# SKRIPSI

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

LUCKY SENJAYA DARMAWAN

NPM: 2012730009

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

UNDERGRADUATE THESIS

# STUDY AND WORKFLOW INTEGRATION USING BPMS AND EMAIL SYSTEM

LUCKY SENJAYA DARMAWAN

NPM: 2012730009

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

## ABSTRAK

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

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



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

**ABSTRACT**

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

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

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

# DAFTAR ISI

# DAFTAR GAMBAR

# 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)*, seperti Camunda. Agar eksekusi *workflow* lebih alamiah dengan model komunikasi organisasi saat ini, maka *event* 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 efisien.

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 mengirimkan email ke pengguna yang akan mengerjakan task tersebut. Email tersebut akan berisi tautan yang mengarah ke tugas yang perlu dikerjakan tersebut. Untuk mencapainya, dibuat sebuah *listener* yang dikaitkan di *event*. *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 memodelkan *workflow* dengan BPMN?
2. Event-event *workflow* apa saja yang dapat dipropagasi ke sistem email?
3. Bagaimana mekanisme propagasi dan integrasi *workflow* dengan sistem email?
4. Bagaimana mengimplementasikan dan menguji integrasi *workflow* dengan sistem email?

## 1.3 Tujuan

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

1. Memodelkan *workflow* dengan BPMN.

2. Mengidentifikasi event-event *workflow* yang dapat dipropagasi ke sistem email.
3. Menentukan mekanisme propagasi dan mengintegrasikan *workflow* dengan sistem email.
4. 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 saat ini.
2. Perangkat lunak BPMS Camunda yang digunakan merupakan versi 7.6.0 dan berjalan pada tomcat versi 8.0.24, yaitu versi terbaru untuk saat ini.
3. Semua uji kasus berada di lingkungan Camunda. Hal ini dilakukan agar skripsi ini lebih fokus kepada integrasi email.

## 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. Mengidentifikasikan *event-event* dari *workflow* yang dapat diintegrasikan dengan sistem email.
4. Merancang integrasi sistem email.
5. Mengimplementasikan sistem email ke BPMS.
6. Melakukan pengujian fungsionalitas.

## 1.6 Sistematika Pembahasan

1. Bab 1 Pendahuluan, berisi latar belakang masalah, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian, dan sistematika penulisan.
2. Bab 2 Dasar Teori, berisi dasar teori yang mencakup *Business Process Management*, *Business Process Model and Notation (BPMN)*, *Business Process Management System (BPMS)*, dan sistem e-mail.
3. Bab 3 Analisis, Berisi analisis BPMN dengan menggunakan skenario, analisis event yang terkait dengan sistem email dan mekanisme integrasi sistem email.
4. Bab 4 Perancangan, Berisi rancangan sistem dan rancangan partisipan dalam otomasi BPMS Camunda, .

5. Bab 5 Implementasi, dan Pengujian Berisi implementasi dari program yang dibuat dan pengujian aplikasi berdasarkan contoh kasus pada bab tiga.

6. Bab 6 Penutup, Berisi kesimpulan dan saran-saran untuk pengembangan selanjutnya.

# BAB 2

# DASAR TEORI

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

## 2.1 *Business Process Management* (BPM)

*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[1] . Dari definisi proses bisnis, *Business Process Management* dapat didefinisikan sebagai kumpulan metode, teknik, dan alat untuk menemukan, menganalisa, mendesain kembali, menjalankan, dan mengawasi proses bisnis.

### 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*.

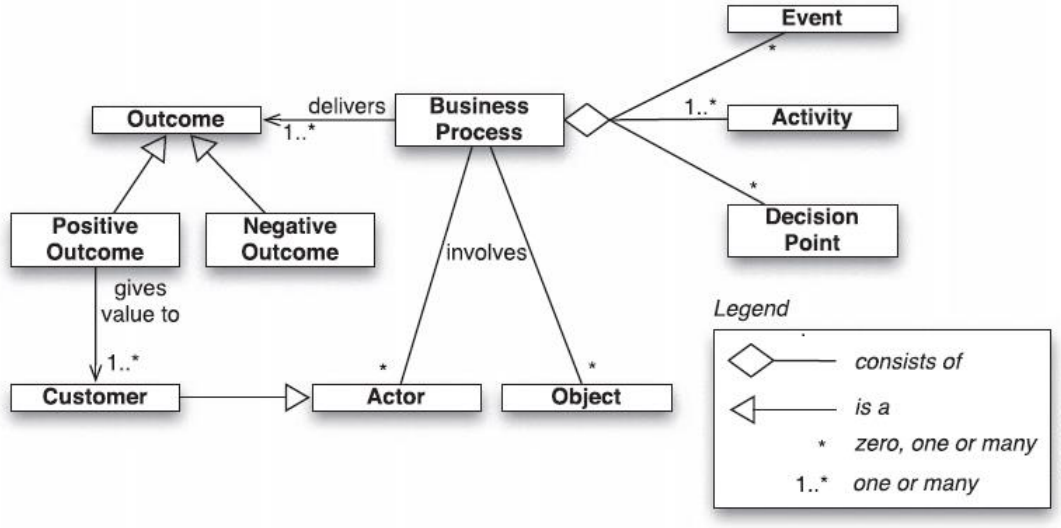
*Decision Point*
: *Decision point* adalah keputusan yang mempengaruhi proses selanjutnya.

*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

### 2.1.2 Siklus *Business Process Management*

Suatu proses bisnis tidak selalu berjalan dengan baik. Banyak hal yang tidak diantisipasi sebelumnya dapat menggangu 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*.

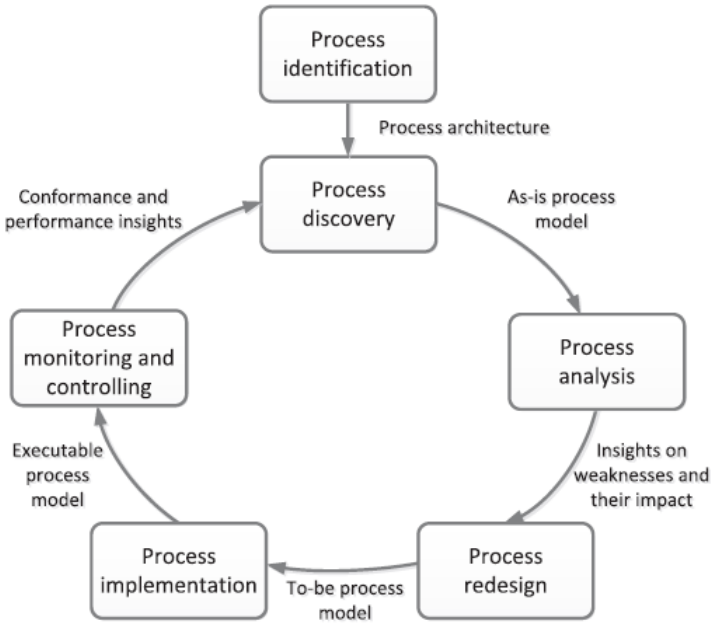
*Process Implementation*

Pada fase ini, model proses diimplementasikan untuk diekseskusi 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

## 2.2 *Business Process Model 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*.

### 2.2.1 *Event*

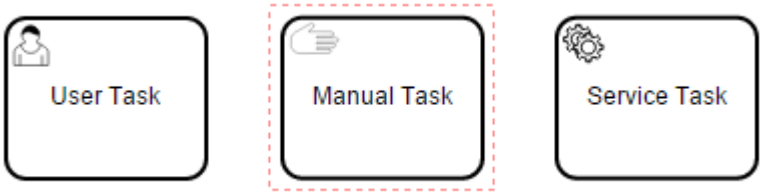
Event merupakan kejadian yang terjadi pada proses bisnis yang dilambangkan dengan bentuk lingkaran. Notasi event secara umum terbagi menjadi tiga, yaitu *start event, intermediate event,* dan *end event*. *Start event* menunjukkan dimulainya proses, *intermediate event* dapat muncul ketika proses berjalan,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


Gambar 2.3: Notasi *Event*

### 2.2.2 *Activity*

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

1. *User Task*, yaitu pekerjaan yang perlu dilakukan oleh manusia melalui sistem.
2. *Manual Task*, yaitu pekerjaan yang dilakukan manusia tanpa melalui sistem.
3. *Service Task*, yaitu pekerjaan yang dilakukan oleh sistem dengan memangggil kode Java.

