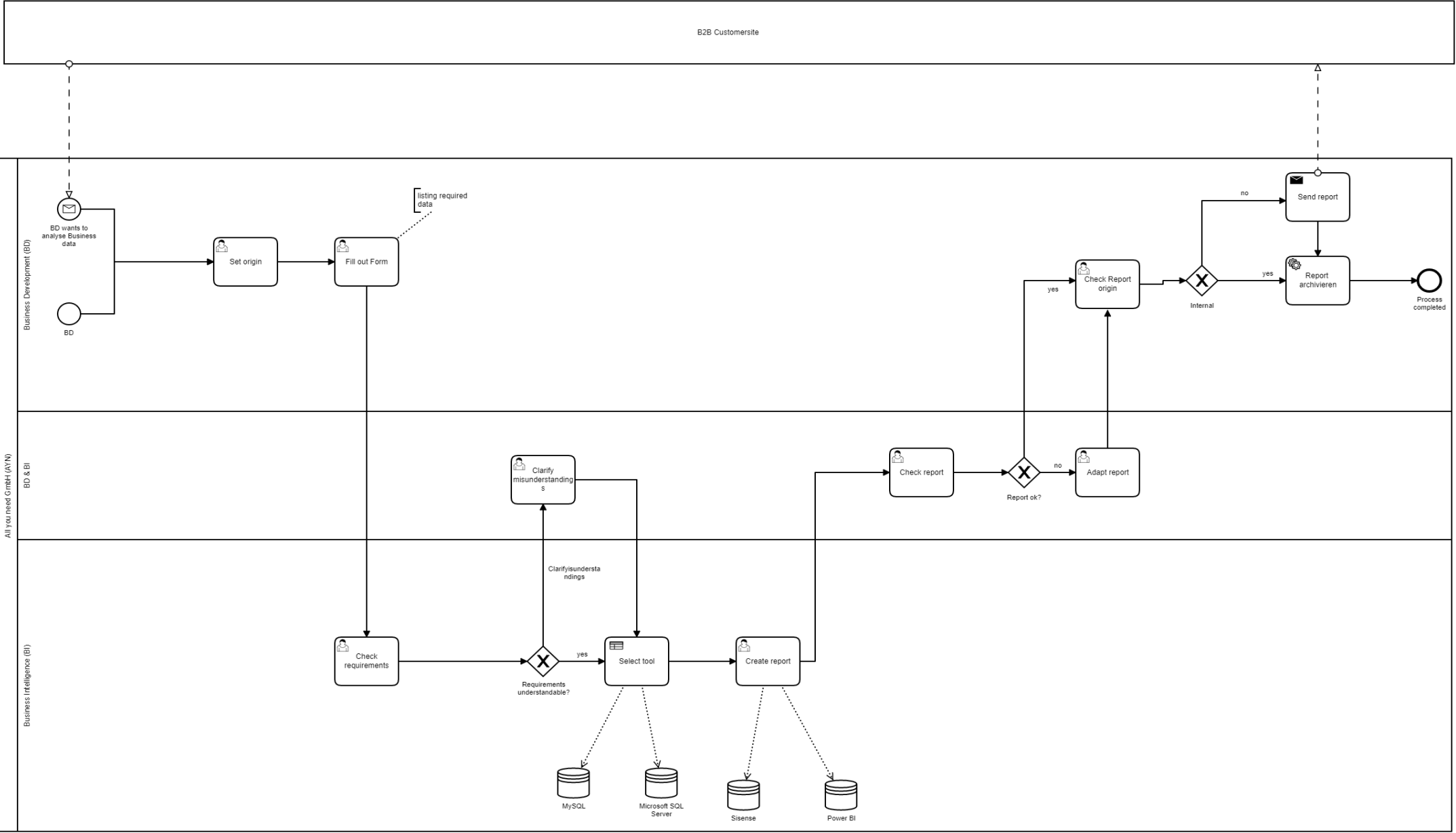


Business Development Report Dokumentation

1. Prozessbeschreibung BPMN

In der Abbildung 1 können Sie sehen, wie die „All you need GmbH“ verschiedene Business Development Reports für sich und seine Kunden erstellt.

