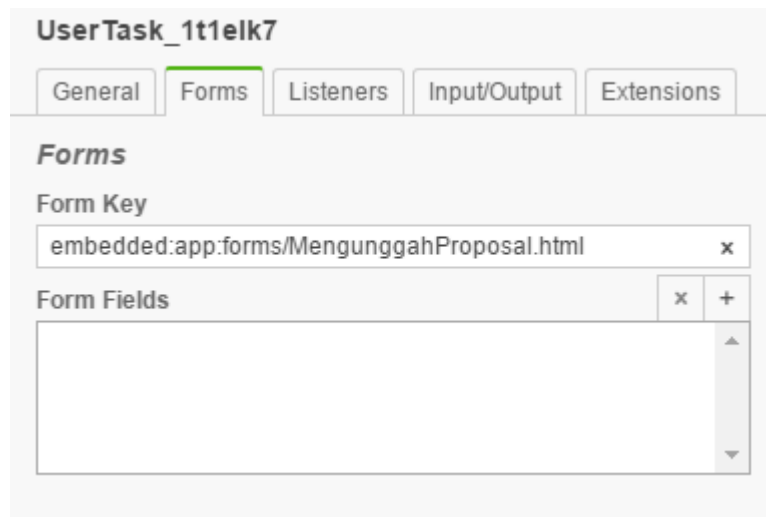
1. Menambah Form HTML untuk setiap *user task* dan menghubungkannya dengan BPMN. File HTML yang dibuat disimpan di direktori src/main/webapp/forms. Proses bisnis ini memiliki dua *user task*, yaitu mengunggah proposal dan memeriksa proposal.
  - (a) *User task* Mengunggah Proposal Task ini merupakan *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

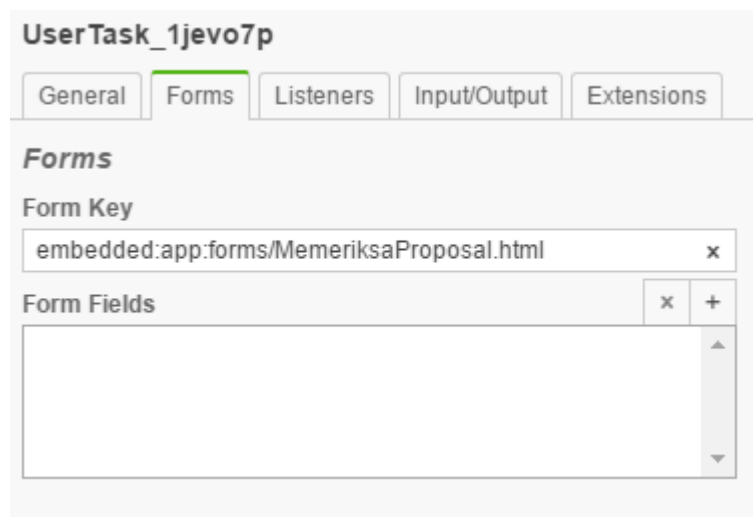
- (b) *User task* Memeriksa Proposal Task ini merupakan *user task* sehingga membutuhkan suatu *form* HTML untuk memeriksa proposal. Isi dari variabel *cam-file-download* adalah "proposal" (sama dengan *cam-variable-name* pada *form* mengunggah proposal, sehingga file yang diunduh sama dengan file yang 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

- (c) *User Task* Melihat Status Proposal, pegawai dapat melihat status proposalnya apabila sudah diterima. Berikut adalah kodenya :

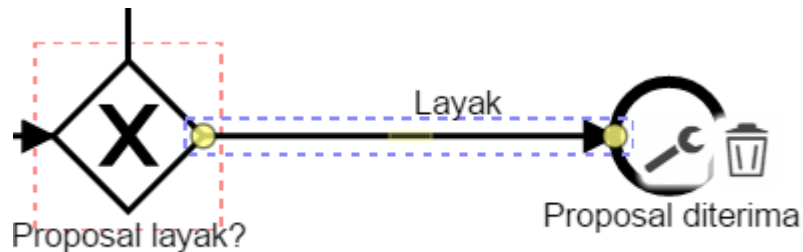
Listing 3.5: MemeriksaProposal.html

```

1 <html>
2   <head></head>
3   <body>
4     <h> Proposal sudah diterima </h>
5
6     <form role="form" name="form">
7       <a cam-file-download="proposal">Lihat Proposal</a>
8     </form>
9
10
11
12 </html>

```

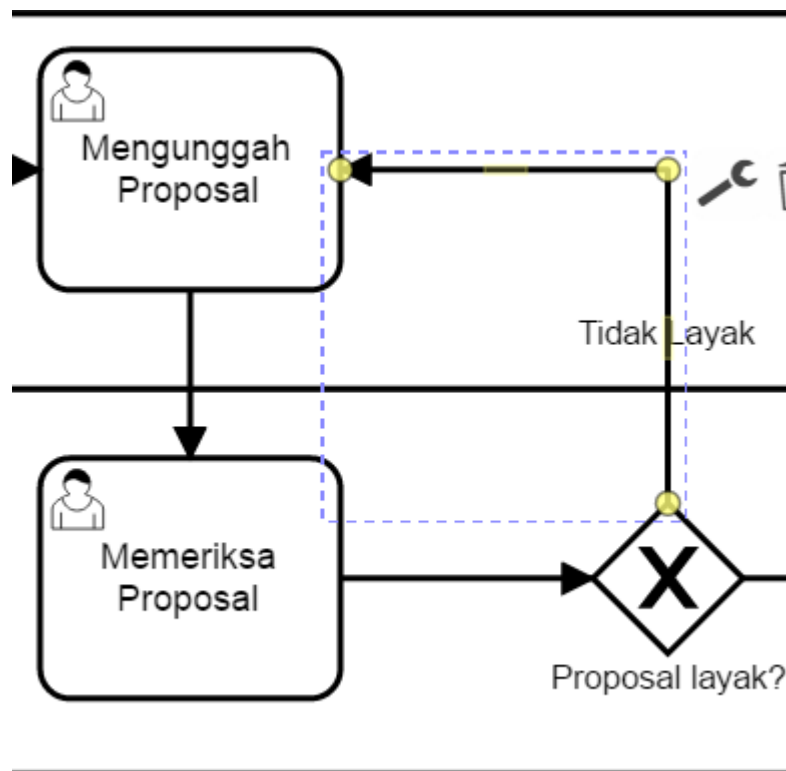
2. Mengatur *Gateway*, untuk mengatur keluaran dari *gateway* pengaturan dapat dilakukan pada modeler dengan menggunakan `cam-variable-name = valid` pada checkbox di form MemeriksaProposal. Apabila proposal layak, maka *expression* yang digunakan adalah `$(valid)`. Jika proposal tidak layak, *expression*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



Gambar 3.8: Proposal Layak

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

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

### 3.2.3 Kasus 2 - Pendaftaran BPJS

Untuk menyiapkan proses bisnis pendaftaran BPJS sehingga dapat dijalankan oleh BPMS Camunda, langkah-langkah yang perlu dilakukan sedikit berbeda dengan proses bisnis pengajuan proposal, yaitu:

1. Menambah Form HTML untuk setiap *user task* dan menghubungkannya dengan BPMN. File HTML yang dibuat disimpan di direktori `src/main/webapp/forms`.
2. Menambah implementasi *Service Task* menggunakan kode Java dan menghubungkannya dengan BPMN.
3. Mengatur keluaran *gateway*.



4. Menyimpan file BPMN ke direktori src/main/resources pada proyek pendaftaran BPJS.

Bagian yang membedakan kasus ini dan kasus pengajuan proposal adalah sebagai berikut :

1. Pengiriman dan penerimaan variabel dari satu form HTML ke form HTML lainnya. Contohnya adalah pemohon mendaftar BPJS pada form pendaftaran-bpjs.html kemudian variabel nama diambil di verifikasi-pendaftaran.html. Isi dari variabel cam-variable-name didaftarkan ke variabelManager (kumpulan variabel) dan diambil dengan fungsi fetchVariable('nama') pada form ringkasan-jadwal.html. Isi dari variabel nama ditempatkan menggunakan \$scope.nama dan <td> nama menggunakan <table>. Berikut adalah potongan kodenya :

Listing 3.6: pilih-jadwal.html

```

1 | <form name="pendaftaranBPJS" role="form">
2 |   <div class="control-group">
3 |     <label class="control-label" for="nama">Nama</label>
4 |     <div class="controls">
5 |       <input id="nama"
6 |         class="form-control"
7 |         cam-variable-name = "nama"
8 |         cam-variable-type = "String"
9 |         type="text"
10 |         required>
11 |     </div>
12 | </form>

```

Listing 3.7: ringkasan-jadwal.html

```

1 | <form role="form" name="form">
2 |   <script cam-script type="text/form-script">
3 |     camForm.on('form-loaded', function() {
4 |       camForm.variableManager.fetchVariable('nama');
5 |     });
6 |     camForm.on('variables-restored', function() {
7 |       \scope.nama = camForm.variableManager.variableValue('nama');
8 |     });
9 |   </script>
10 |
11 |   <table>
12 |     <tr>
13 |       <td>Nama:</td>
14 |       <td>{{ nama }}</td> <!-- Lokasi variabel nama ditampilkan -->
15 |     </tr>
16 |   </table>

```

2. Tiga *Service Task*, yaitu PembangkitBarcode, PembangkitJadwal, dan PembangkitNomor beserta pengiriman dan penerimaan variabel dari *Service Task* ke form HTML maupun sebaliknya. Kelas PembangkitJadwal mengambil data jadwal yang dimasukkan pemohon pada form pilih-jadwal.html menggunakan perintah getVariable("jadwalHari"). Kelas ini juga membangkitkan nomor antrian secara acak. Variabel jadwal dan nomor antrian dikirimkan ke variabelManager menggunakan perintah setVariable(). Berikut adalah potongan kode dari PembangkitJadwal.java

Listing 3.8: PembangkitJadwal.java

```

1 | public int nomorAntrian() {
2 |     Random rand = new Random();
3 |     int nomor = rand.nextInt(10);
4 |     return nomor;
5 | }
6 |
7 | public void execute(DelegateExecution execution) throws Exception {
8 |     execution.getVariable("jadwalHari");
9 |     String jadwal = this.jadwalKedatangan(execution.getVariable("jadwalHari"));

```

```

10      int nomor = this.nomorAntrian();
11
12      execution.setVariable("jadwal", jadwal);
13      execution.setVariable("nomor", nomor);
14  }

```

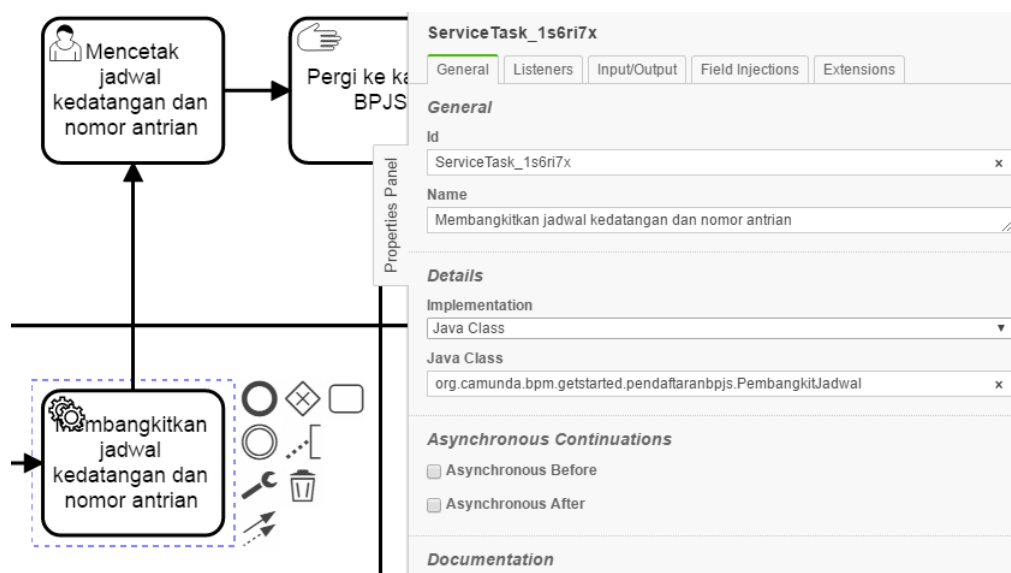
Listing 3.9: pilih-jadwal.html

```

1  <form>
2  <select
3      cam-variable-name="jadwalHari"
4      cam-variable-type="String"
5      cam-choices="jadwalHariPilihan"
6  >
7      <option value="senin">Senin</option>
8      <option value="selasa">Selasa</option>
9      <option value="rabu">Rabu</option>
10     <option value="kamis">Kamis</option>
11     <option value="jumat">Jumat</option>
12 </select>
13 </form>