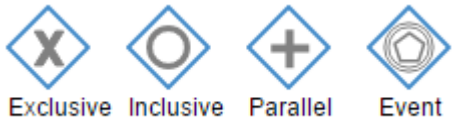



Gambar 2.4: Notasi *Task*

### 2.2.3 *Gateway*

*Gateway* merupakan simbol yang menentukan percabangan dan penggabungan jalur dalam proses. Gateway dilambangakan 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*

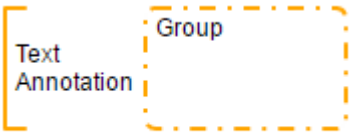
### 2.2.4 *Data*

*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*

### 2.2.5 *Artifact*

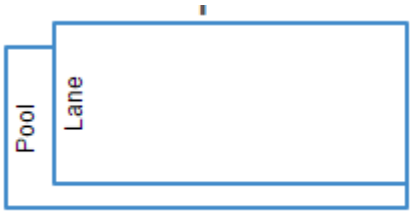
*Artifact* tidak mempengaruhi jalannya proses, tetapi hanya sebagai informasi tambahan agar proses lebih mudah dimengerti. Terdapat dua jenis, yaitu *Text Annotation* dan *Group*




Gambar 2.7: Notasi *Artifact*

### 2.2.6 *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*

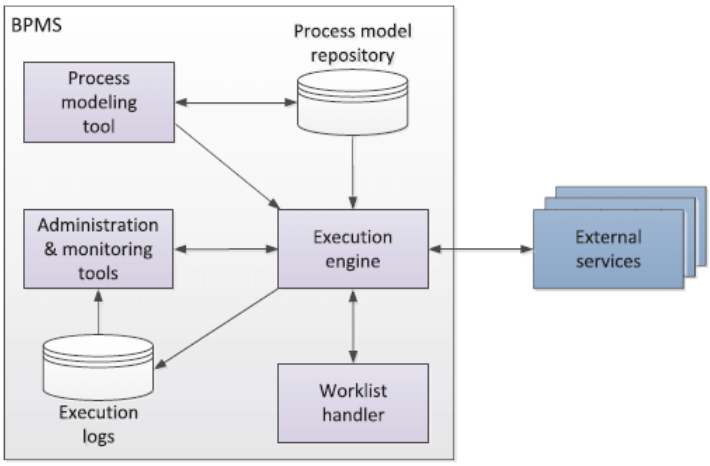
## 2.3 *Business Process Management System (BPMS)*

*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.



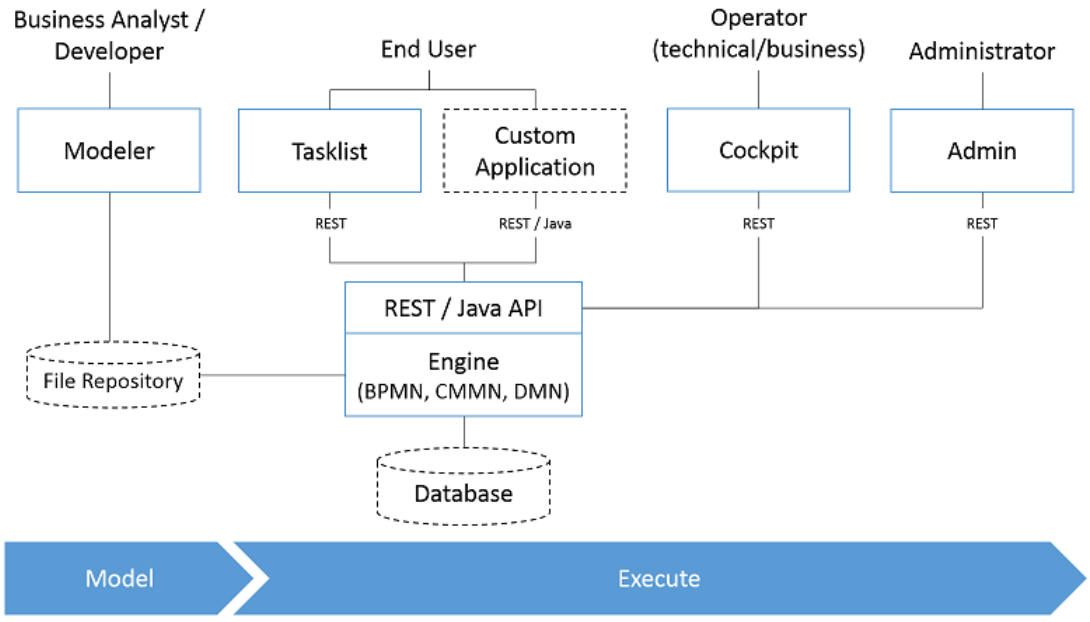

Gambar 2.9: Arsitektur BPMS

## 2.4 BPMS Camunda

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

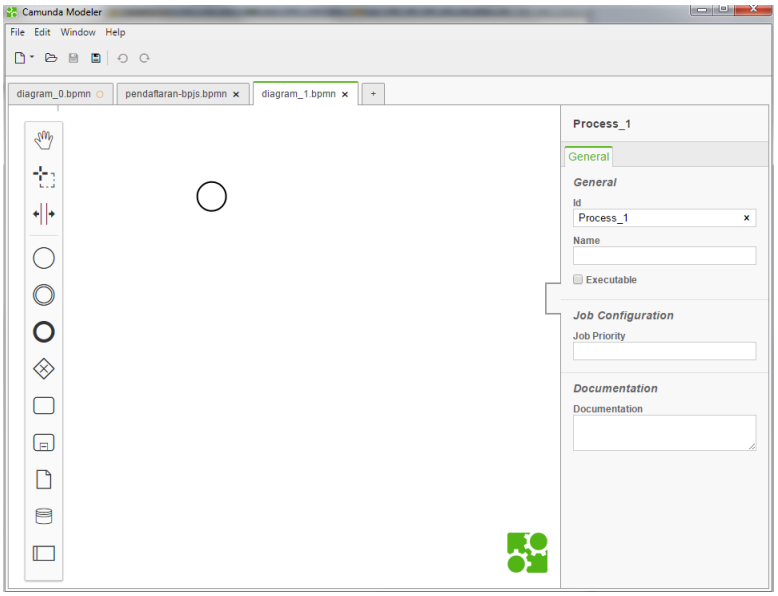
### 2.4.1 Arsitektur BPMS Camunda

Komponen-komponen pada BPMS Camunda adalah sebagai berikut :