Der Prozess beginnt entweder mit einer Anfrage der B2B Kundenseite oder mit einer Anfrage aus dem eigenen Unternehmen, um seine aktuellen Geschäftsdaten zu analysieren. Als nächstes wird von der Abteilung Business Developement (BD) ein Auftragsformular ausgefüllt. Das ausgefüllte Formular wird per E-Mail an die Abteilung Business Intelligence (BI) gesendet. Die Abteilung BI nimmt die Anfrage an und überprüft diese. Sollten die Anforderungen unklar sein, muss die Abteilung BI dies mit der Abteilung BD besprechen, um die Anforderungen zu klären. Die Abteilung BI entscheidet sich daraufhin für ein Reporting-Tool, womit es den Report erstellt. Der Report sollte von den zwei Abteilungen inhaltlich überprüft, ggf. angepasst und schließlich abgenommen werden. Sobald die Aufgabe von der Abteilung BD abgenommen wurde, wird der Report archiviert. Wenn es eine unternehmensinterne Anfrage war oder es wird eine Mail an die B2B Kundenseite geschickt und daraufhin archiviert. Sobald dies erledigt wurde ist der Prozess beendet.



2. Prozessbeschreibung DMN

Bevor der Mitarbeiter beginnt, den Report zu erstellen, muss noch ein Reporting-Tool ausgewählt werden. Diese Aktion entspricht der Decision Task „select reporting tool“ im BPMN-Modell. Er kann zwischen: Sisense (Im vorherigen Teil des Moduls „Softwareauswahl“ das ausgewählte Tool) und Microsoft SQL Server Reporting Tool (im Datenbankserver eingebettetes Tool) auswählen.

Die Input-Daten dieses DMN-Modells lauten: Endbenutzer, Datenquellen und Datenvisualisierung.

Endnutzer:

Datentyp: string

Variablen: „internal“, „B2B customer“

Datenquellen:

Datentyp: string

Variablen: „My SQL“, „Microsoft SQL Server“

Datenvisualisierung

Datentyp: boolean

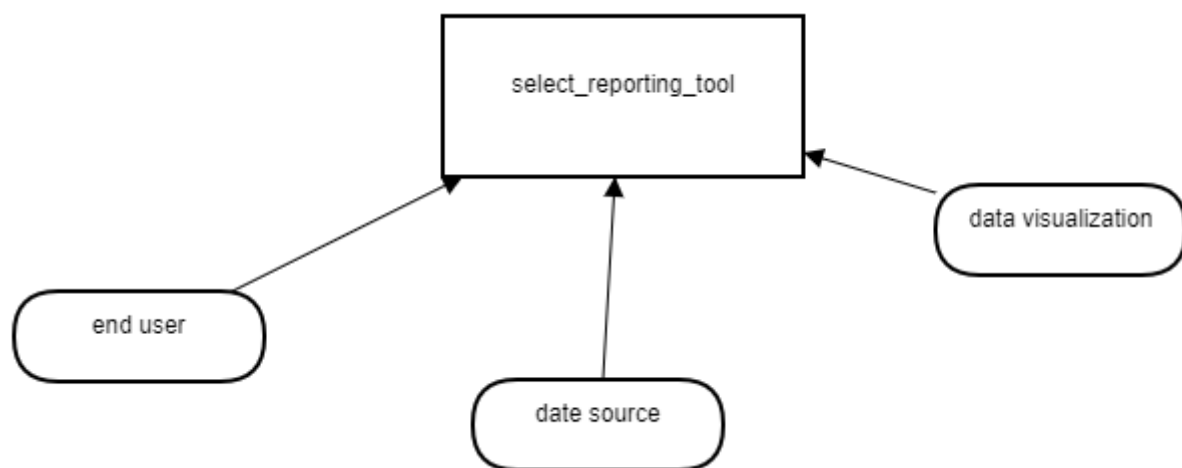
Variablen: true → hohe Anforderung an Datenvisualisierung, z.B. Dashboard, Landeskarte, dynamische Diagramme
false → niedrige oder keine Anforderung an Datenvisualisierung, z.B. einfache Tabelle, einfache Summe

Die Reports für Interne mit hoher Anforderung an Datenvisualisierung sollen in der Regel mittels Sisense fertiggestellt werden. Hingegen sollen die einfachen Reports mit dem Microsoft SQL Server Reporting Tool erstellt werden.

Bei den Reports für die Abteilung Business Intelligence spielt die Anforderung der Datenvisualisierung eine wichtige Rolle. Wenn die Anforderungen an die Datenvisualisierung hoch sind, soll Sisense ausgewählt werden. Ansonsten sollen die Reports mithilfe des

Microsoft SQL Server Reporting Tool erstellt werden. Die Datenquellen sind hier in diesem Fall nicht zu beachten.