```

Untuk menambahkan kode Java ke BPMN, pilih *implementation* Java Class pada modeler dan isi lokasi dari kelas Java tersebut. Contohnya sebagai berikut :

Gambar 3.12: Menghubungkan *Service Task* dengan kode Java

### 3.3 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

#### 3.3.1 Otomasi Kasus 1 - Pengajuan Proposal

Berikut adalah hasil otomasi kasus Pengajuan Proposal

1. John, sebagai bagian dari divisi sales mengunggah proposal

Camunda Tasklist

Keyboard Shortcuts Create task Start process John Doe

Mengunggah Proposal

Set follow-up date Set due date accounting, Sales John Doe

Form History Diagram Description

Choose File Dokumen.pdf

Save Complete

Gambar 3.13: Mengunggah Proposal

2. Peter, sebagai bagian dari divisi manajemen memeriksa dan menyetujui proposal

Camunda Tasklist

Keyboard Shortcuts Create task Start process Peter Meter

Memeriksa Proposal

Set follow-up date Set due date Management Peter Meter

Form History Diagram Description

Download Dokumen  
Apakah Proposal layak?  
☒

Save Complete

Gambar 3.14: Memeriksa Proposal

3. John, sebagai bagian dari divisi sales melihat proposal telah disetujui

Camunda Tasklist

Keyboard Shortcuts Create task Start process John Doe

Melihat Status Proposal

Set follow-up date Set due date Sales, accounting John Doe

Form History Diagram Description

Proposal sudah diterima  
Lihat Proposal

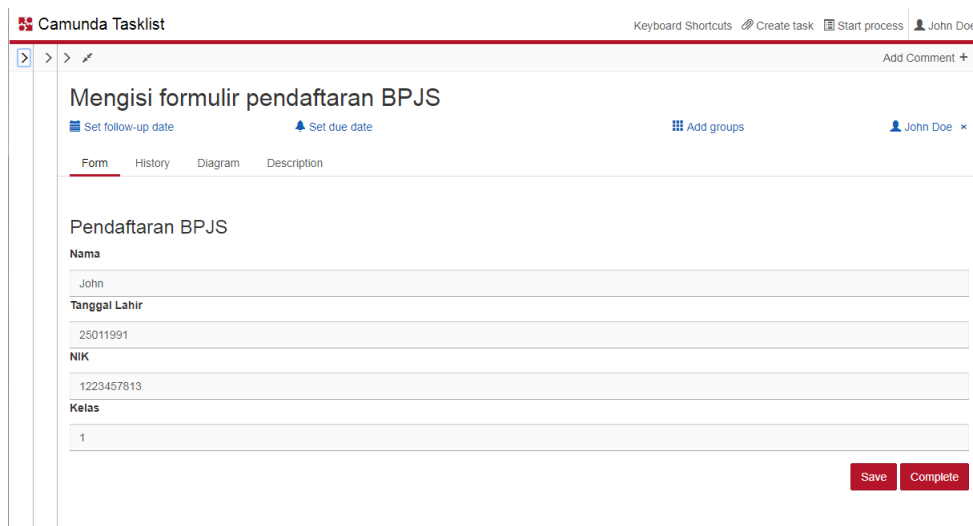
Save Complete

Gambar 3.15: Melihat Status Proposal

### 3.3.2 Otomasi Kasus 2 - Pendaftaran BPJS

Berikut adalah hasil otomasi kasus Pendaftaran BPJS

1. John mengisi data diri untuk mendaftar BPJS



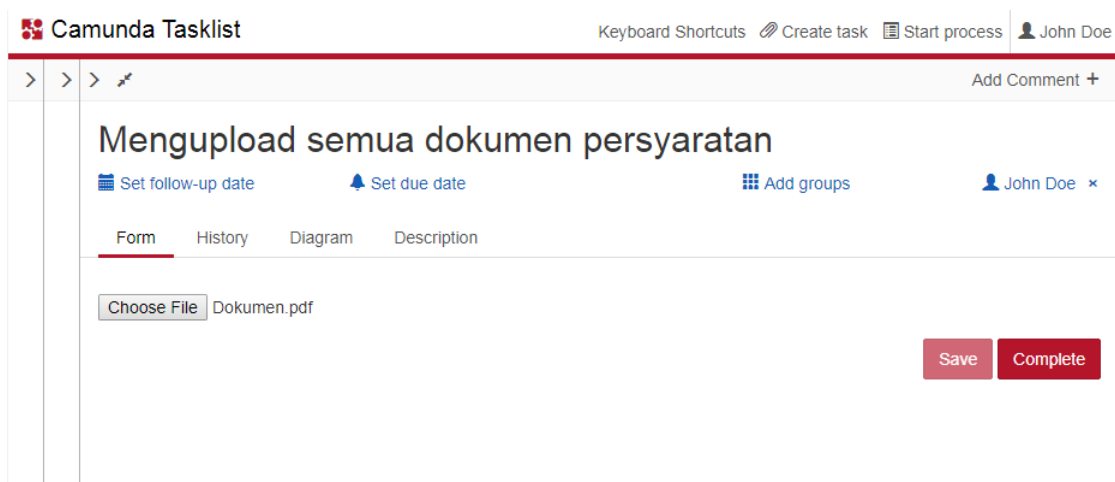
The screenshot shows the Camunda Tasklist interface for a task titled "Mengisi formulir pendaftaran BPJS". The task is assigned to "John Doe". The form contains the following fields:

- Nama**: John
- Tanggal Lahir**: 25011991
- NIK**: 1223457813
- Kelas**: 1

At the bottom right of the form are two buttons: "Save" and "Complete".

Gambar 3.16: Mendaftar BPJS

2. John mengunggah semua dokumen persyaratan BPJS



The screenshot shows the Camunda Tasklist interface for a task titled "Mengupload semua dokumen persyaratan". The task is assigned to "John Doe". The form contains the following elements:

- A "Choose File" button followed by the text "Dokumen.pdf".

At the bottom right of the form are two buttons: "Save" and "Complete".

Gambar 3.17: Mengunggah Dokumen

3. John melihat nomor pembayaran dan besar uang pendaftaran BPJS

The screenshot shows the Camunda Tasklist interface. At the top, there's a header with the Camunda logo, 'Camunda Tasklist', and navigation links: 'Keyboard Shortcuts', 'Create task', 'Start process', and a user profile 'John Doe'. Below the header is a breadcrumb trail with three arrows and a pencil icon, followed by an 'Add Comment +' button. The main title of the task is 'Melihat nomor pembayaran dan besar uang pendaftaran'. Below the title are four action buttons: 'Set follow-up date', 'Set due date', 'Add groups', and a user profile 'John Doe' with a close icon. There are four tabs: 'Form', 'History', 'Diagram', and 'Description'. The 'Form' tab is active, showing the task details: 'Uang Daftar: 50000' and 'Nomor: 7'. At the bottom right, there are two red buttons: 'Save' and 'Complete'.

Gambar 3.18: Melihat Nomor dan Biaya Pendaftaran

4. John memilih hari untuk verifikasi dokumen

The screenshot shows the Camunda Tasklist interface. At the top, there's a header with the Camunda logo, 'Camunda Tasklist', and navigation links: 'Keyboard Shortcuts', 'Create task', 'Start process', and a user profile 'John Doe'. Below the header is a breadcrumb trail with three arrows and a pencil icon, followed by an 'Add Comment +' button. The main title of the task is 'Memilih jadwal verifikasi dokumen'. Below the title are four action buttons: 'Set follow-up date', 'Set due date', 'Add groups', and a user profile 'John Doe' with a close icon. There are four tabs: 'Form', 'History', 'Diagram', and 'Description'. The 'Form' tab is active, showing the task details: 'Halaman pilih jadwal verifikasi dokumen.' and a dropdown menu with 'Rabu' selected. At the bottom right, there are two red buttons: 'Save' and 'Complete'.

Gambar 3.19: Memilih Hari

5. John Mencetak Jadwal Kedatangan dan Nomor Antrian

The screenshot shows the Camunda Tasklist interface. At the top, there's a header with the Camunda logo, 'Camunda Tasklist', and navigation links: 'Keyboard Shortcuts', 'Create task', 'Start process', and a user profile 'John Doe'. Below the header is a breadcrumb trail with three arrows and a pencil icon, followed by an 'Add Comment +' button. The main title of the task is 'Mencetak jadwal kedatangan dan nomor antrian'. Below the title are four action buttons: 'Set follow-up date', 'Set due date', 'Add groups', and a user profile 'John Doe' with a close icon. There are four tabs: 'Form', 'History', 'Diagram', and 'Description'. The 'Form' tab is active, showing the task details: 'Nomor : 4' and 'jadwal : rabu'. Below these, there is a button labeled 'Cetak Jadwal'. At the bottom right, there are two red buttons: 'Save' and 'Complete'.

Gambar 3.20: Mencetak Jadwal

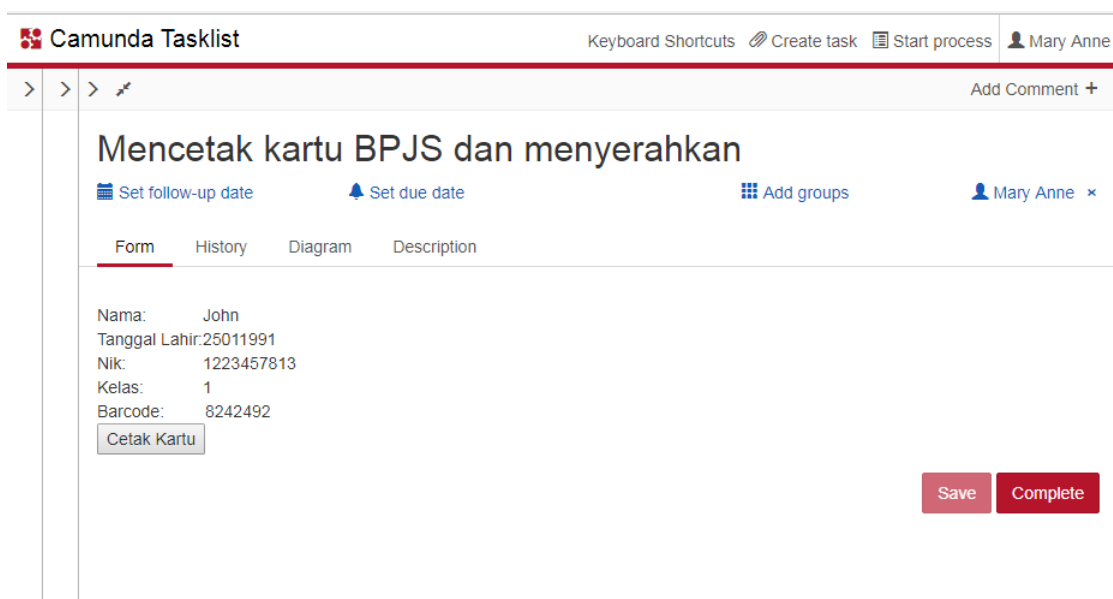
## 6. Mary Memverifikasi pendaftaran BPJS



The screenshot shows the Camunda Tasklist interface. At the top, the header includes the Camunda logo, the title 'Camunda Tasklist', and navigation links for 'Keyboard Shortcuts', 'Create task', 'Start process', and the user 'Mary Anne'. Below the header, there is a breadcrumb trail with three right-pointing arrows and a search icon, followed by an 'Add Comment +' button. The main task title is 'Memverifikasi pendaftaran dan attachment dokumen persyaratan dan keaslian'. Below the title, there are four action buttons: 'Set follow-up date', 'Set due date', 'Add groups', and a user selection dropdown showing 'Mary Anne'. Below these buttons are four tabs: 'Form', 'History', 'Diagram', and 'Description'. The 'Form' tab is active, displaying a form with the following fields: 'Download Dokumen Persyaratan' (a link), 'Download' (a link), 'Nama: John', 'Tanggal Lahir: 25011991', 'Nik: 1223457813', 'Kelas: 1', 'Nomor: 4', 'jadwal: rabu', and 'Apakah dokumen dan persyaratan lengkap?' (a checkbox). At the bottom right of the form, there are two red buttons: 'Save' and 'Complete'.