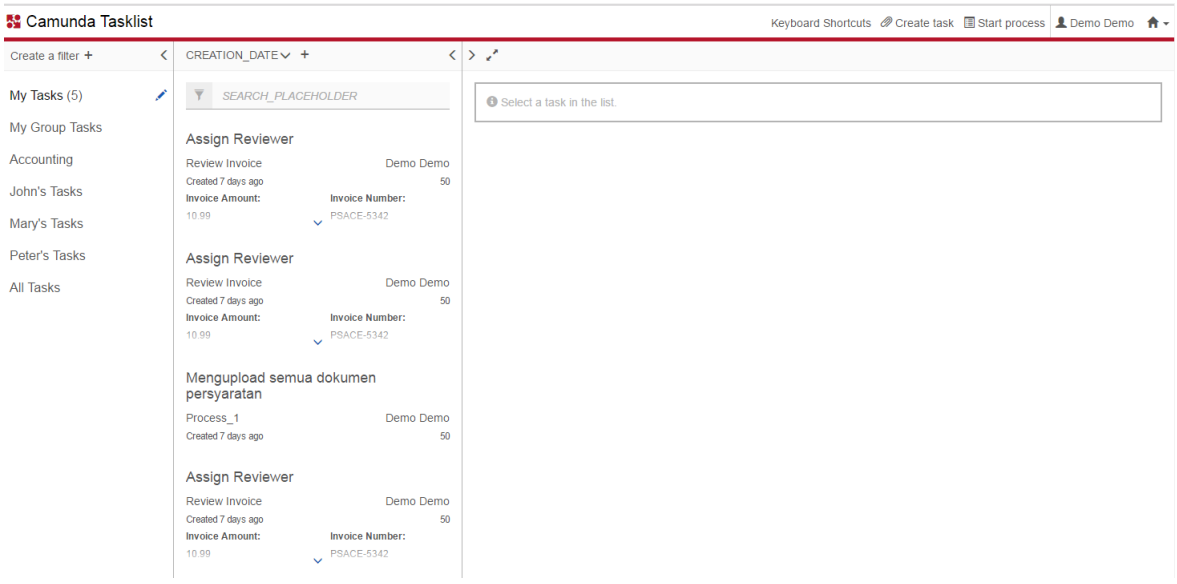

Gambar 2.10: Arsitektur BPMS Camunda

- *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.



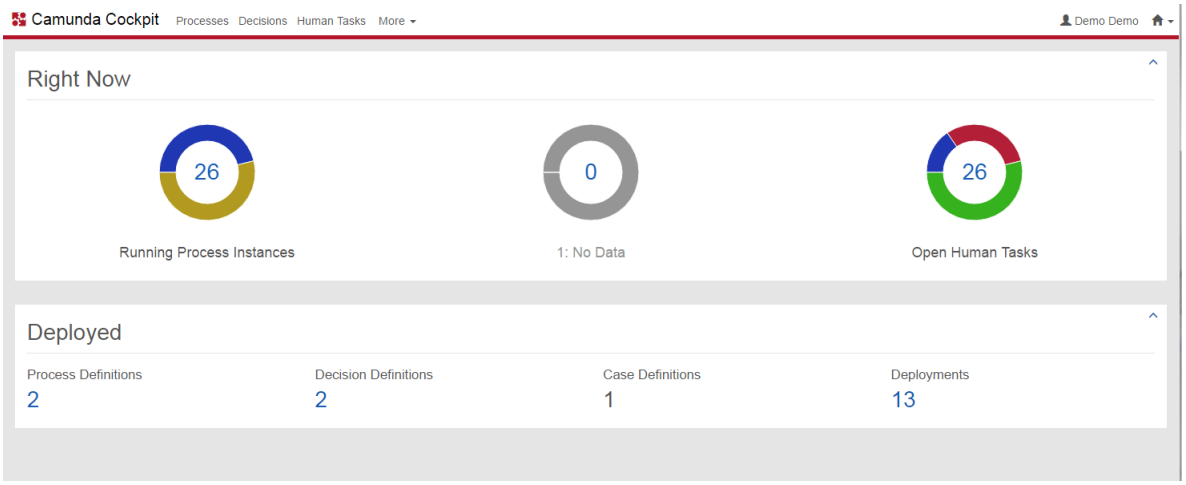

Gambar 2.11: Camunda Modeler

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



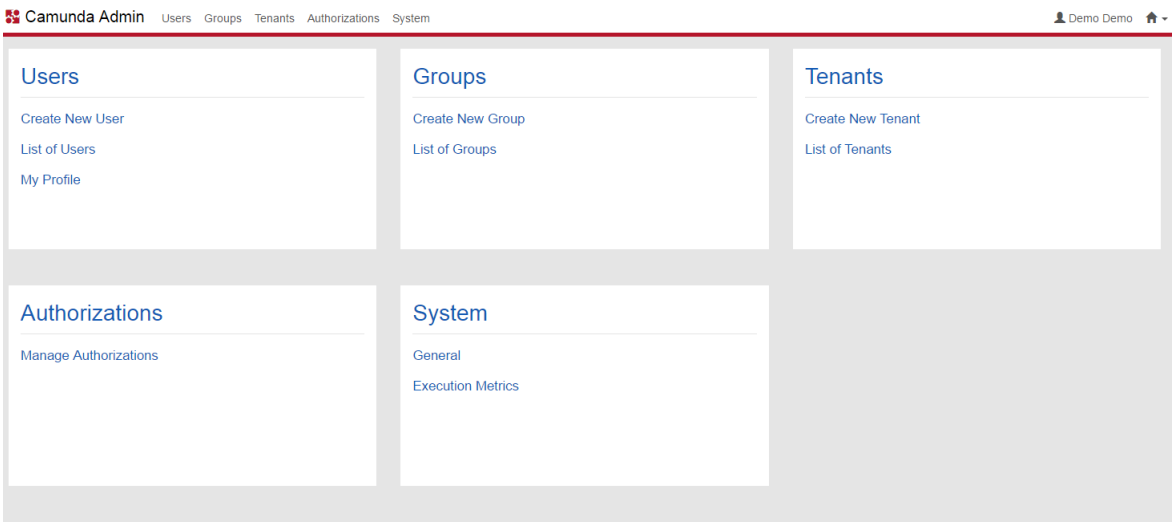

Gambar 2.12: Camunda Tasklist

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




Gambar 2.13: Camunda Cockpit

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




Gambar 2.14: Camunda Admin

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

### 2.4.2 Pengoperasian Camunda

#### Instalasi

Untuk menjalankan Camunda, diperlukan beberapa *tool*[4], 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. Nama kelas dapat diganti dengan nama proses yang dibuat.