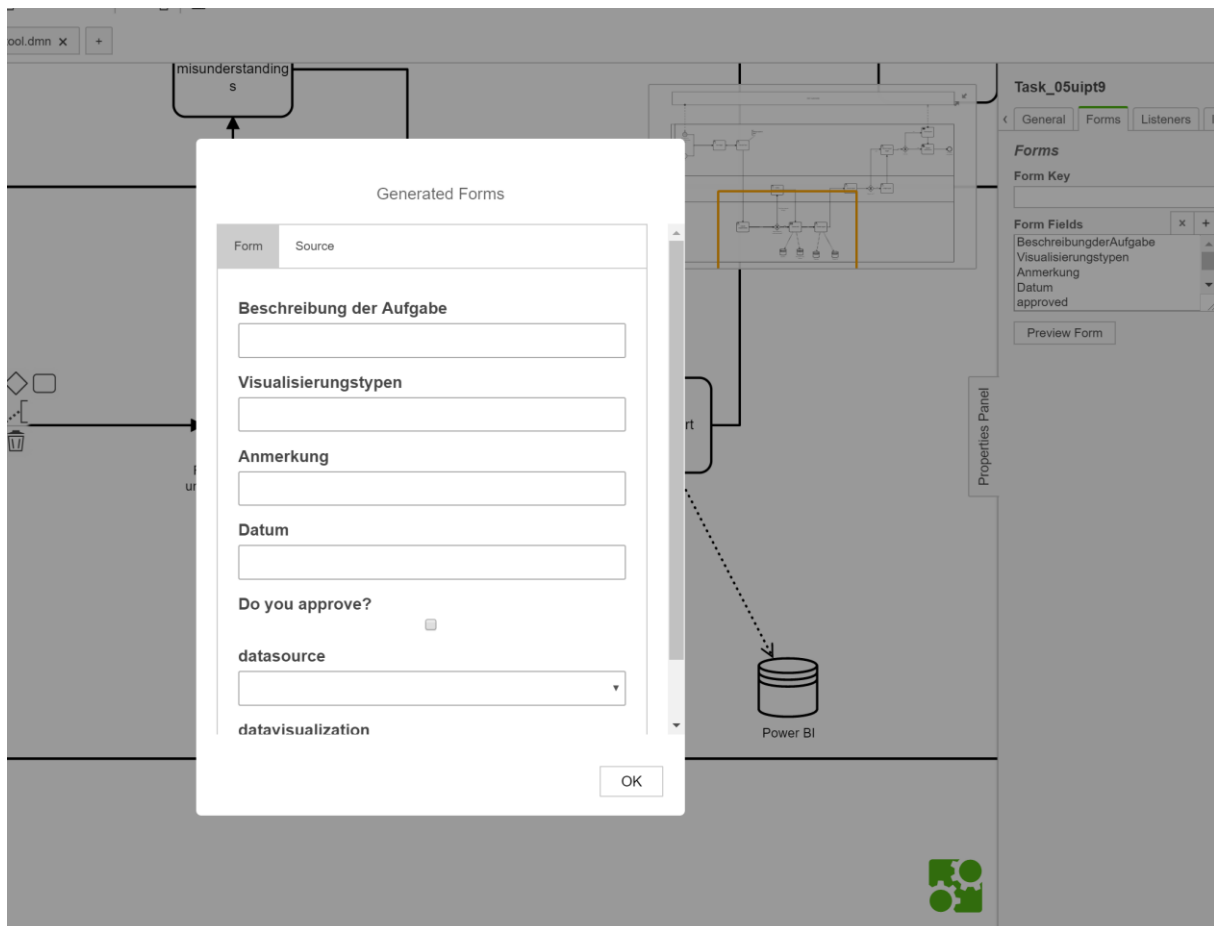
Wenn es die B2B-Kunden betrifft, müssen die Datenquellen und Anforderung der Datenvisualisierung gleichzeitig beachtet werden. Wenn die Datenquellen die Microsoft SQL Datenbank sind, soll Sisense bei einer hohen Anforderung an Grad der Datenvisualisierung eingesetzt werden, während das Microsoft SQL Server Reporting Tool für die einfachen Reports geeignet ist. Wenn die Daten aus der MySQL Datenbanken herkommen, soll Sisense ausgewählt werden.