Gambar 3.21: Verifikasi Pendaftaran BPJS

## 7. Mary Mencetak Kartu BPJS dan Menyerahkannya ke John



The screenshot shows the Camunda Tasklist interface. At the top, the header includes the Camunda logo, the title 'Camunda Tasklist', and navigation links for 'Keyboard Shortcuts', 'Create task', 'Start process', and the user 'Mary Anne'. Below the header, there is a breadcrumb trail with three right-pointing arrows and a search icon, followed by an 'Add Comment +' button. The main task title is 'Mencetak kartu BPJS dan menyerahkan'. Below the title, there are four action buttons: 'Set follow-up date', 'Set due date', 'Add groups', and a user selection dropdown showing 'Mary Anne'. Below these buttons are four tabs: 'Form', 'History', 'Diagram', and 'Description'. The 'Form' tab is active, displaying a form with the following fields: 'Nama: John', 'Tanggal Lahir: 25011991', 'Nik: 1223457813', 'Kelas: 1', 'Barcode: 8242492', and a 'Cetak Kartu' button. At the bottom right of the form, there are two red buttons: 'Save' and 'Complete'.

Gambar 3.22: Mencetak Kartu BPJS

## 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

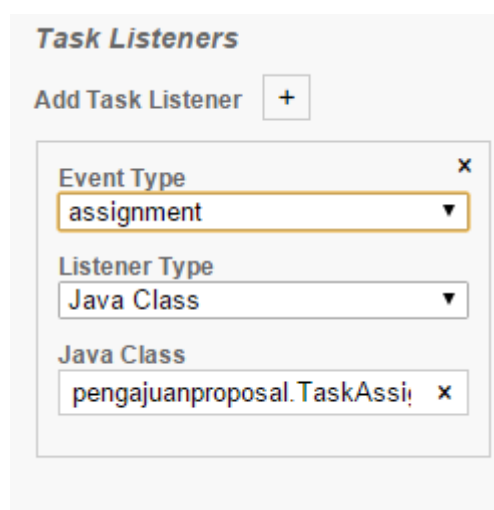
##### 4.1.1 *Event* yang Terkait dengan Integrasi Sistem Email

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. It includes an 'Add Task Listener' button with a plus icon. Below this, there are 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 icon (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.

## 4.2 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.

Selain itu, dibutuhkan partisipan yang memiliki perannya masing-masing. Seorang Desainer bertugas merancang BPMN, seorang admin bertugas mengatur jalannya otomasi proses bisnis, sedangkan aktor bertugas mengerjakan *tasks*

1. Tugas Desainer Berdasarkan perancangan sistem di atas, seorang desainer model proses memiliki beberapa tugas, yaitu :
  - (a) Merancang model proses.
  - (b) Menambahkan form HTML pada *user task*, *implementasi service task*, *task listener* untuk propagasi email, dan berbagai atribut lainnya sesuai kebutuhan.
  - (c) Mendelegasikan task kepada *user/group* yang akan mengerjakan.
2. Tugas Admin



- (a) Membuat alamat email yang merepresentasikan sistem.
- (b) Menambahkan *username*, *password*, dan *host* email pada kode task listener yang berhubungan dengan propagasi email.
- (c) Menambahkan user/group yang akan mengerjakan *tasks* pada Camunda Admin.
- (d) Menjalankan dan memulai proses.

### 3. Tugas Aktor

- (a) Memberitahu alamat email kepada admin.
- (b) Mengerjakan *task*.

## 4.3 Perancangan Sistem

### 4.3.1 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@yahoo.com dan bagian dari grup *accounting*.
3. Peter, dengan alamat email petercamunda@gmail.com dan bagian dari grup *management*.

### 4.3.2 Perancangan 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)).

### 4.3.3 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.

### 4.3.4 Task Event Listener

*Task Event Listener* bereaksi ke *Task Event* seperti *Created*, *Assigned*, dan *Completed*. Implementasi *Task Listener* berupa kode Java dapat dilihat di bawah ini :

Listing 4.1: PembangkitJadwal.java

```
1 | public class TaskAssignmentListener implements TaskListener {  
2 |     public void notify(DelegateTask delegateTask) {  
3 |  
4 |     }  
5 | }
```

Parameter *delegateTask* dapat mengambil maupun menyimpan informasi *task* yang sedang berjalan. Beberapa informasi yang dapat diambil maupun disimpan adalah sebagai berikut :

- *getAssignee()*, mendapatkan pemilik *task* dan *setAssignee* untuk menentukan pemilik *task*
- *getId()*, mendapatkan id dari *task*
- *getName()*, mendapatkan nama *task* dan *setName* untuk menentukan nama *task*
- *getCandidates()*, mendapatkan *candidate users/groups* dan *addCandidateGroup()* / *addCandidateUser()* untuk menentukan *candidate users/groups*

## BAB 5

### IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan diimplementasikan kode program untuk propagasi email dan pengujian dua skenario yang ada pada Bab 3.

#### 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 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
2  session = Session.getDefaultInstance(props, null);
3      message = new MimeMessage(session);
4      message.addRecipient(Message.RecipientType.TO, new InternetAddress(recipient)
5      );
6      message.setSubject("Task_" + delegateTask.getName());
7
8  String name = user.getFirstName();
9      String emailBody = "";
10     emailBody += "Dear_" + name + "<br>";
11     emailBody += "Anda_mendapatkan_task_" + taskName + "_untuk_dikerjakan.<br>";
12     emailBody += "Segera_akses_http://localhost:1234/camunda/app/tasklist/default/#/?"
13     task="+taskId_" + "untuk_menjalankannya.<br>";
14     emailBody += "Terima_kasih.";
15     message.setContent(emailBody, "text/html");

```

- Kode untuk mengirimkan email.

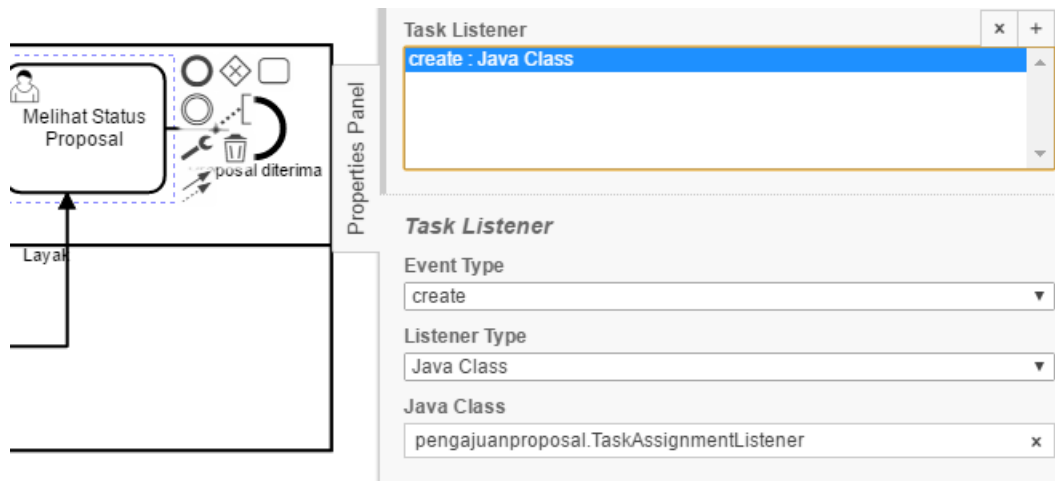
Listing 5.7: TaskAssignmentListener.java