Listing 2.1: 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("Loan Approval App")
7 public class LoanApprovalApplication extends ServletProcessApplication {
8   // empty implementation
9 }
```

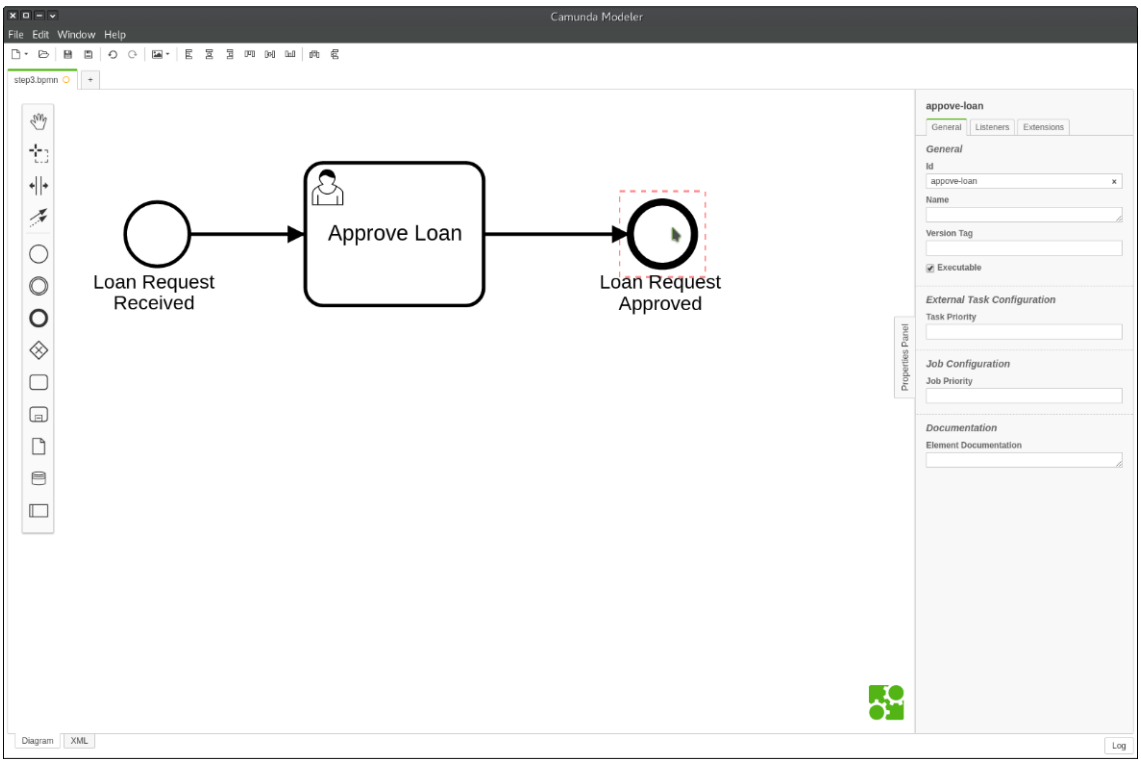
Tambahkan *Deployment Descriptor* di META-INF/processes.xml.

Listing 2.2: 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>
```

### Memodelkan Proses

1. Membuat file BPMN baru dengan File / New File / BPMN Diagram.
2. Memodelkan proses.




Gambar 2.15: Contoh Model Proses

3. Menambahkan Form HTML pada User Task.

Listing 2.3: Contoh Task Form

```
        <form name="approveLoan">
  <div class="form-group">
    <label for="customerId">Customer ID</label>
    <input class="form-control"
           cam-variable-type="String"
           cam-variable-name="customerId"
           name="customerId"
           readonly="true" />
  </div>
  <div class="form-group">
    <label for="amount">Amount</label>
    <input class="form-control"
           cam-variable-type="Double"
           cam-variable-name="amount"
           name ="amount" />
  </div>
</form>
```

4. Menambahkan Service Task.

Listing 2.4: Contoh Implementasi Service Task

```
        package org.camunda.bpm.getstarted.loanapproval;

import java.util.logging.Logger;
import org.camunda.bpm.engine.delegate.DelegateExecution;
import org.camunda.bpm.engine.delegate.JavaDelegate;

public class ProcessRequestDelegate implements JavaDelegate {

  private final static Logger LOGGER = Logger.getLogger("LOAN-REQUESTS");

  public void execute(DelegateExecution execution) throws Exception {
    LOGGER.info("Processing request by '"+execution.getVariable("customerId")+"'...");
  }

}
```

**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

## 2.5 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)[5]. Untuk menggunakan JavaMail diperlukan beberapa kelas, yaitu :

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

Listing 2.5: 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

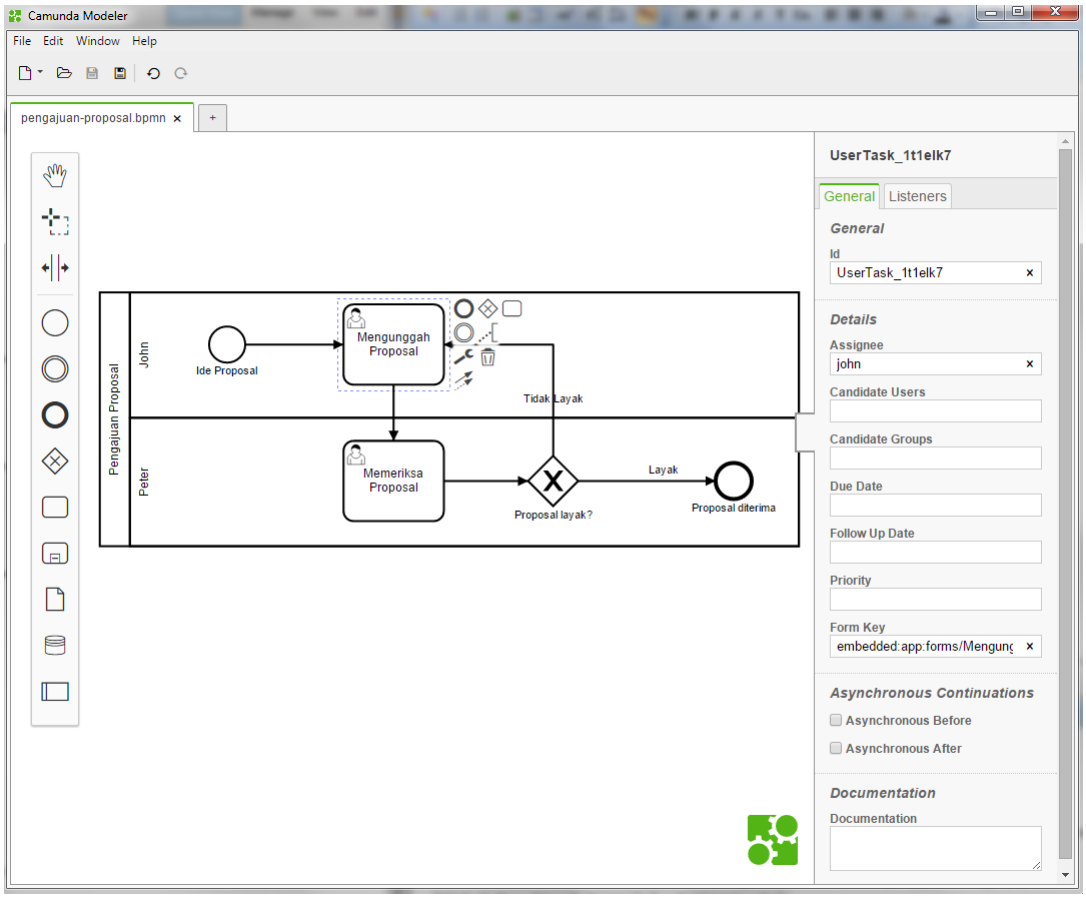
# ANALISIS

Bab ini berisi analisis BPMN dengan menggunakan skenario, analisis *event* yang terkait dengan integrasi sistem email, dan mekanisme integrasi sistem email.