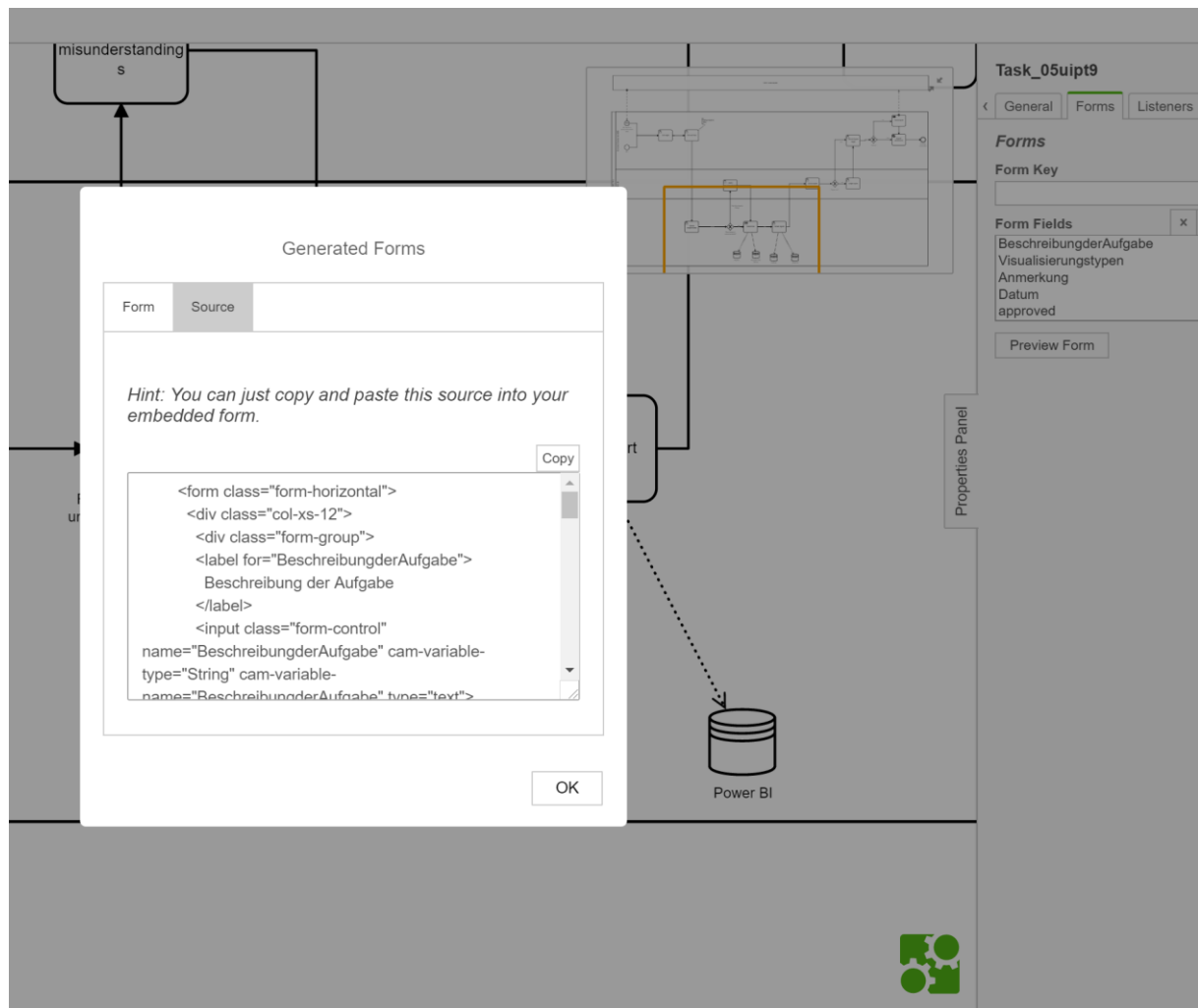
3. Automatisierung des Prozesses in der Process Engine

Nun gehen wir die technische Implementierung des Prozesses in der Camunda Process Engine durch. Für die Implementierung haben wir mehrere Tools verwendet und mit Hilfe der Camunda Dokumentation konnten wir die Automatisierung realisieren. Unter den verwendeten Tools gehören: Camunda, Postmann API und Eclipse (hauptsächlich Maven).

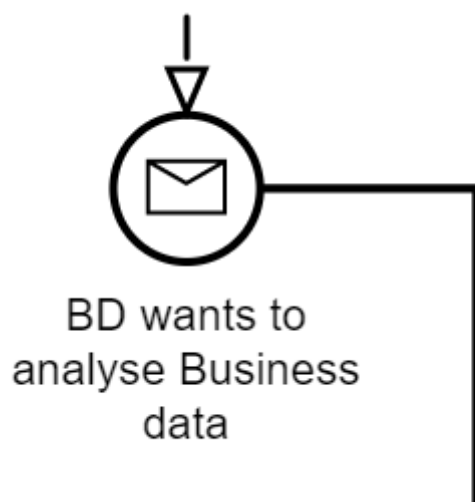
Camunda haben wir verwendet um unser BPMN und DMN -Modell zu erstellen. Durch seine zahlreichen Funktionen konnten wir die Modelle erstellen. Dazu haben wir verschiedene Plugins verwendet, z.B. das Generated Forms Preview Plugin, damit wir unser Formular lokal einsehen konnten.



Der Vorteil dieses Plugin ist, dass bei klicken auf den rechten Reiter „source“, der source(html) code angezeigt wird, der automatisch generiert wurde. Dieser kann später in Eclipse verwendet werden, um Zeit zu sparen. Wie genau es in Eclipse verwendet werden kann, wird später ausführlicher beschrieben.