```

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

```

Implementasi algoritma pengiriman email (TaskAssignmentListener.java) dikaitkan dengan setiap *user task* pada BPMN menggunakan *event create*. Contohnya adalah sebagai berikut :



Gambar 5.1: Task Listener pada BPMN

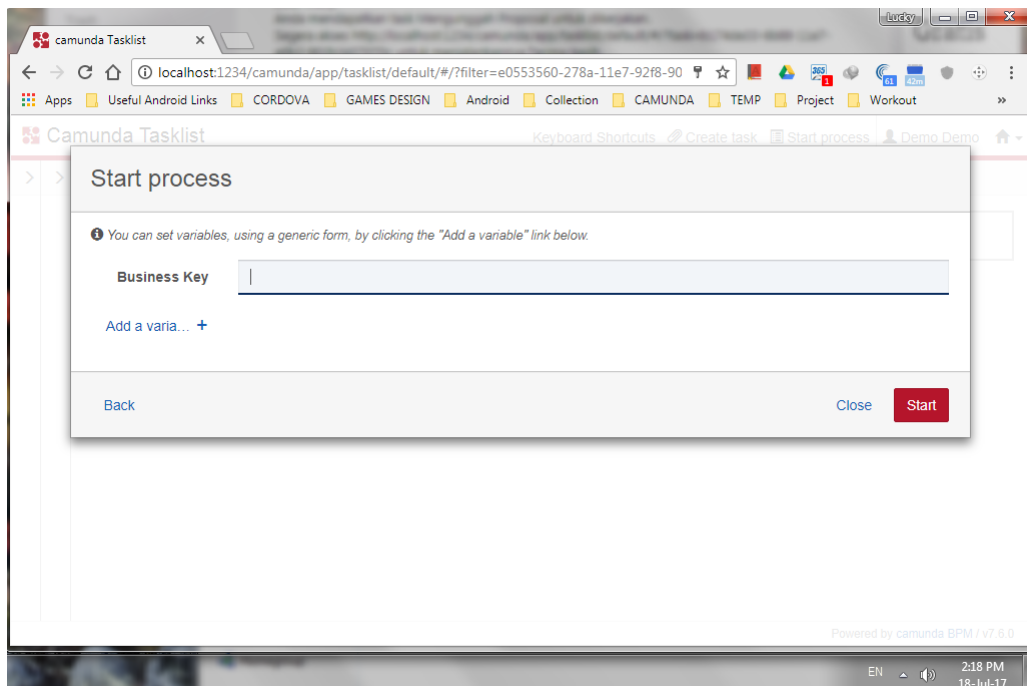
## 5.3 Pengujian

Pengujian dilakukan pada skenario yang ada pada subbab 3.1.1 Masalah Proses Bisnis. Ada dua skenario yang diuji, yaitu Pengajuan Proposal dan Pendaftaran BPJS. Kriteria yang diuji adalah berhasil atau tidaknya pengiriman email dan lama pengiriman email.

### 5.3.1 Pengujian Kasus Pengajuan Proposal

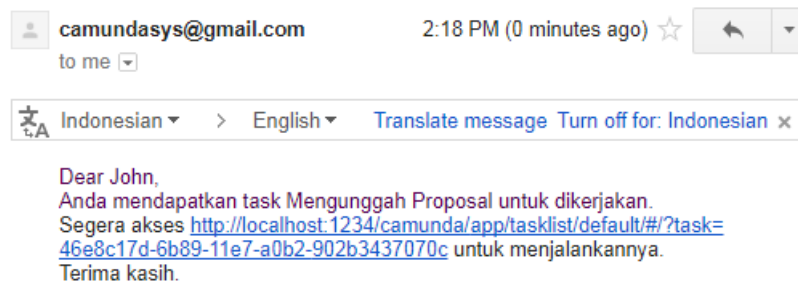
Workflow Pengajuan Proposal dapat dilihat pada subbab 3.1.2

1. Memulai proses Pengajuan Proposal pada 2:18 PM

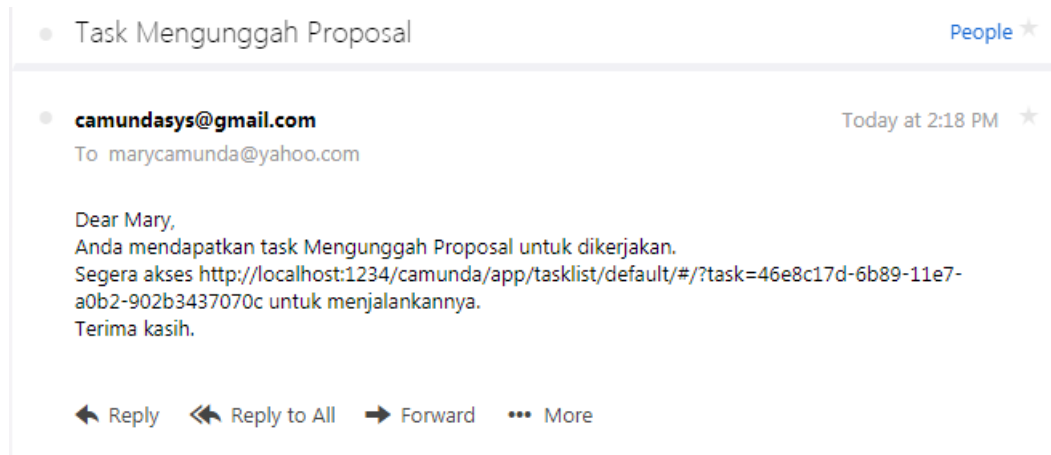


Gambar 5.2: Memulai Proses Pengajuan Proposal

2. John dan Mary menerima email untuk mengunggah proposal pada 2:18 PM

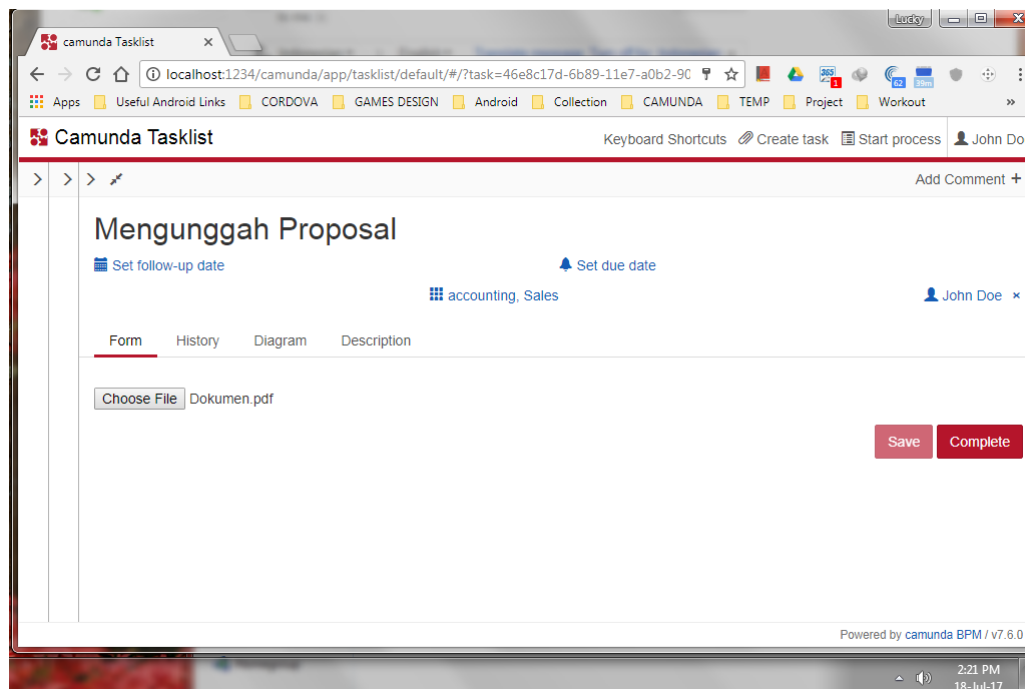


Gambar 5.3: Email Mengunggah Proposal



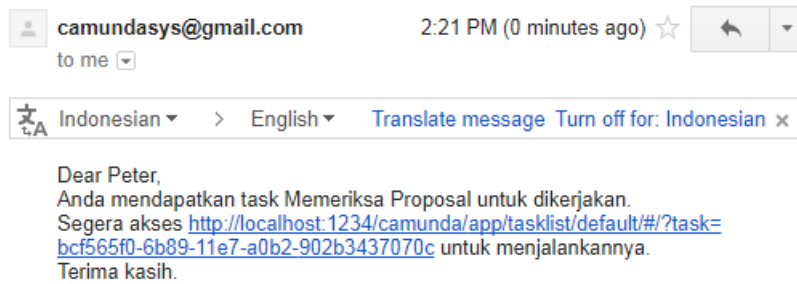
Gambar 5.4: Email Mengunggah Proposal

3. John mengklaim *task* dan mengunggah proposal pada 2:21 PM



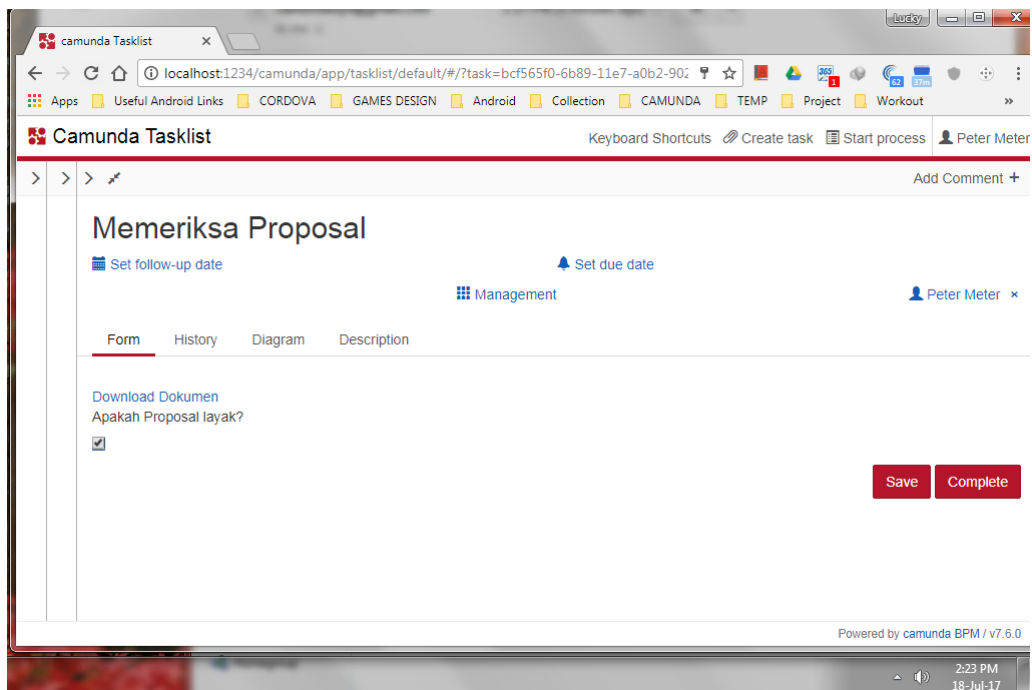
Gambar 5.5: Mengunggah Proposal

4. Peter menerima email untuk memeriksa proposal pada 2:21 PM



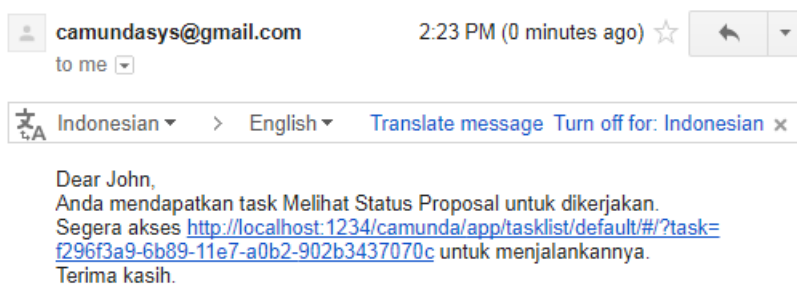
Gambar 5.6: Email Memeriksa Proposal

5. Peter memeriksa proposal pada 2:23 PM



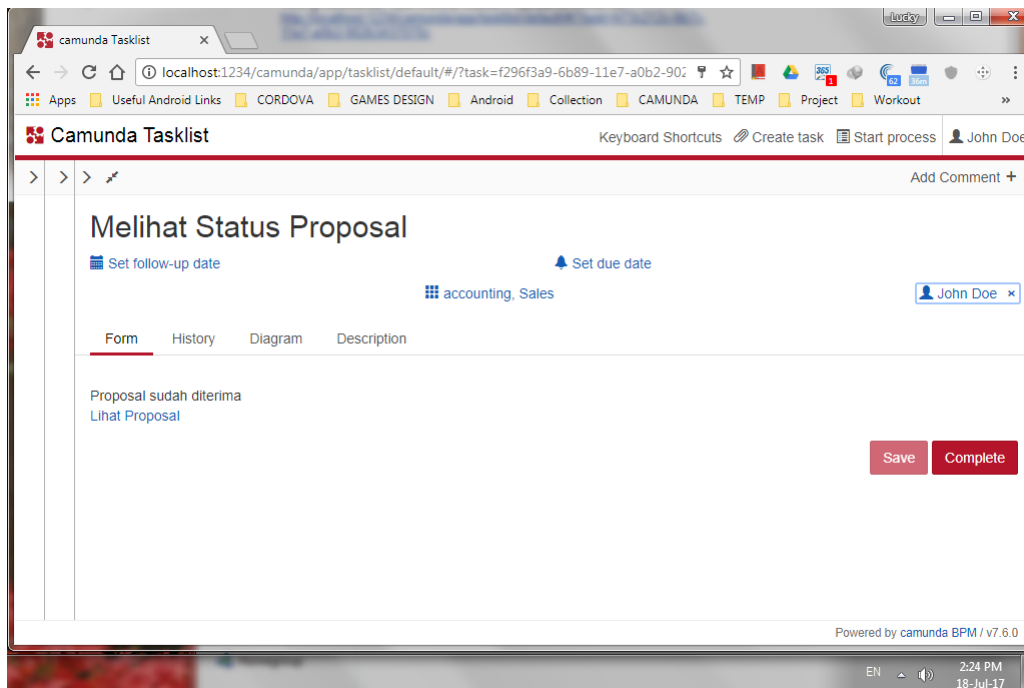
Gambar 5.7: Peter Memeriksa Proposal

6. John menerima email untuk melihat status proposal pada 2:23 PM



Gambar 5.8: Email Melihat Status Proposal

7. John melihat status proposal pada 2:24 PM

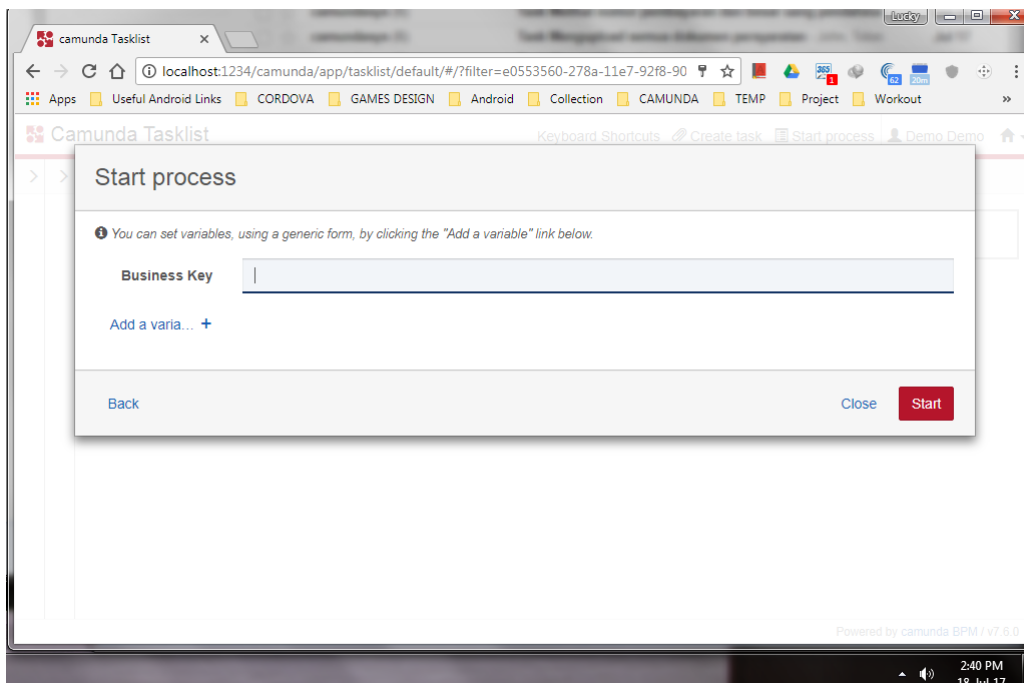


Gambar 5.9: John Melihat Status Proposal

### 5.3.2 Pengujian Kasus Pendaftaran BPJS

Workflow Kasus Pendaftaran BPJS dapat dilihat pada subbab 3.1.2

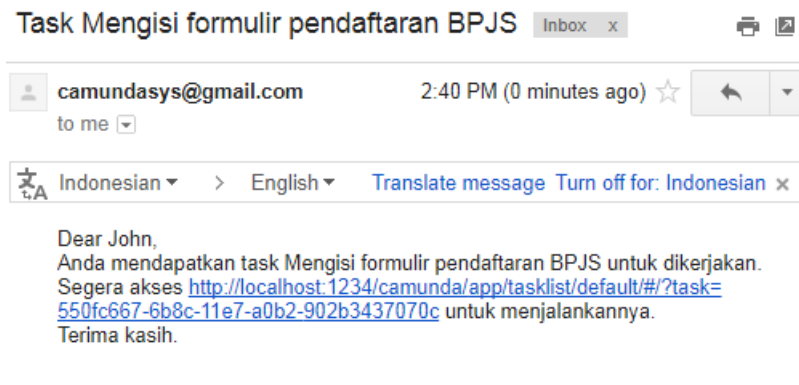
1. Proses Pendaftaran BPJS dimulai pukul 2:40 PM.



Gambar 5.10: Memulai Proses Pendaftaran BPJS

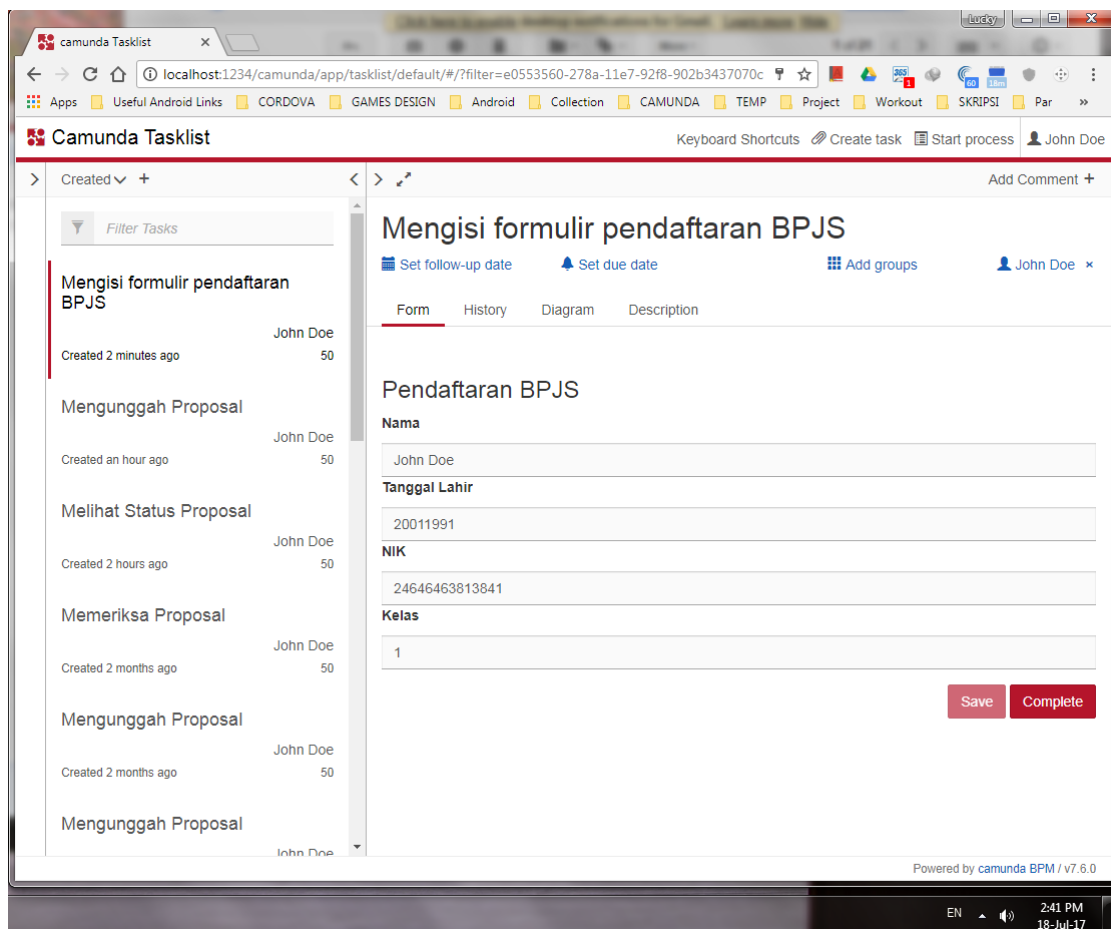
2. John menerima email untuk mengisi formulir pendaftaran BPJS pada pukul 2:40PM.





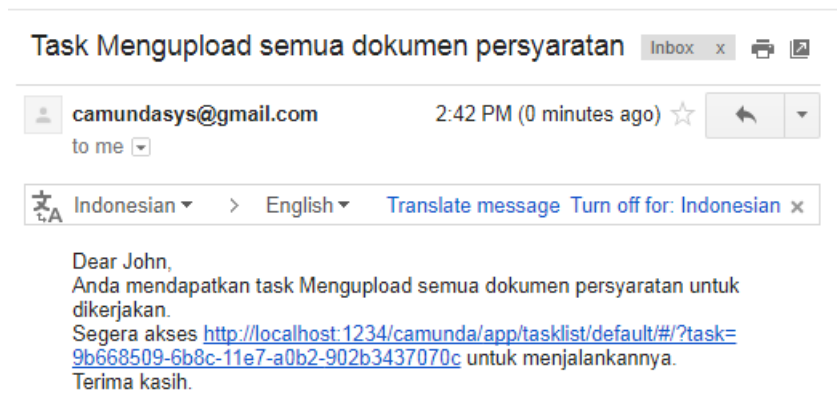
Gambar 5.11: Email Mengisi Formulir Pendaftaran BPJS

3. John mengisi formulir pendaftaran BPJS pukul 2:41 PM



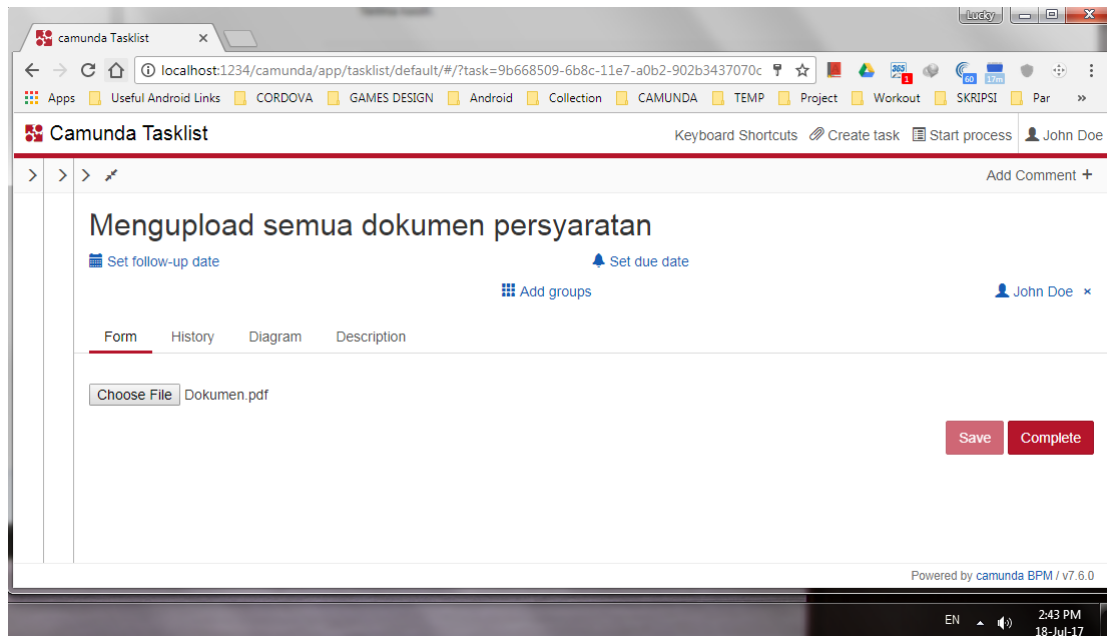
Gambar 5.12: Mengisi Formulir Pendaftaran BPJS

4. John menerima email untuk mengunggah semua dokumen persyaratan pada pukul 2:42 PM.



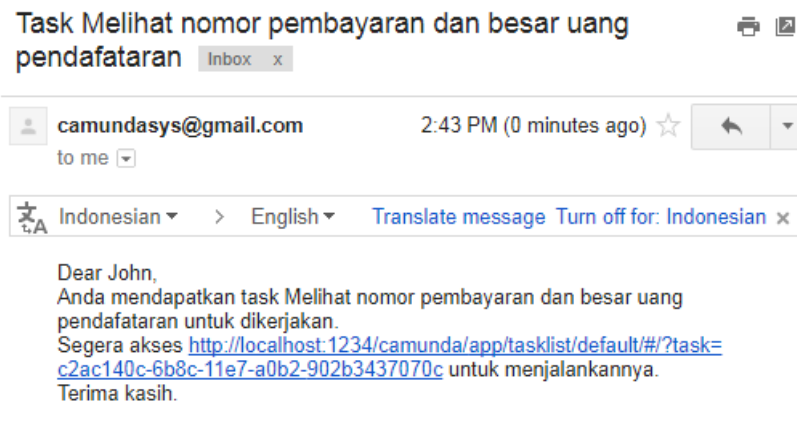
Gambar 5.13: Email Mengunggah Dokumen Persyaratan

5. John mengunggah semua dokumen persyaratan pada pukul 2:43 PM



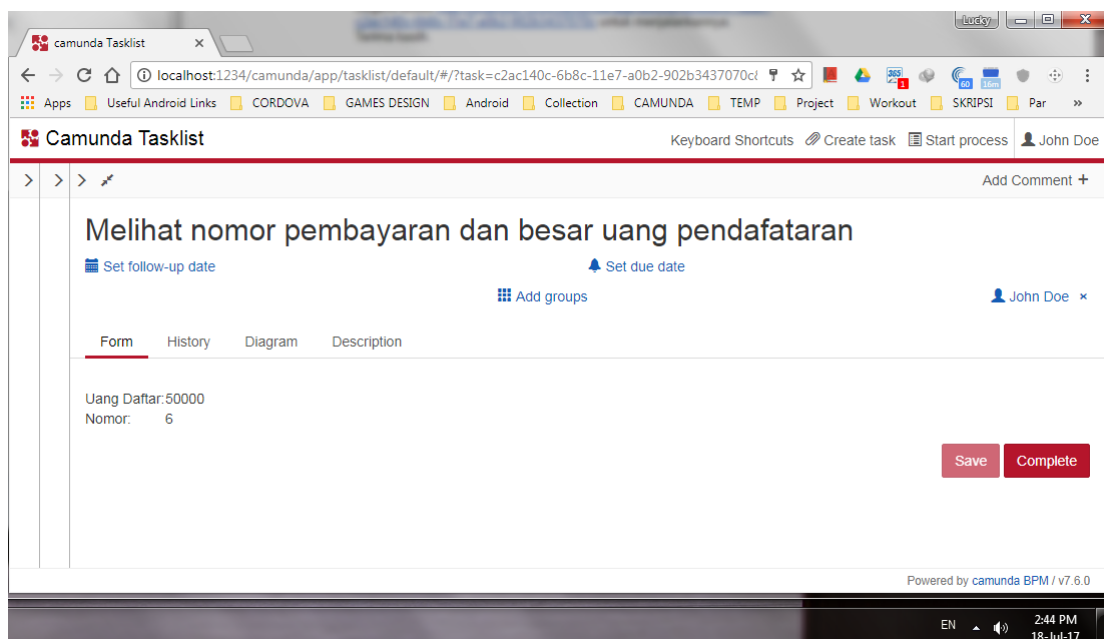
Gambar 5.14: Mengunggah Dokumen Persyaratan

6. John menerima email untuk melihat nomor pembayaran dan besar uang pendaftaran pada pukul 2:43 PM



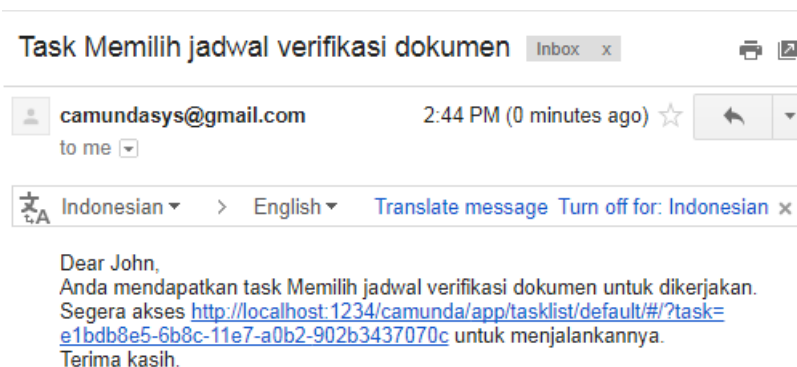