## 3.1 Analisis BPMN

### 3.1.1 Skenario Proposal Bisnis

John mempunyai ide proposal bisnis untuk manajernya, Peter. John menulis dan mengunggah proposal melalui sistem Camunda. Sebelum proposal disetujui, Peter harus memeriksa apakah proposalnya layak atau tidak. Jika proposalnya tidak layak, John harus memperbaiki dan mengunggahnya kembali. *Workflow* dari skenario ini sebagai berikut :



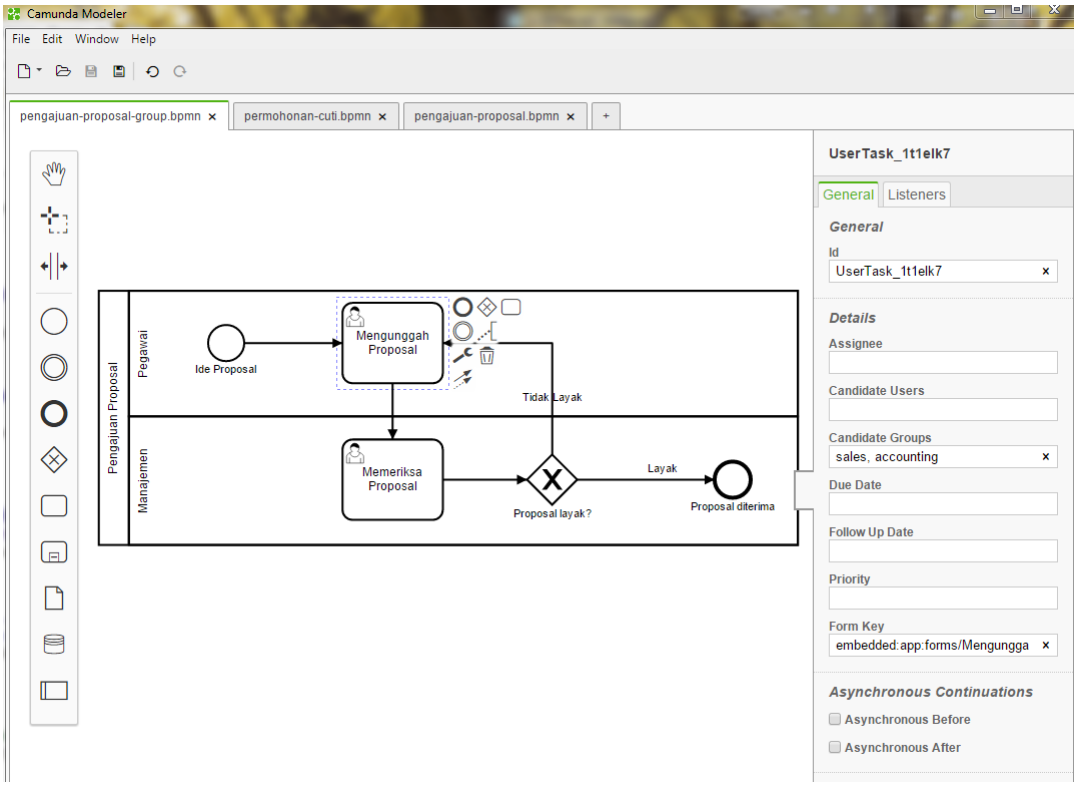

Gambar 3.1: Mengunggah Proposal

Pada Gambar 3.1, terdapat beberapa atribut yang memiliki nilai, yaitu :

- Id, yaitu id dari *task* yang dipilih
- Assignee, yaitu aktor yang akan mengerjakan *task*
- Form Key, yaitu tautan ke file HTML yang berupa tampilan untuk mengunggah proposal.

### 3.1.2 Skenario Proposal Bisnis dari Group

Pegawai di perusahaan X memiliki tiga divisi yaitu *accounting*, *sales*, dan *management*. Divisi *accounting* dan *sales* dapat mengajukan proposal bisnis ke divisi *management*. Sama seperti Skenario 3.1.1, divisi *management* harus memeriksa apakah proposalnya layak atau tidak. Jika proposalnya tidak layak, pembuat proposal harus memperbaiki dan mengunggahnya kembali. Workflow dari skenario ini sebagai berikut :




Gambar 3.2: Mengunggah Proposal

Pada Gambar 3.2, terdapat atribut *Candidate Groups*. Atribut ini melambangkan bahwa *task* ini dapat dikerjakan oleh salah satu anggota dari grup *accounting* atau grup *sales*.

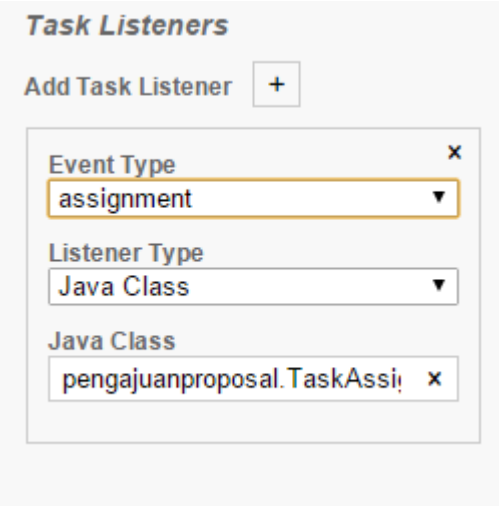
## 3.2 *Event* yang Terkait dengan Integrasi Sistem Email

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*.

## 3.3 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*.




Gambar 3.3: 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.

# BAB 4

# PERANCANGAN

Untuk mempropagasi email, diperlukan perancangan sistem dan beberapa peran yang harus dilakukan oleh partisipan.

## 4.1 Perancangan Sistem

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

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

### 4.1.1 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*).

### 4.1.2 Algoritma Pengiriman Email

Berikut adalah algoritma untuk mengirimkan email.

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

## 4.2 Peran Partisipan

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

### 4.2.1 Tugas Desainer

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

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

### 4.2.2 Tugas Admin

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

### 4.2.3 Tugas Aktor

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

### 4.2.4 Perancangan Aktor

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

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

# BAB 5

# IMPLEMENTASI DAN PENGUJIAN

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

### 5.2.1 Implementasi Algoritma Pengiriman Email

Beberapa potongan kode di bawah ini adalah kode untuk pengiriman email. Kode secara keseluruhan dapat dilihat pada Lampiran A

- Konfigurasi email admin.

Listing 5.1: TaskAssignmentListener.java

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

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

Listing 5.2: TaskAssignmentListener.java

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

- Konfigurasi SMTP Gmail.

Listing 5.3: TaskAssignmentListener.java

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

- Kode untuk mendapatkan aktor apabila atribut assignee memiliki nilai.

Listing 5.4: TaskAssignmentListener.java

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

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

Listing 5.5: TaskAssignmentListener.java

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

- Kode untuk membangkitkan subjek dan isi email

Listing 5.6: TaskAssignmentListener.java

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

- Kode untuk mengirimkan email.

Listing 5.7: TaskAssignmentListener.java

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

### 5.2.2 Implementasi Skenario

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

Kode ini adalah kode file HTML untuk Skenario 3.1.1 dan Skenario 3.1.2. Terdapat dua form HTML yaitu MengunggahDokumen.html dan MemeriksaDokumen.html.

Listing 5.8: MengunggahDokumen.html

```
<html>
<head>
<body>
    <form method="post" name="upload-dokumen">
        <input type="file"
        cam-variable-name="proposal"
        cam-variable-type="File"
        cam-max-filesize="10000000"/>
    </form>
</body>
</html>
```