Als nächstes haben wir die Postman API genutzt, um die Rest API von Camunda zu nutzen. Die Postman API sollte im Nachrichten Start-Event verwendet werden. Dafür müssen zuerst die Eigenschaften für dieses Event im Properties Panel festgelegt werden.



StartEvent_1hukpue

<

General

Forms

Listeners

E >

General

Id

StartEvent_1hukpue x

Name

BD wants to analyse Business data

Details

Message

+

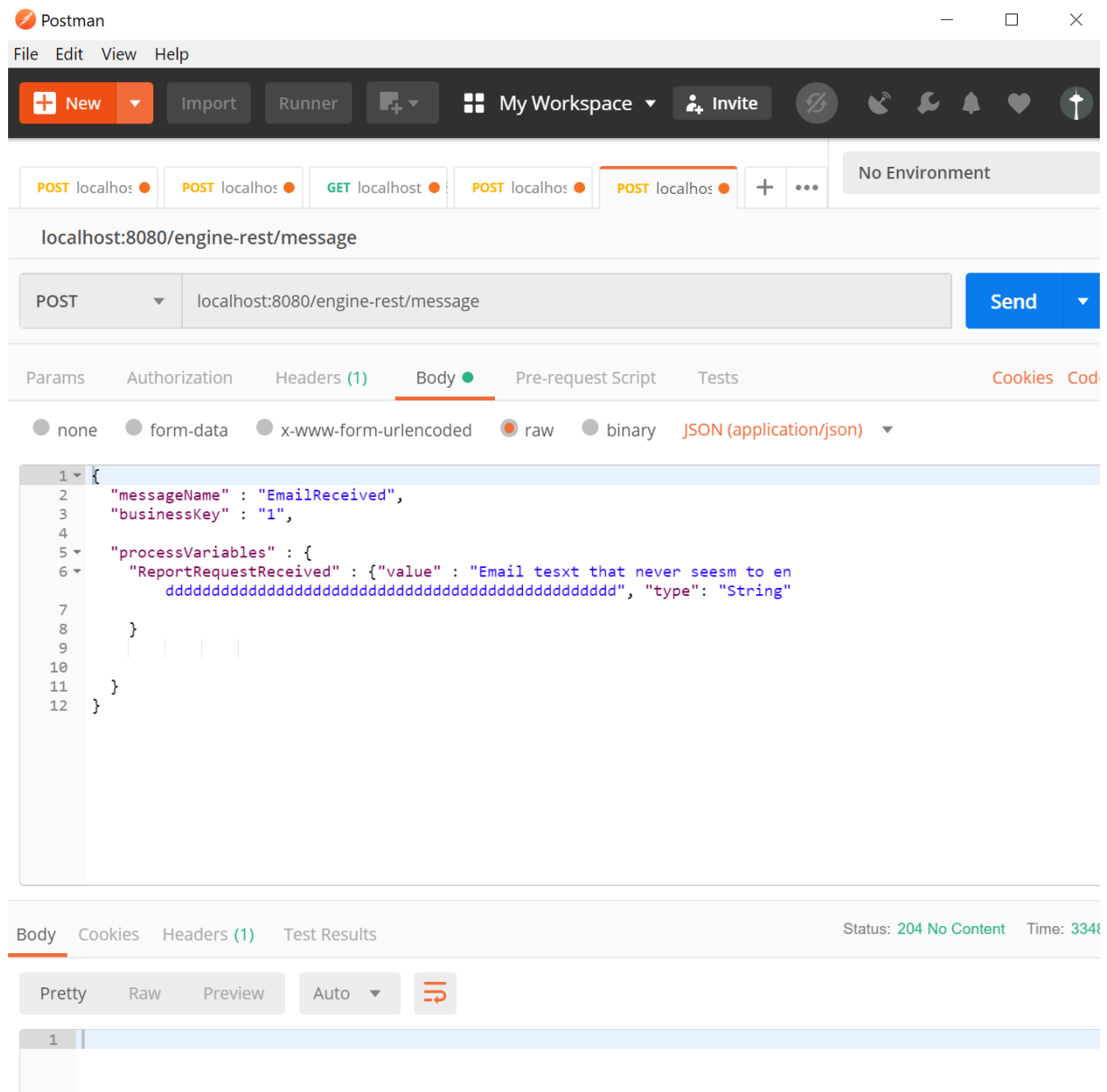
EmailReceived (id=Message_1pu ▼

Message Name

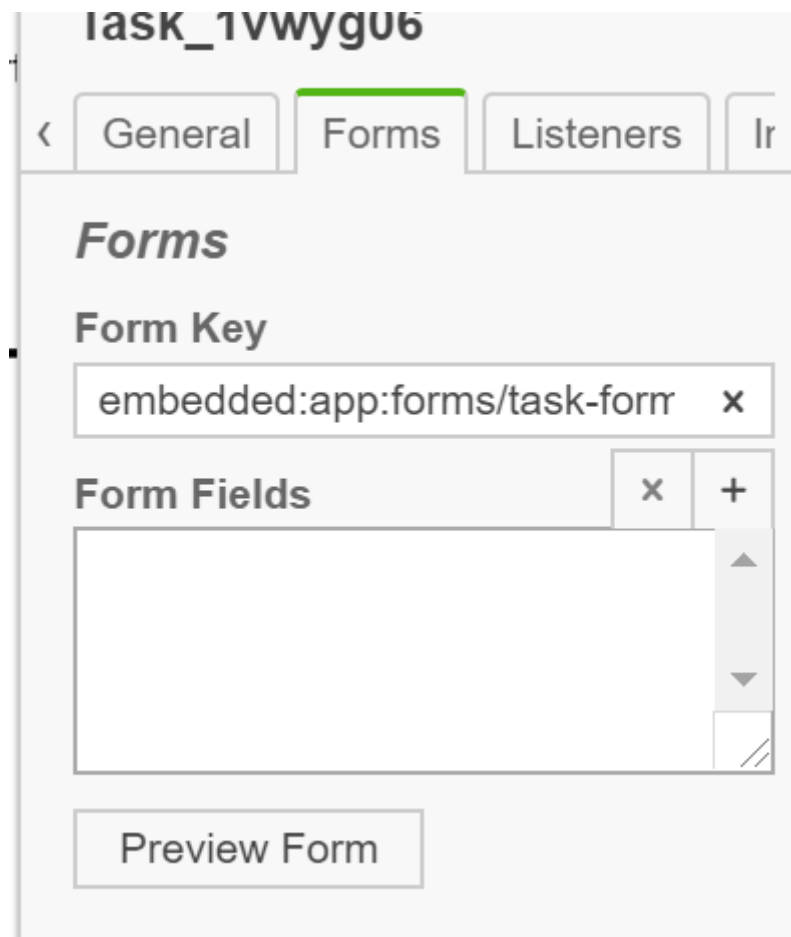
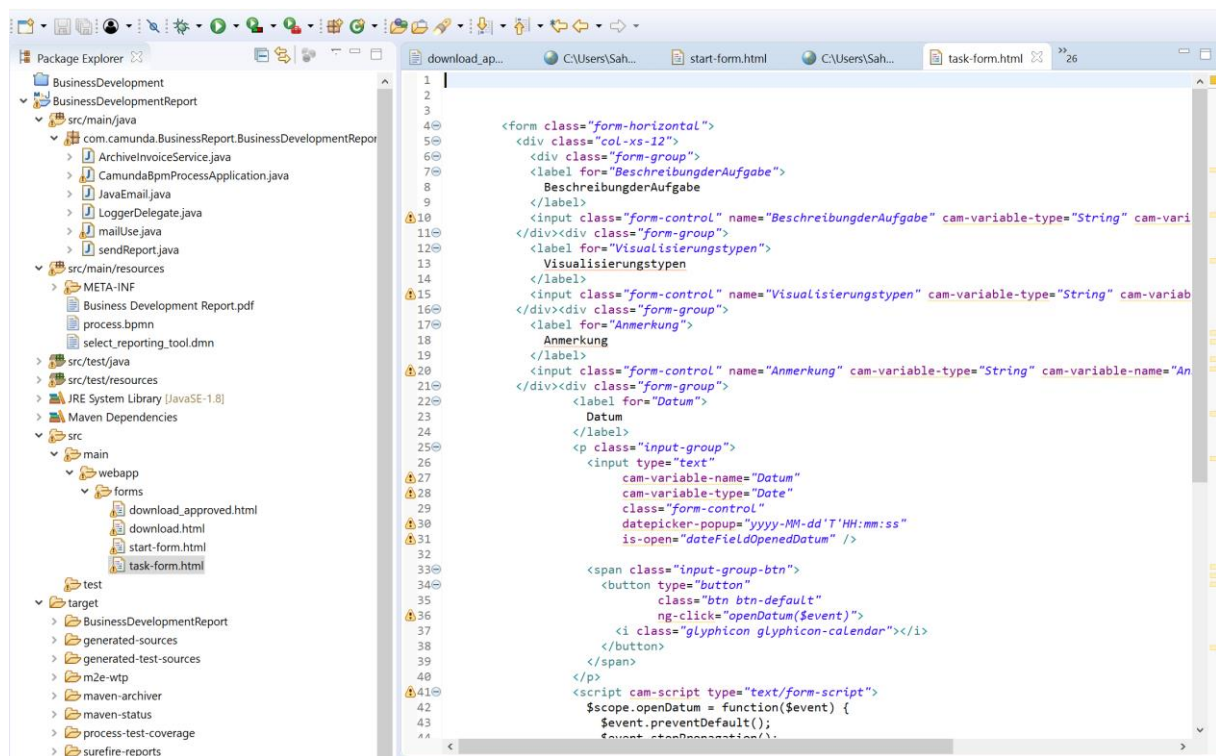
EmailReceived x