Gambar 5.15: Email Nomor Pembayaran dan Uang Pendaftaran

7. John melihat nomor pembayaran dan besar uang pendaftaran pada pukul 2:44 PM



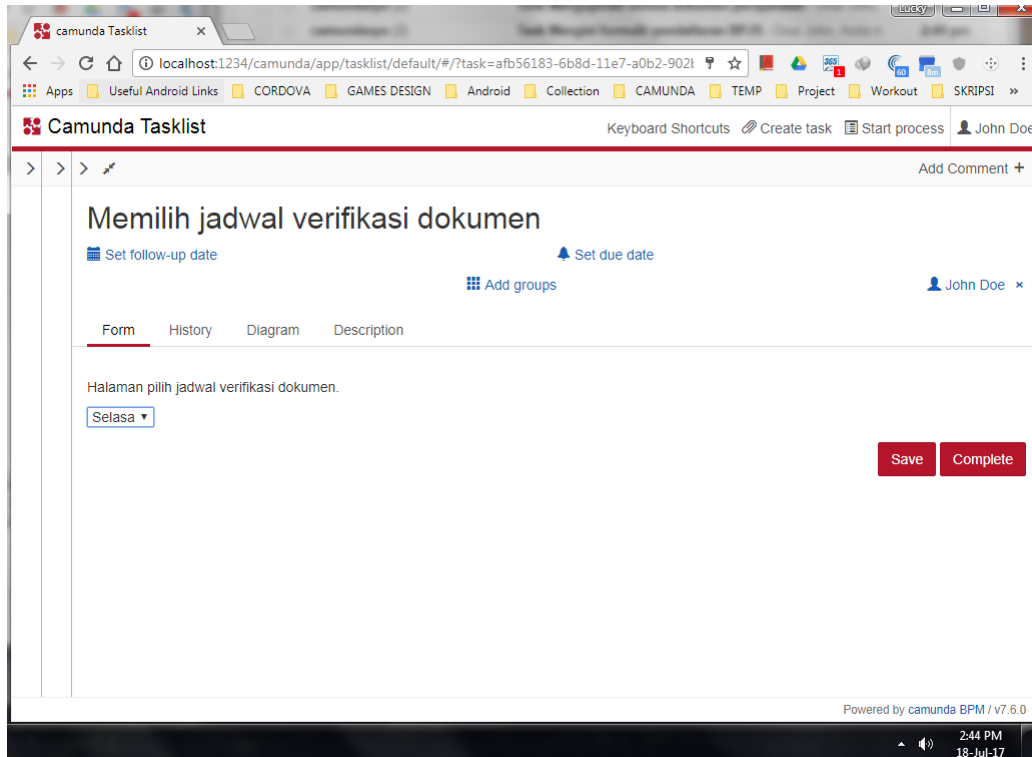
Gambar 5.16: Melihat Nomor Pembayaran dan Uang Pendaftaran

8. John menerima email untuk memilih jadwal verifikasi dokumen pada pukul 2:43 PM



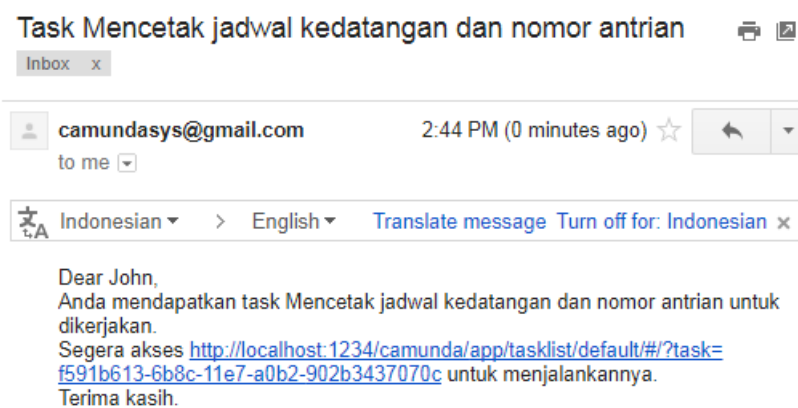
Gambar 5.17: Email Memilih Jadwal Verifikasi Dokumen

9. John memilih jadwal verifikasi dokumen pada pukul 2:44 PM



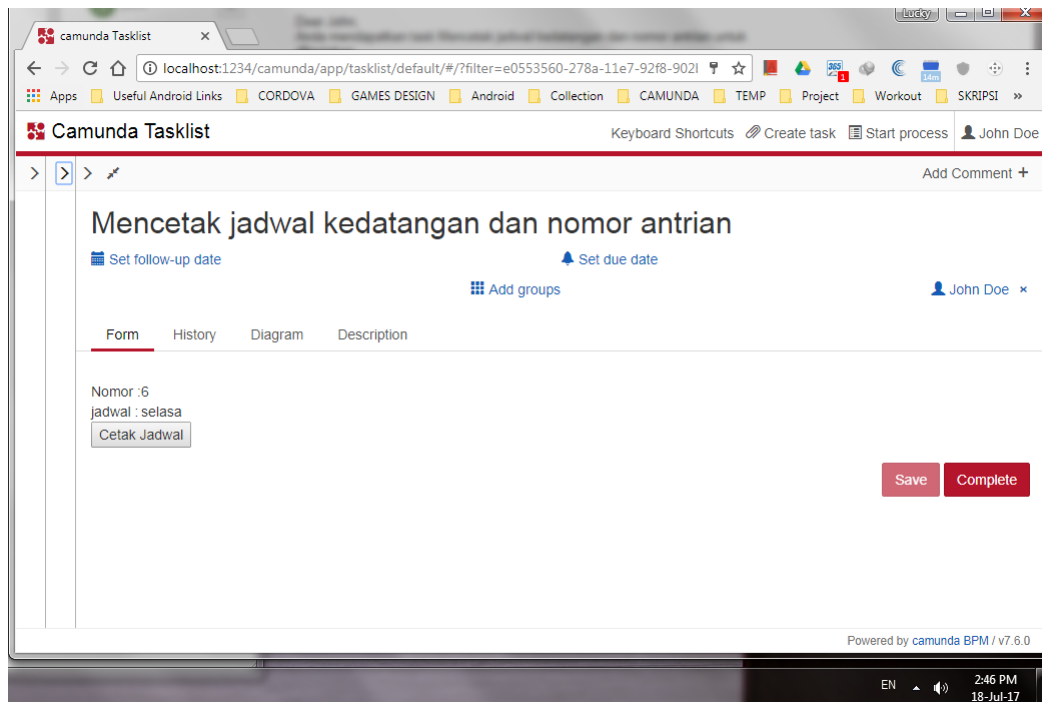
Gambar 5.18: Memilih Jadwal Verifikasi Dokumen

10. John menerima email untuk mencetak jadwal dan nomor antrian pada pukul 2:44 PM



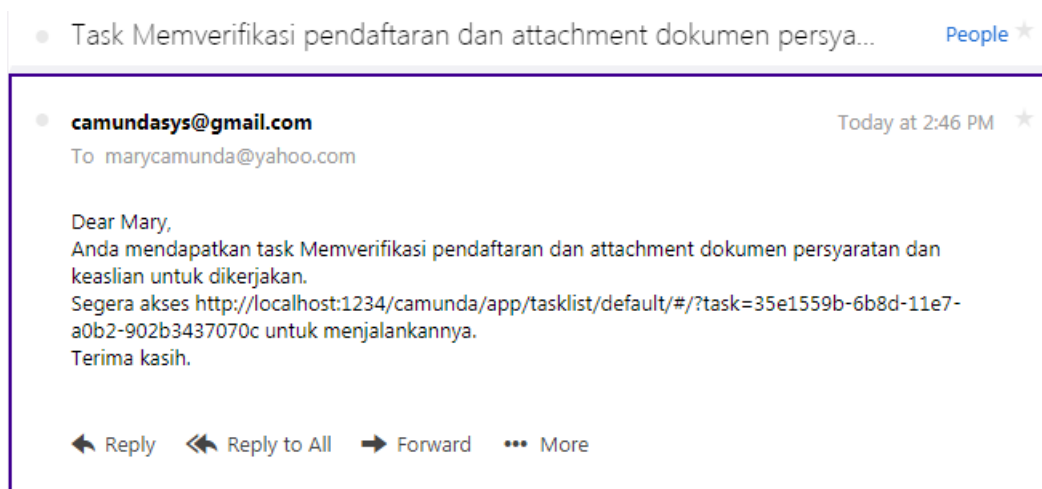
Gambar 5.19: Email Mencetak Jadwal

11. John mencetak jadwal dan nomor antrian pada pukul 2:46 PM



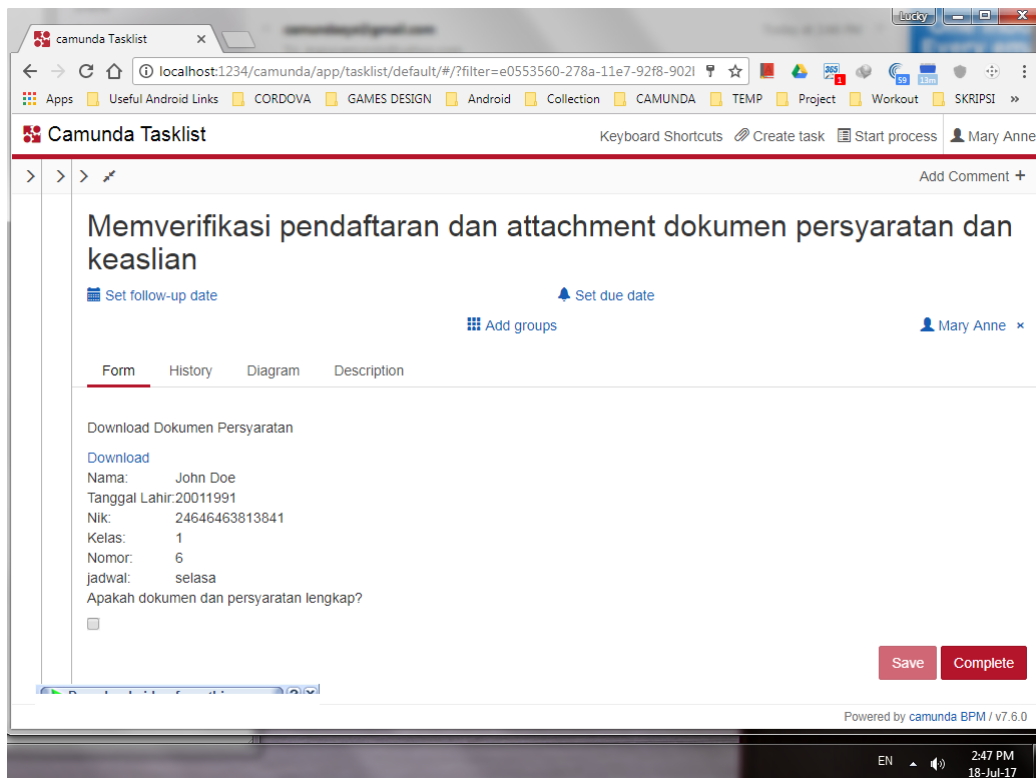
Gambar 5.20: Mencetak Jadwal dan Nomor Antrian

12. Mary, sebagai petugas BPJS menerima email untuk memverifikasi pendaftaran pada pukul 2:46 PM



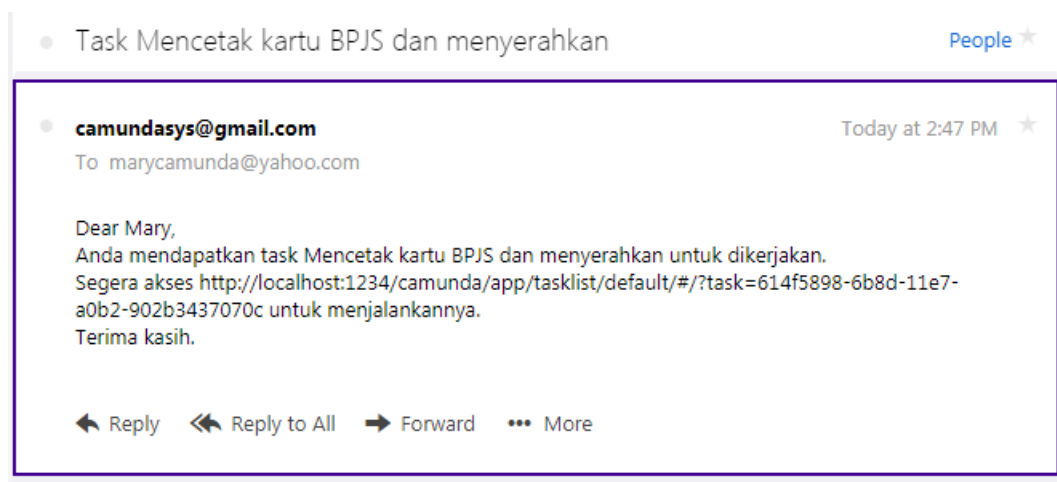
Gambar 5.21: Email Verifikasi Pendaftaran

13. Mary memverifikasi pendaftaran dan semua persyaratan pada pukul 2:47 PM



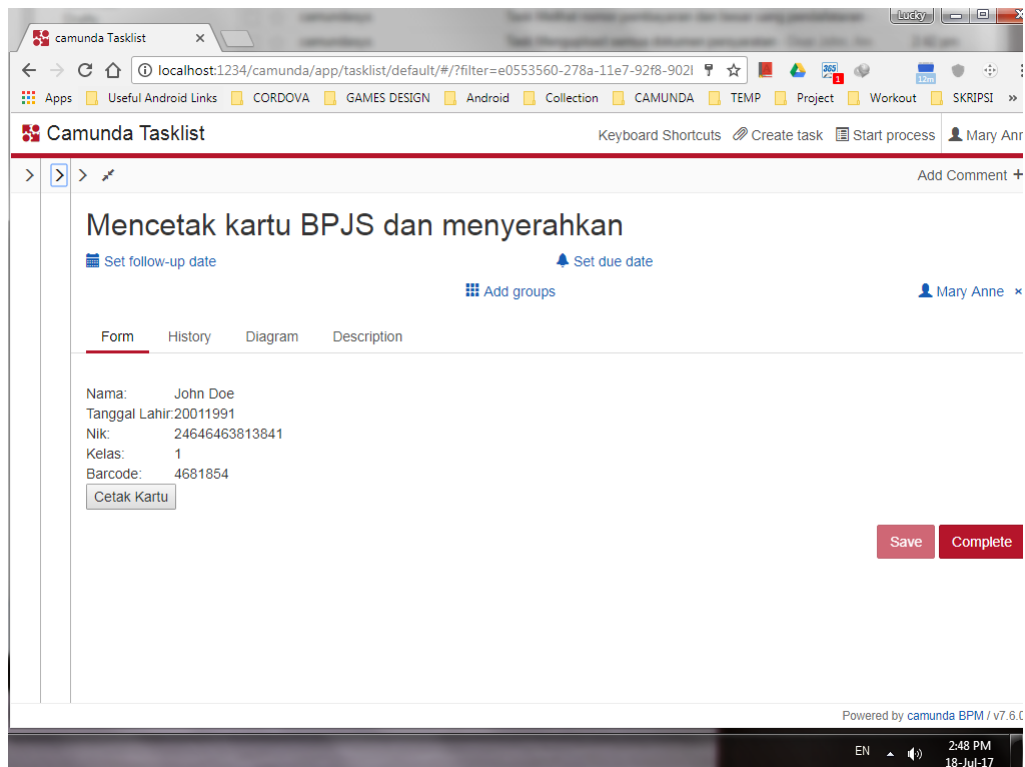
Gambar 5.22: Memverifikasi Pendaftaran dan Semua Persyaratan

14. Mary menerima email untuk mencetak kartu BPJS pada pukul 2:47 PM



Gambar 5.23: Email Mencetak Kartu BPJS

15. Mary mencetak kartu BPJS pada pukul 2:48 PM



Gambar 5.24: Mencetak Kartu BPJS

## 5.4 Hasil Pengujian

Berdasarkan pengujian dua kasus yang telah dilakukan (Pengajuan Proposal dan Pendaftaran BPJS), hasilnya adalah :

1. Seluruh *user task* mengirimkan email ke pemilik *task*.
2. Email langsung dikirim ke pemilik *task* setelah *task* siap dikerjakan. Dapat dilihat dari waktu *task* sebelumnya selesai dan waktu email diterima oleh pemilik *task* yang akan dikerjakan.





## 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 yaitu Pengajuan Proposal dan Pendaftaran BPJS. Berdasarkan subbab 5.4 Hasil Pengujian, didapati bahwa setiap *user task* yang disisipkan *TaskAssignmentListener.java* dapat mengirim email ke pemilik *user task* masing-masing segera setelah *task* siap untuk dikerjakan.

#### 6.2 Saran

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

1. Menambahkan statistik efektifitas proses bisnis.
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     String email;
39
40     String[] recipient;
41
42     static Properties props;
43     static Session session;
44     static MimeMessage message;
45
46
47     public void notify(DelegateTask delegateTask) {
48         assignee = delegateTask.getAssignee();
49         taskId = delegateTask.getId();
50         taskName = delegateTask.getName();
51         delegateTask.getCandidates();
52
53         if (assignee != null) {
54             IdentityService identityService = Context.getProcessEngineConfiguration().getIdentityService