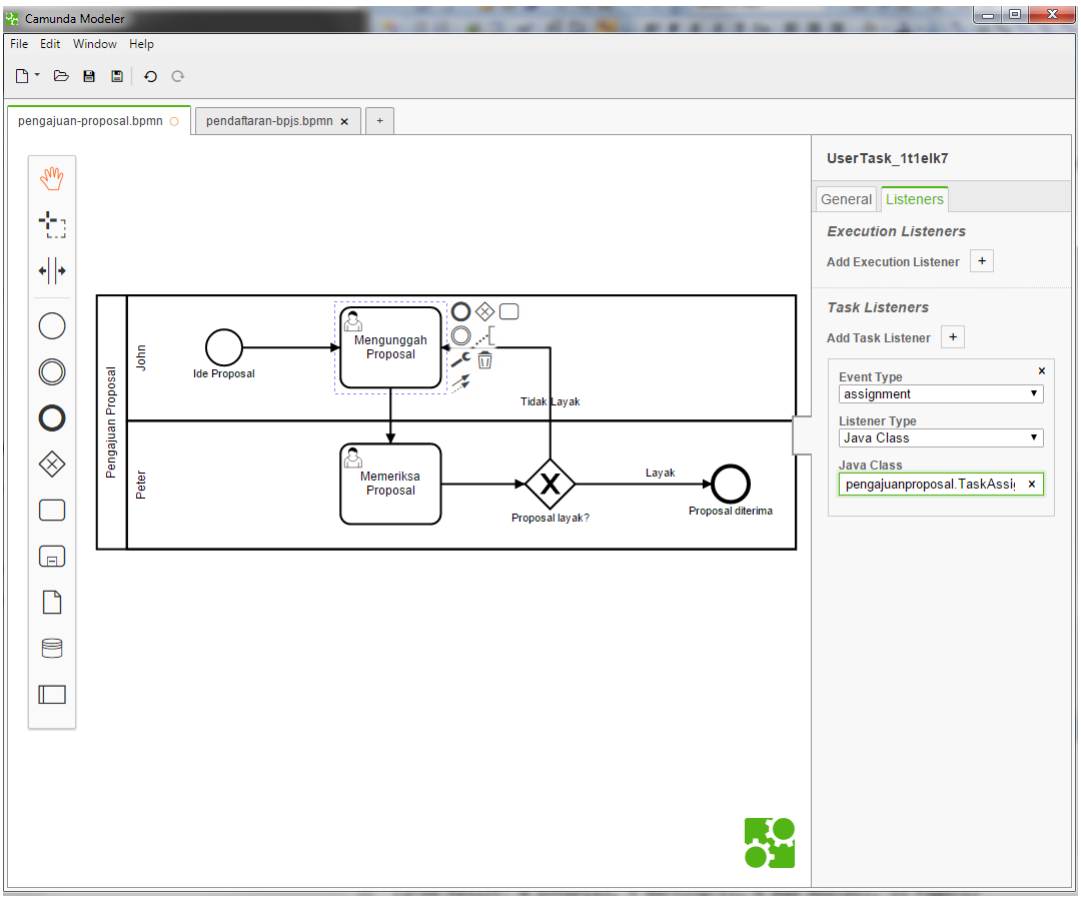
Listing 5.9: MemeriksaDokumen.html

```


<html>
<head></head>

<body>
<form role="form" name="form">
    <a cam-file-download="proposal">Download Dokumen</a>
    <p>Apakah Proposal layak?</p>
    <input cam-variable-name="valid"
           cam-variable-type="Boolean"
           type="checkbox"
           name="valid"
           class="form-control" />
</form>
</body>
</html>
```

## 5.3 Pengujian

### 5.3.1 Pengujian Skenario 1

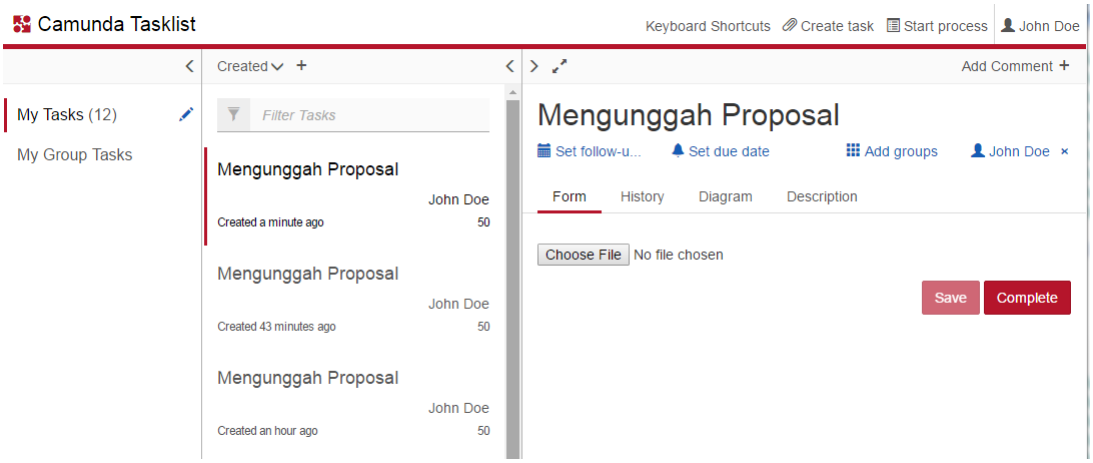
labelujiskenario1 Pada Skenario 3.1.1, ditambahkan form HTML dan Task Listener untuk mengirimkan email.




Gambar 5.1: Mengunggah Proposal

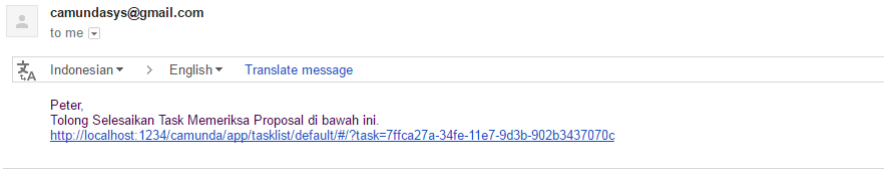
Proses pengujian :