Initiator

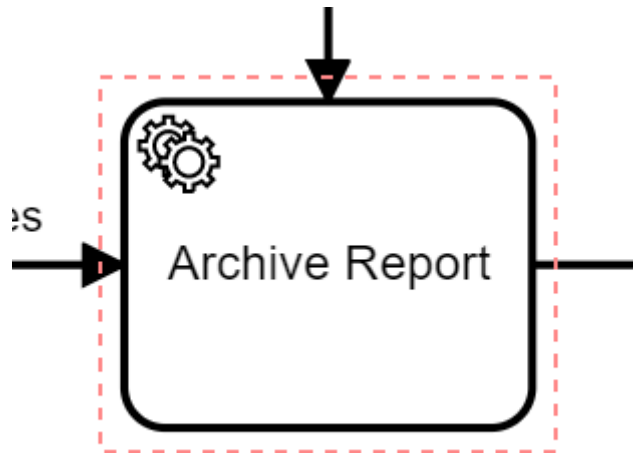
Anschließend legen wir einen java script code in der Postman API fest und senden diesen per Post-Message Befehl an die Process Engine, damit der Prozess startet.



Die restlichen Aspekte des Prozesses wurden mit der Eclipse IDE implementiert. Zuerst haben wir die User Task implementiert. Auch für diesen Schritt war das Plugin sehr hilfreich. Dabei konnten wir den generierten Source-Code in Eclipse kopieren und diese Datei dann durch den Formkey und in der properties-Leiste mit unserem Prozess verbinden.



Anschließend kam die Service Task. Dafür haben wir eine Java-Klasse erstellt, die unseren Report für uns archiviert. Wichtig bei den Java-Klassen ist, dass sie als java delegate Klasse erstellt werden.



Source folder:

Package:


☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

 org.camunda.bpm.engine.delegate.JavaDelegate

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

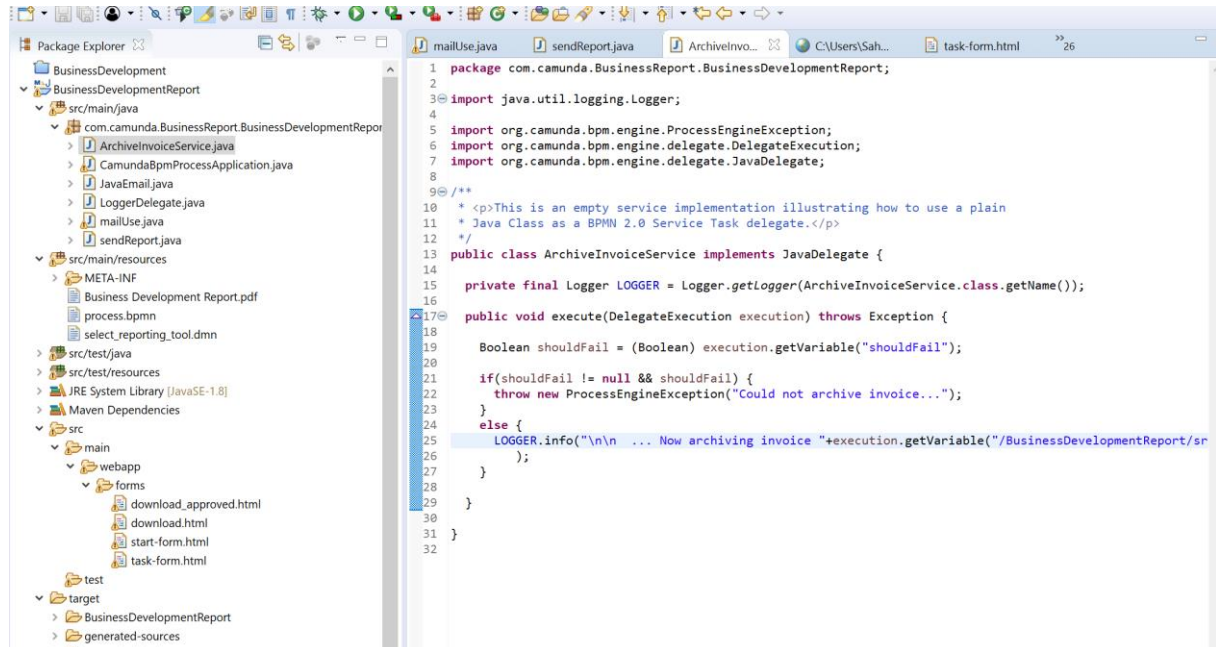
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