55                 ();
56             User user = identityService.createUserQuery().userId(assignee).singleResult();
57             if (user != null) {
58                 this.sendEmail(user);
59             }
60         }
61         else {
62             TaskEntity task = (TaskEntity) delegateTask;
63             List<IdentityLinkEntity> identityLinks = task.getIdentityLinks();
```

```

64         for(IdentityLinkEntity link : identityLinks) {
65             if(link.getType().equals(IdentityLinkType.CANDIDATE)) {
66                 if(link.isUser()) {
67                     User user = Context.getProcessEngineConfiguration().getIdentityService().
68                         createUserQuery().userId(link.getUserId()).singleResult();
69                     sendEmail(user);
70                 }
71                 if(link.isGroup()) {
72                     List<User> users = Context.getProcessEngineConfiguration().getIdentityService
73                         ().createUserQuery().memberOfGroup(link.getGroupId()).list();
74                     for(User user : users) {
75                         sendEmail(user);
76                     }
77                 }
78             }
79         }
80
81     public void sendEmail(User user){
82         try {
83             props = System.getProperties();
84             props.put("mail.smtp.port", "587");
85             props.put("mail.smtp.auth", "true");
86             props.put("mail.smtp.starttls.enable", "true");
87
88
89             session = Session.getDefaultInstance(props, null);
90             message = new MimeMessage(session);
91             message.addRecipient(Message.RecipientType.TO, new InternetAddress(user.getEmail()));
92             message.setSubject("Task" + taskName);
93
94             String name = user.getFirstName();
95             String emailBody = "";
96             emailBody += "Dear" + name + "<br>";
97             emailBody += "Anda mendapatkan task" + taskName + " untuk dikerjakan.<br>";
98             emailBody += "Segera akses http://localhost:1234/camunda/app/tasklist/default/#/?task="+
99                 taskId + " untuk menjalankannya.<br>";
100            emailBody += "Terima kasih.";
101            message.setContent(emailBody, "text/html");
102
103            Transport transport = session.getTransport("smtp");
104            transport.connect(HOST, USER, PWD);
105            transport.sendMessage(message, message.getAllRecipients());
106            transport.close();
107        } catch (NoSuchProviderException ex) {
108            Logger.getLogger(TaskAssignmentListener.class.getName()).log(Level.SEVERE, null, ex);
109        } catch (MessagingException ex) {
110            Logger.getLogger(TaskAssignmentListener.class.getName()).log(Level.SEVERE, null, ex);
111        }
112    }
113 }