- John mengunggah dokumen proposal



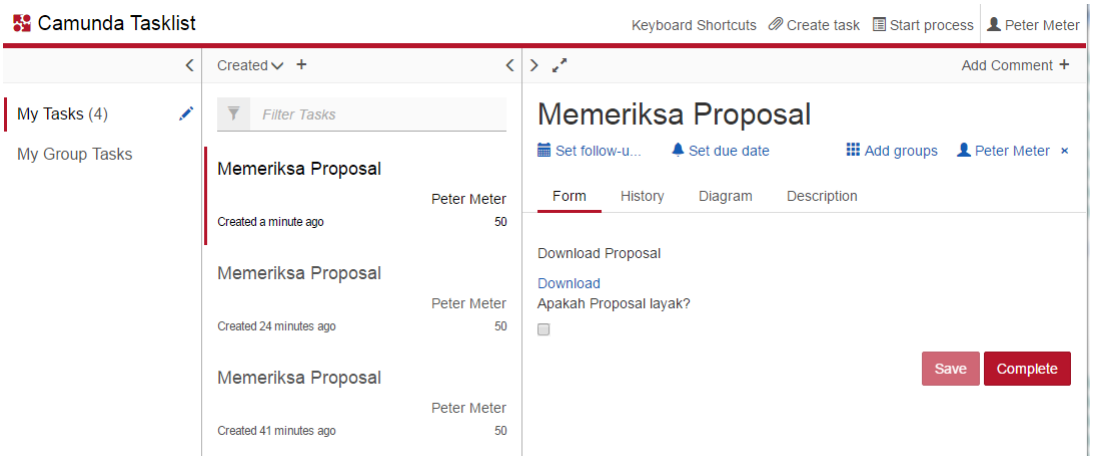

Gambar 5.2: Mengunggah Proposal

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




Gambar 5.3: Menerima Email

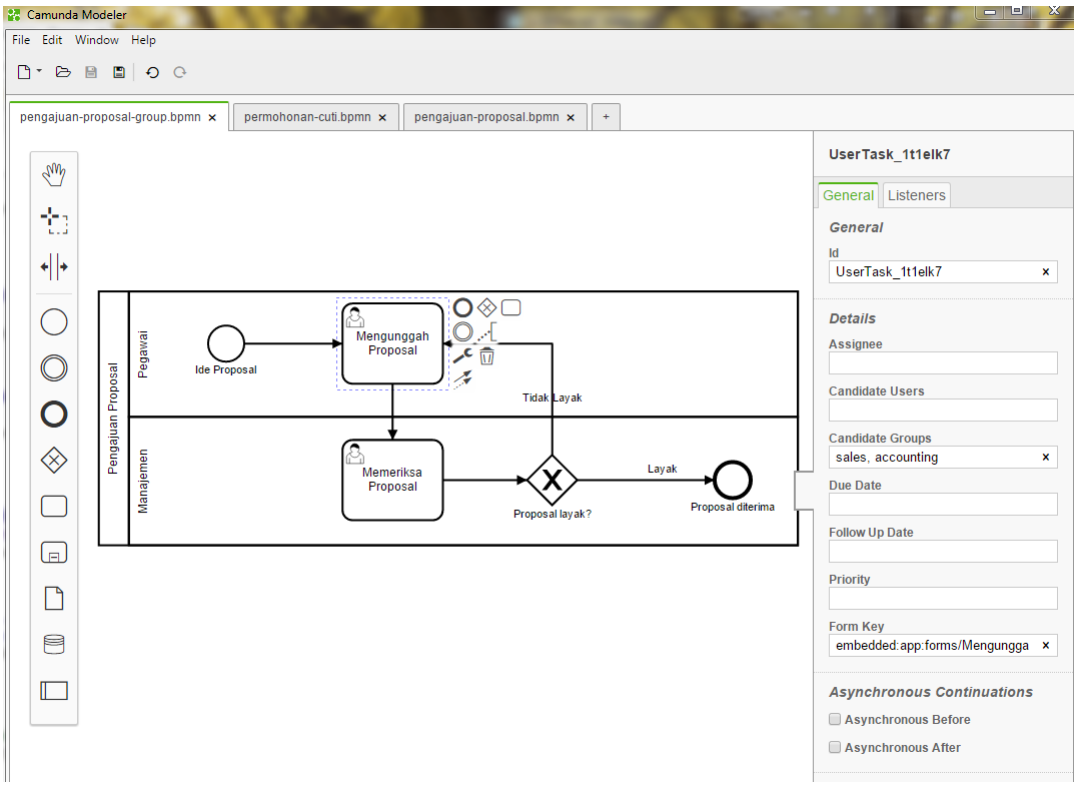
- Tautan email membawa Peter ke *task* yang harus dikerjakan.




Gambar 5.4: Tasklist Peter

### 5.3.2 Pengujian Skenario 2

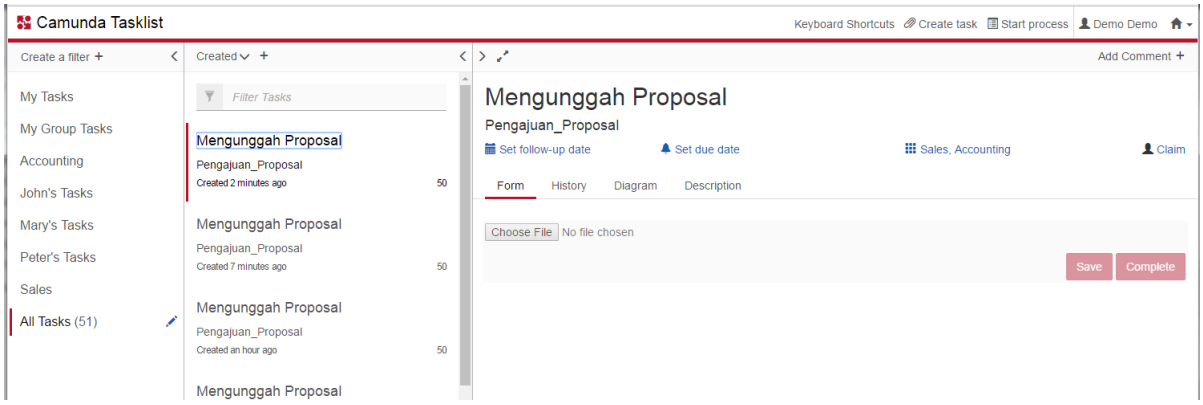
Pada Skenario 3.1.2, ditambahkan form HTML dan Task Listener untuk mengirimkan email. Perbedaan dengan Skenario 3.1.1 adalah *task* didelegasikan ke grup *accounting*, *sales*, dan *management*. John adalah bagian dari divisi *sales*, Mary adalah bagian dari divisi *accounting*, sedangkan Peter adalah bagian dari divisi *management*.




Gambar 5.5: Mengunggah Proposal Group

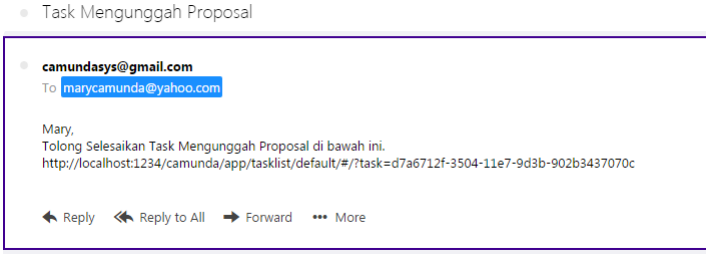
Proses pengujian :