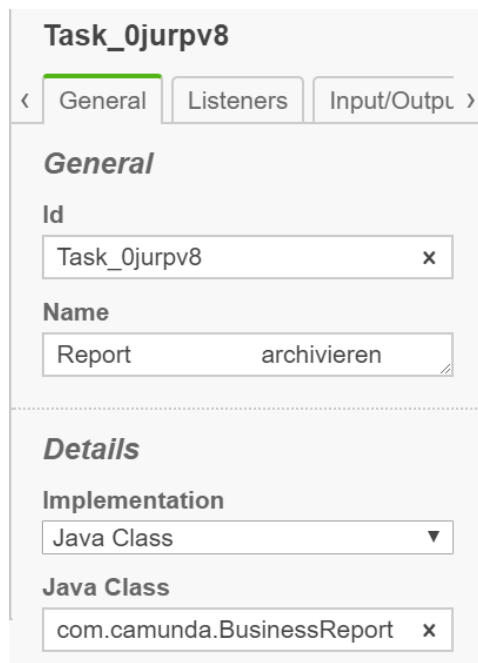
Die Alternative ist in der normalen Java-Klasse der Standard `java.delegate` Klasse zu implementieren.

```
public class mailUse implements JavaDelegate {
```

Dies haben wir getan und den Code für die Service Task festgelegt.



Und durch die Properties Panel wurde die Service Task mit der Java Klasse verbunden.



Als letztes kam die Implementierung der Send Task. Dafür haben wir eine Java-Klasse programmiert und den code mit der Send Task durch die Properties-Leiste verbunden. Ähnlich wie mit der Service Task.

```
package com.camunda.BusinessReport.BusinessDevelopmentReport;

import java.util.Properties;

public class JavaEmail {

    Properties emailProperties;
    Session mailSession;
    MimeMessage emailMessage;

    String emailHost = "smtp.gmail.com";
    String emailPort = "587"; // gmail's smtp port
    String fromUser = "sahversheri@gmail.com"; // your gmail id
    String fromUserEmailPassword = "Password";
    String[] toEmails = { "sahversheri@gmail.com" };

    public void setMailServerProperties() {
        emailProperties = System.getProperties();
        emailProperties.put("mail.smtp.port", emailPort);
        emailProperties.put("mail.smtp.auth", "true");
        emailProperties.put("mail.smtp.starttls.enable", "true");
    }

    public void createEmailMessage(String emailSubject, String emailBody)
        throws AddressException, MessagingException {
        mailSession = Session.getDefaultInstance(emailProperties, null);
        emailMessage = new MimeMessage(mailSession);
        for (int i = 0; i < toEmails.length; i++) {
            emailMessage.addRecipient(Message.RecipientType.TO,
                new InternetAddress(toEmails[i]));
        }
        emailMessage.setSubject(emailSubject);
        // emailMessage.setText(emailBody, "text"); // for a text email
    }

    public void sendEmail() throws AddressException, MessagingException {
        Transport transport = mailSession.getTransport("smtp");
        transport.connect(emailHost, fromUser, fromUserEmailPassword);
        transport.sendMessage(emailMessage, emailMessage.getAllRecipients());
        transport.close();
    }
}
```

Daraufhin erstellten wir den Prozess in Eclipse und laden es auf dem Process Engine Server hoch.

« camunda-bpm-tomcat-7.10.0 » server » apache-tomcat-9.0.12 » webapps					"webapps" ...	
	Name	Änderungsdatum	Typ	Größe		
	BusinessDevelopmentReport	22.01.2019 00:14	Dateiordner			
	camunda	21.01.2019 22:33	Dateiordner			
	camunda-invoice	21.01.2019 22:34	Dateiordner			
	camunda-welcome	21.01.2019 22:34	Dateiordner			
	docs	21.01.2019 22:33	Dateiordner			
	engine-rest	21.01.2019 22:33	Dateiordner			
	examples	21.01.2019 22:33	Dateiordner			
	h2	21.01.2019 22:33	Dateiordner			
	host-manager	21.01.2019 22:33	Dateiordner			
	manager	21.01.2019 22:33	Dateiordner			
	ROOT	21.01.2019 22:32	Dateiordner			
	BusinessDevelopmentReport.war	22.01.2019 00:11	WAR-Datei	20 KB		

Damit war unsere Implementierung beendet.