```

## 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>
```





# LAMPIRAN C

## KODE SKENARIO

### C.1 Kasus 1 - Pengajuan Proposal

Listing C.1: PengajuanProposal.java

```
1 package pengajuanproposal;
2
3 import org.camunda.bpm.application.ProcessApplication;
4 import org.camunda.bpm.application.impl.ServletProcessApplication;
5
6 @ProcessApplication("PengajuanProposalApp")
7 public class PengajuanProposal extends ServletProcessApplication{
8
9 }
```

Listing C.2: MengunggahProposal.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-file-size="10000000" />
9     </form>
10 </body>
11 </html>
```

Listing C.3: MemeriksaProposal.html

```
1 <html>
2 <head></head>
3
4 <body>
5 <form role="form" name="form">
6     <a cam-file-download="proposal">Download Dokumen</a>
7
8
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
16 </form>
17 </body>
18 </html>
```

Listing C.4: MelihatStatusProposal.html

```
1 <html>
2     <head></head>
3     <body>
4         <h> Proposal sudah diterima </h>
5
6         <form role="form" name="form">
7             <a cam-file-download="proposal">Lihat Proposal</a>
8         </form>
9
10
```

```
11 |
12 | </html>
```

## C.2 Kasus 2 - Pendaftaran BPJS

Listing C.5: PendaftaranBPJS.java

```
1 | package pengajuanproposal;
2 |
3 | package org.camunda.bpm.getstarted.pendaftaranbpjs;
4 |
5 | import org.camunda.bpm.application.ProcessApplication;
6 | import org.camunda.bpm.application.impl.ServletProcessApplication;
7 |
8 | @ProcessApplication("PendaftaranBPJS_App")
9 | public class PendaftaranBPJS extends ServletProcessApplication {
10 |
11 | }
```

Listing C.6: PembangkitBarcode.java

```
1 | package org.camunda.bpm.getstarted.pendaftaranbpjs;
2 |
3 | import java.util.Random;
4 |
5 | import org.camunda.bpm.engine.delegate.DelegateExecution;
6 | import org.camunda.bpm.engine.delegate.JavaDelegate;
7 |
8 | public class PembangkitBarcode implements JavaDelegate{
9 |
10 |     public int getBarcode(){
11 |         Random rand = new Random();
12 |         int nomor = rand.nextInt(10000000)+100000;
13 |         return nomor;
14 |     }
15 |
16 |     public void execute(DelegateExecution execution) throws Exception {
17 |         String barcode = ""+this.getBarcode();
18 |         execution.setVariable("barcode", barcode);
19 |     }
20 | }
```