- Admin memulai proses

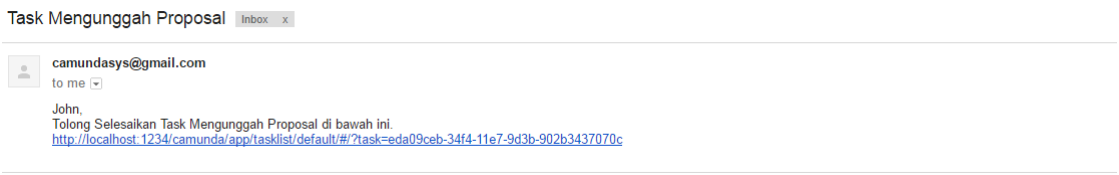



Gambar 5.6: John Mengunggah Proposal

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

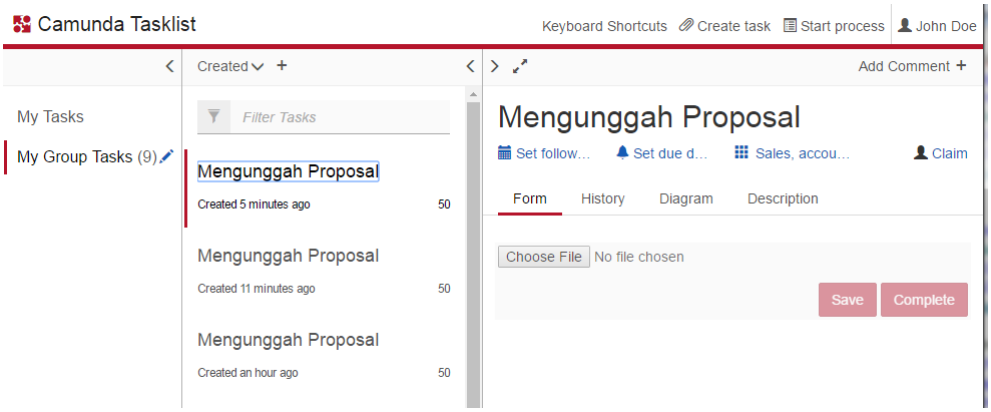



Gambar 5.7: Mary Mendapat Email

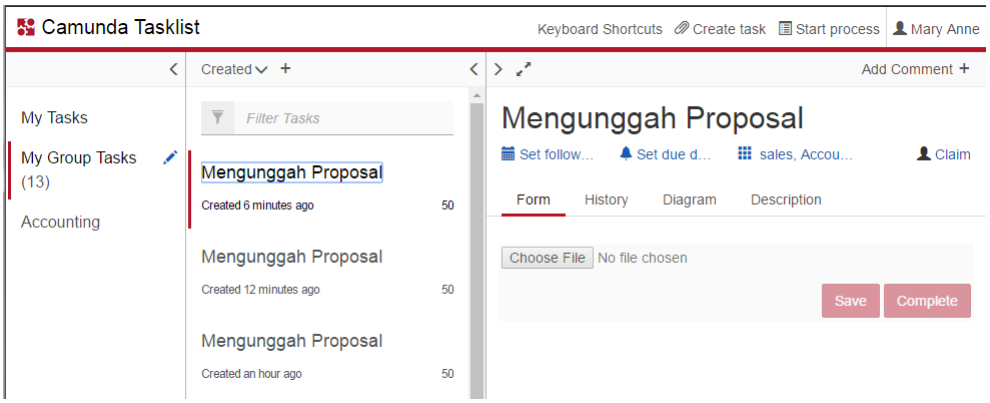



Gambar 5.8: John Mendapat Email

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

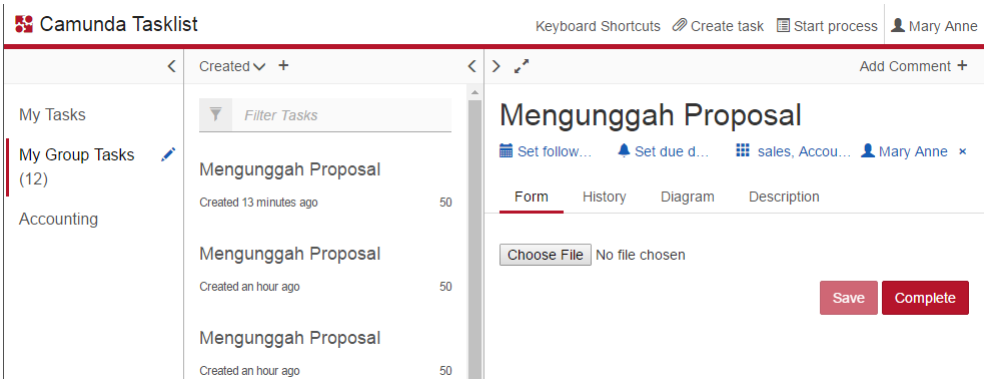



Gambar 5.9: John Mengunggah Proposal



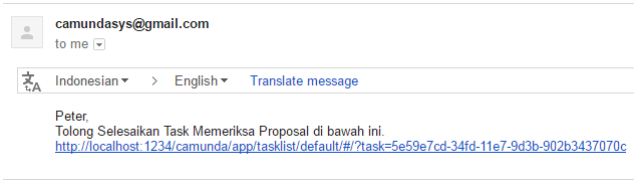

Gambar 5.10: Mary Mengunggah Proposal

- Mary mengklaim task dan mengerjakan *task*



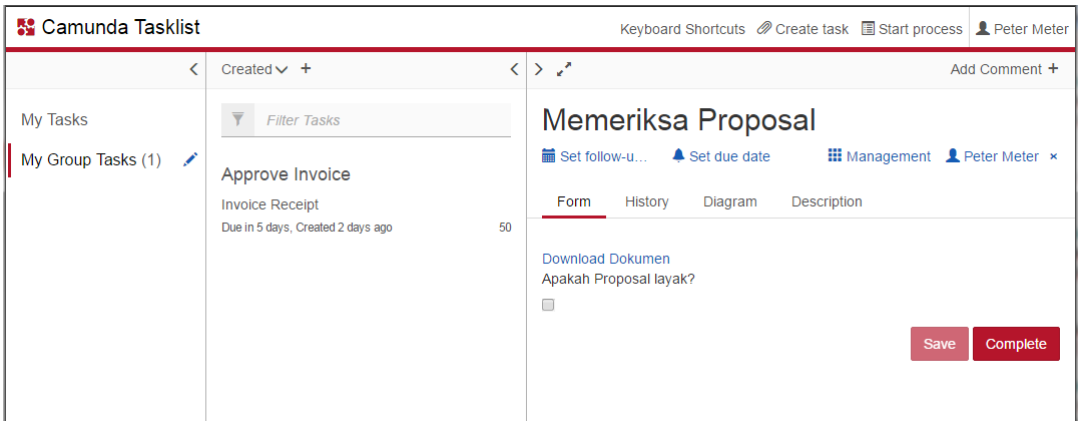

Gambar 5.11: Mary mengklaim Task

- Peter mendapatkan email untuk mengerjakan *task*.



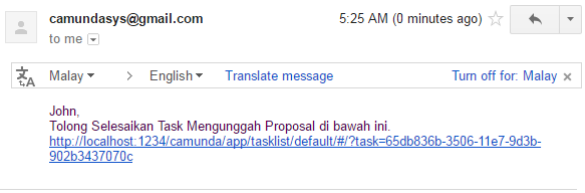

Gambar 5.12: Mary mengklaim Task

- Peter mengklaim task dan menolak proposal

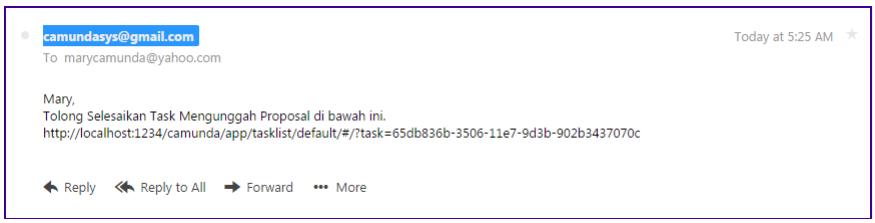



Gambar 5.13: Peter mengklaim Task

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




Gambar 5.14: John mendapatkan Email




Gambar 5.15: Mary mendapatkan Email

## 5.4 Hasil Pengujian

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

# BAB 6

# KESIMPULAN DAN SARAN

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

## 6.1 Kesimpulan

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

1. *Workflow* dapat dimodelkan sebagai BPMN sehingga 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. *P*ropagasi email dapat dilakukan dengan cara menyisipkan *Task Listener* di event yang akan dipropagasi. Selain itu dibutuhkan peran admin untuk mendaftarkan alamat email aktor.
4. *P*engujian telah dilakukan dengan dua skenario dan dapat berjalan dengan baik.

## 6.2 Saran

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

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

# DAFTAR REFERENSI


[1] Dumas, M., Rosa, M. L., Mendling, J., dan Reijers, H. A. (2013) *Fundamentals of Business Process Management*. Springer-Verlag, Berlin.

[2] Camunda (2015) Bpmn modeling reference. https://camunda.org/bpmn/reference/.

[3] Version 7.6 (2015) *The Camunda BPM Manual*. Camunda BPM. Berlin, Germany.

[4] Camunda (2015) Get started with camunda and bpmn 2.0. https://docs.camunda.org/get-started/bpmn20/.

[5] Oracle Javamail. http://www.oracle.com/technetwork/java/javamail/index.html/.

[6] Google Use smtp settings to send mail from a printer, scanner, or app. https://support.google.com/a/answer/176600?hl=en.

# LAMPIRAN A

# KODE PROGRAM PENGIRIMAN EMAIL

Listing A.1: TaskAssignmentListener.java

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

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

# LAMPIRAN B

# KODE POM.XML

Listing B.1: pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.camunda.bpm.getstarted</groupId>
  <artifactId>loan-approval</artifactId>
  <version>0.1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.camunda.bpm</groupId>
        <artifactId>camunda-bom</artifactId>
        <version>7.6.0</version>
        <scope>import</scope>
        <type>pom</type>
      </dependency>
    </dependencies>
  </dependencyManagement>

  <dependencies>
    <dependency>
      <groupId>org.camunda.bpm</groupId>
      <artifactId>camunda-engine</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.0.1</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.3</version>
        <configuration>
          <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```