Listing C.7: PembangkitJadwal.java

```
1 | package org.camunda.bpm.getstarted.pendaftaranbpjs;
2 |
3 | import java.util.Random;
4 | import java.util.logging.Logger;
5 |
6 | import org.camunda.bpm.engine.delegate.DelegateExecution;
7 | import org.camunda.bpm.engine.delegate.JavaDelegate;
8 | import org.camunda.bpm.engine.runtime.ProcessInstance;
9 |
10 | public class PembangkitJadwal implements JavaDelegate{
11 |     public final static Logger LOGGER = Logger.getLogger("pembangkit-jadwal");
12 |     public String jadwalKedatangan(Object hari){
13 |
14 |         return hari+"";
15 |     }
16 |
17 |     public int nomorAntrian(){
18 |         Random rand = new Random();
19 |         int nomor = rand.nextInt(10);
20 |         return nomor;
21 |     }
22 |
23 |     public void execute(DelegateExecution execution) throws Exception {
24 |         execution.getVariable("jadwalHari");
25 |         String jadwal = this.jadwalKedatangan(execution.getVariable("jadwalHari"));
26 |         int nomor = this.nomorAntrian();
27 |
28 |         execution.setVariable("jadwal", jadwal);
29 |         execution.setVariable("nomor", nomor);
30 |
31 |     }
32 | }
```

Listing C.8: PembangkitNomor.java

```
1 | package org.camunda.bpm.getstarted.pendaftaranbpjs;
```

```

2
3 import java.util.Random;
4
5 import java.util.logging.Logger;
6
7 import org.camunda.bpm.engine.delegate.DelegateExecution;
8 import org.camunda.bpm.engine.delegate.JavaDelegate;
9
10 public class PembangkitNomor implements JavaDelegate{
11     public final static Logger LOGGER = Logger.getLogger("pendaftaran-bpjs");
12     int nomor;
13     int uangDaftar;
14
15     public int nomorPembayaran() {
16
17         Random rand = new Random();
18         nomor = rand.nextInt(10)+1;
19
20         return nomor;
21     }
22
23     public int uangPendaftaran() {
24         uangDaftar = 50000;
25         return uangDaftar;
26     }
27
28     public void execute(DelegateExecution execution) throws Exception {
29
30
31         this.nomorPembayaran();
32         this.uangPendaftaran();
33         execution.setVariable("uangDaftar", uangDaftar);
34         execution.setVariable("nomor", nomor);
35
36     }
37
38 }

```

Listing C.9: pendaftaran-bpjs.html

```

1 <html>
2 <head><title>Pendaftaran BPJS</title></head>
3 <body>
4     <form name="pendaftaranBPJS" role="form">
5         <h3>Pendaftaran BPJS</h3>
6
7         <div class="control-group">
8             <label class="control-label" for="nama">Nama</label>
9             <div class="controls">
10                 <input id="nama"
11                     class="form-control"
12                     cam-variable-name = "nama"
13                     cam-variable-type = "String"
14                     type="text"
15                     required>
16             </div>
17             <label class="control-label" for="tanggalLahir">Tanggal Lahir</label>
18             <div class="controls">
19                 <input id="tanggalLahir"
20                     class="form-control"
21                     cam-variable-name = "tanggalLahir"
22                     cam-variable-type = "String"
23                     type="text"
24                     required>
25             </div>
26             <label class="control-label" for="nik">NIK</label>
27             <div class="controls">
28                 <input id="nik"
29                     class="form-control"
30                     cam-variable-name = "nik"
31                     cam-variable-type = "String"
32                     type="text"
33                     required>
34             </div>
35             <label class="control-label" for="kelas">Kelas</label>
36             <div class="controls">
37                 <input id="kelas"
38                     class="form-control"
39                     cam-variable-name = "kelas"
40                     cam-variable-type = "String"
41                     type="text"
42                     required>
43             </div>
44
45

```

```

46 |     </form>
47 |
48 | </body>
49 | </html>

```

Listing C.10: upload-dokumen.html

```

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

```

Listing C.11: pilih-jadwal.html

```

1 | <html>
2 |
3 | <head></head>
4 |
5 | <body>
6 |     <p>Halaman pilih jadwal verifikasi dokumen.</p>
7 |     <form>
8 |         <select
9 |             cam-variable-name="jadwalHari"
10 |             cam-variable-type="String"
11 |             cam-choices="jadwalHariPilihan"
12 |         >
13 |             <option value="senin">Senin</option>
14 |             <option value="selasa">Selasa</option>
15 |             <option value="rabu">Rabu</option>
16 |             <option value="kamis">Kamis</option>
17 |             <option value="jumat">Jumat</option>
18 |         </select>
19 |     </form>
20 |
21 |
22 | </body>
23 | </html>

```

Listing C.12: nomor-pembayaran.html

```

1 | <html>
2 |
3 | <head></head>
4 |
5 |
6 |
7 | <body>
8 | <form role="form" name="form">
9 |     <script cam-script type="text/form-script">
10 |         camForm.on('form-loaded', function() {
11 |             camForm.variableManager.fetchVariable('uangDaftar');
12 |             camForm.variableManager.fetchVariable('nomor');
13 |
14 |         });
15 |         camForm.on('variables-restored', function() {
16 |             $scope.uangDaftar = camForm.variableManager.variableValue('uangDaftar');
17 |             $scope.nomor = camForm.variableManager.variableValue('nomor');
18 |
19 |         });
20 |     </script>
21 |     <table>
22 |
23 |     <tr>
24 |         <td>Uang Daftar:</td>
25 |         <td>{{ uangDaftar }}</td>
26 |     </tr>
27 |
28 |     <tr>
29 |         <td>Nomor:</td>
30 |         <td>{{ nomor }}</td>
31 |     </tr>
32 |     </table>
33 | </form>
34 | </body>
35 | </html>

```

Listing C.13: ringkasan-jadwal.html

```

1 <html>
2
3 <head></head>
4 <script>
5     function printDiv(divName){
6         var printContents = document.getElementById(divName).innerHTML;
7         var originalContents = document.body.innerHTML;
8
9         document.body.innerHTML = printContents;
10
11         window.print();
12
13         document.body.innerHTML = originalContents;
14         window.close();
15
16     }
17
18
19 </script>
20
21
22
23 <body>
24 <form role="form" name="form">
25     <script cam-script type="text/form-script">
26         camForm.on('form-loaded', function() {
27             camForm.variableManager.fetchVariable('nomor');
28             camForm.variableManager.fetchVariable('jadwal');
29
30         });
31         camForm.on('variables-restored', function() {
32             $scope.nomor = camForm.variableManager.variableValue('nomor');
33             $scope.jadwal = camForm.variableManager.variableValue('jadwal');
34
35         });
36     </script>
37 <div id="printableArea">
38     <table>
39
40     <tr>
41         <td>Nomor :</td>
42         <td>{{ nomor }}</td>
43     </tr>
44
45     <tr>
46         <td>jadwal :</td>
47         <td>{{ jadwal }}</td>
48     </tr>
49
50     </table>
51 </div>
52 </form>
53 <input type="button" onclick="printDiv('printableArea')" value="Cetak Jadwal"/>
54 </body>
55 </html>

```

Listing C.14: ringkasa-pembayaran.html

```

1 <html>
2
3 <head></head>
4
5
6
7 <body>
8 <form role="form" name="form">
9     <script cam-script type="text/form-script">
10         camForm.on('form-loaded', function() {
11             camForm.variableManager.fetchVariable('nomor');
12             camForm.variableManager.fetchVariable('uangDaftar');
13
14         });
15         camForm.on('variables-restored', function() {
16             $scope.nomor = camForm.variableManager.variableValue('nomor');
17             $scope.uangDaftar = camForm.variableManager.variableValue('uangDaftar');
18
19         });
20     </script>
21     <table>
22
23     <tr>
24         <td>Nomor :</td>
25         <td>{{ nomor }}</td>

```

```

26     </tr>
27
28     <tr>
29         <td>Uang Daftar:</td>
30         <td>{{ uangDaftar }}</td>
31     </tr>
32 </table>
33 </form>
34
35 </body>
36 </html>

```

Listing C.15: verifikasi-pendaftaran.html

```

1 <html>
2
3 <head></head>
4
5
6
7 <body>
8 <form role="form" name="form">
9     <script cam-script type="text/form-script">
10         camForm.on('form-loaded', function() {
11             camForm.variableManager.fetchVariable('nama');
12             camForm.variableManager.fetchVariable('tanggalLahir');
13             camForm.variableManager.fetchVariable('nik');
14             camForm.variableManager.fetchVariable('kelas');
15             camForm.variableManager.fetchVariable('jadwal');
16             camForm.variableManager.fetchVariable('nomor');
17
18         });
19         camForm.on('variables-restored', function() {
20             $scope.nama = camForm.variableManager.variableValue('nama');
21             $scope.tanggalLahir = camForm.variableManager.variableValue('tanggalLahir');
22             $scope.nik = camForm.variableManager.variableValue('nik');
23             $scope.kelas = camForm.variableManager.variableValue('kelas');
24             $scope.nomor = camForm.variableManager.variableValue('nomor');
25             $scope.jadwal = camForm.variableManager.variableValue('jadwal');
26
27         });
28     </script>
29     <p>Download Dokumen Persyaratan</p>
30     <a cam-file -download="INVOICE_DOCUMENT">Download</a>
31     <table>
32
33
34
35     <tr>
36         <td>Nama:</td>
37         <td>{{ nama }}</td>
38     </tr>
39
40     <tr>
41         <td>Tanggal Lahir:</td>
42         <td>{{ tanggalLahir }}</td>
43     </tr>
44
45     <tr>
46         <td>Nik:</td>
47         <td>{{ nik }}</td>
48     </tr>
49
50     <tr>
51         <td>Kelas:</td>
52         <td>{{ kelas }}</td>
53     </tr>
54
55     <tr>
56         <td>Nomor:</td>
57         <td>{{ nomor }}</td>
58     </tr>
59
60     <tr>
61         <td>jadwal:</td>
62         <td>{{ jadwal }}</td>
63     </tr>
64 </table>
65
66     <p>Apakah dokumen dan persyaratan lengkap?</p>
67     <input cam-variable -name="valid"
68         cam-variable-type="Boolean"
69         type="checkbox"
70         name="valid"
71         class="form-control" />

```

```

72
73 </form>
74 </body>
75 </html>

```

Listing C.16: cetak-kartu.html

```

1 <html>
2
3 <head></head>
4 <script>
5     function printDiv(divName){
6         var printContents = document.getElementById(divName).innerHTML;
7         var originalContents = document.body.innerHTML;
8
9         document.body.innerHTML = printContents;
10
11        window.print();
12
13        document.body.innerHTML = originalContents;
14        window.close();
15
16    }
17
18
19
20 </script>
21
22 <body>
23 <form role="form" name="form">
24     <script cam-script type="text/form-script">
25         camForm.on('form-loaded', function() {
26             camForm.variableManager.fetchVariable('nama');
27             camForm.variableManager.fetchVariable('tanggalLahir');
28             camForm.variableManager.fetchVariable('nik');
29             camForm.variableManager.fetchVariable('kelas');
30             camForm.variableManager.fetchVariable('barcode');
31
32         });
33         camForm.on('variables-restored', function() {
34             $scope.nama = camForm.variableManager.variableValue('nama');
35             $scope.tanggalLahir = camForm.variableManager.variableValue('tanggalLahir');
36             $scope.nik = camForm.variableManager.variableValue('nik');
37             $scope.kelas = camForm.variableManager.variableValue('kelas');
38             $scope.barcode = camForm.variableManager.variableValue('barcode');
39
40         });
41     </script>
42     <div id="printableArea">
43     <table>
44
45
46
47     <tr>
48         <td>Nama:</td>
49         <td>{{ nama }}</td>
50     </tr>
51
52     <tr>
53         <td>Tanggal Lahir:</td>
54         <td>{{ tanggalLahir }}</td>
55     </tr>
56
57     <tr>
58         <td>Nik:</td>
59         <td>{{ nik }}</td>
60     </tr>
61
62     <tr>
63         <td>Kelas:</td>
64         <td>{{ kelas }}</td>
65     </tr>
66
67     <tr>
68         <td>Barcode:</td>
69         <td>{{ barcode }}</td>
70     </tr>
71
72     </table>
73
74
75 </form>
76 </div>
77     <input type="button" onclick="printDiv('printableArea')" value="Cetak_Kartu"/>
78 </body>

```

79|</html>