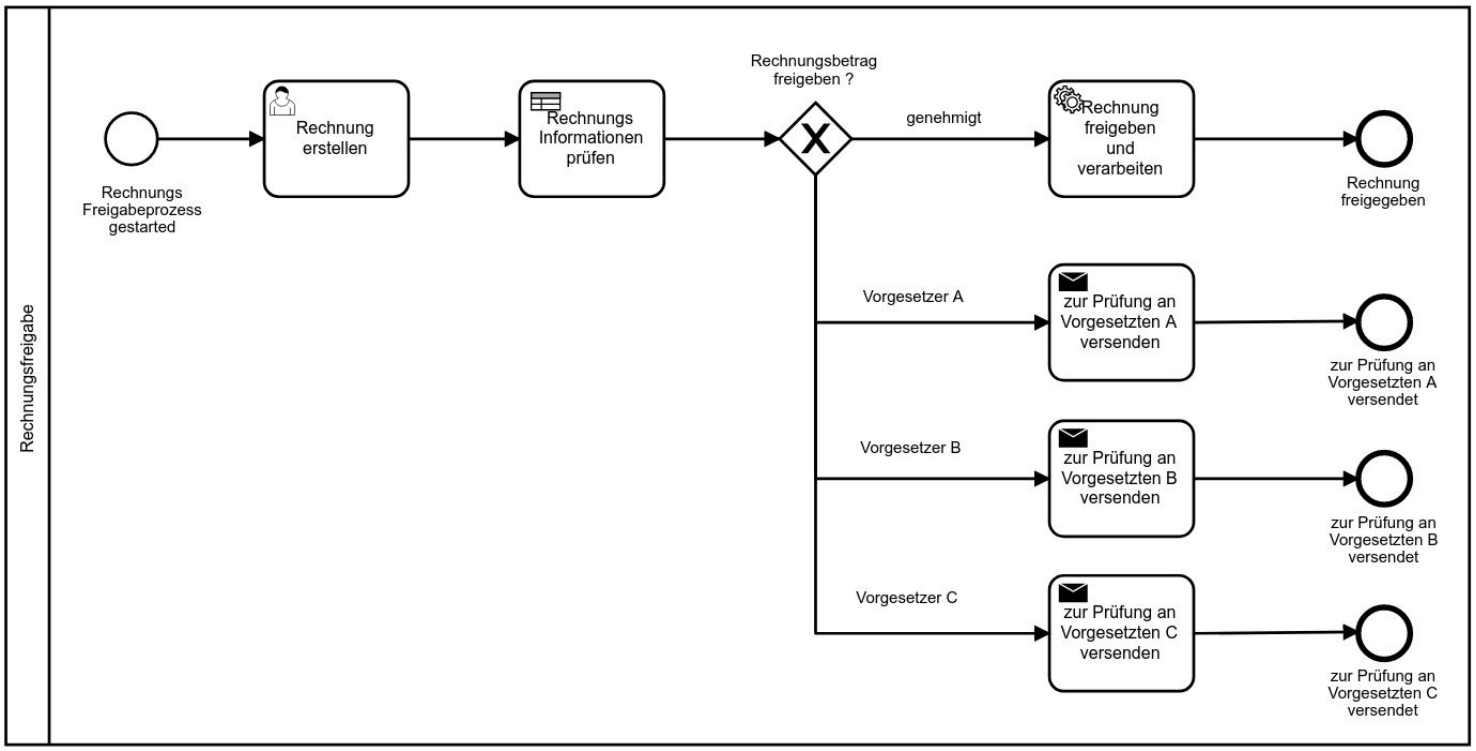
Der BPM Prozess im Dokumentenmanagement System zur Rechnungsfreigabe

Der Prozess im Camunda Modeller

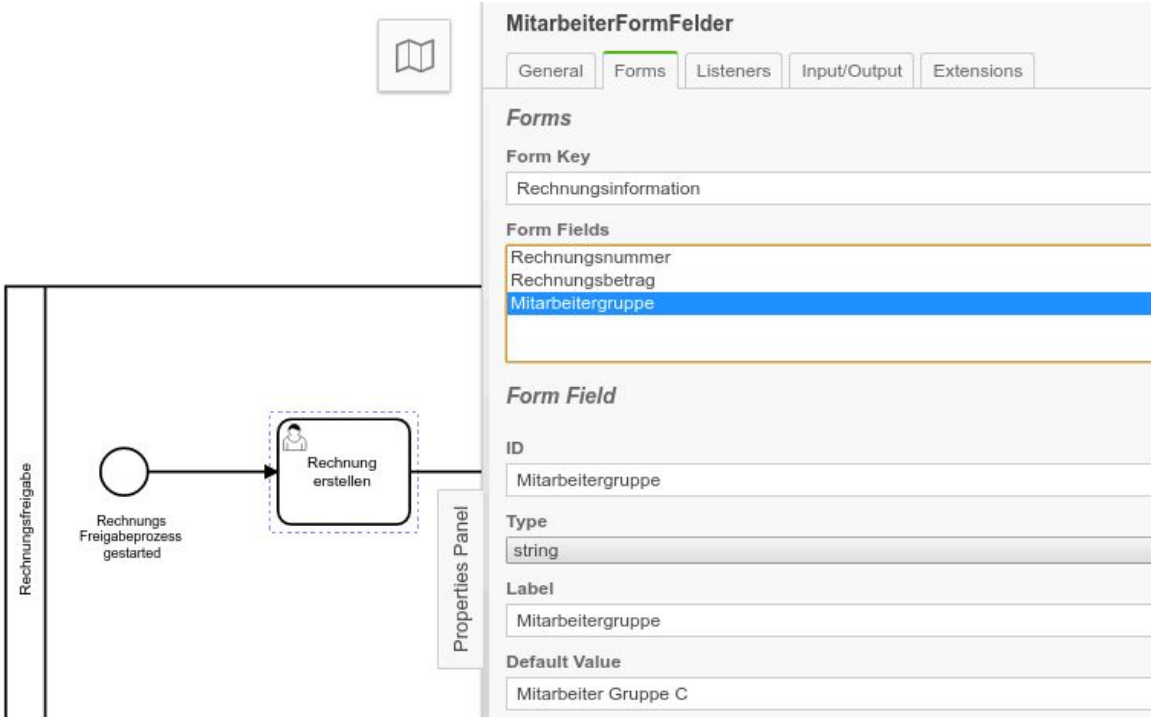
Der Prozess wurde in Camunda modelliert.
In unserem Dokumenten Management System gibt es den Prozess für die Rechnungsfreigabe.
Rechnungen sind Mitarbeitergruppen zugeordnet und haben einen Rechnungsbetrag.
Der Prozess im Dokumentenmanagement System sieht für diesen Prozess wie folgt aus.



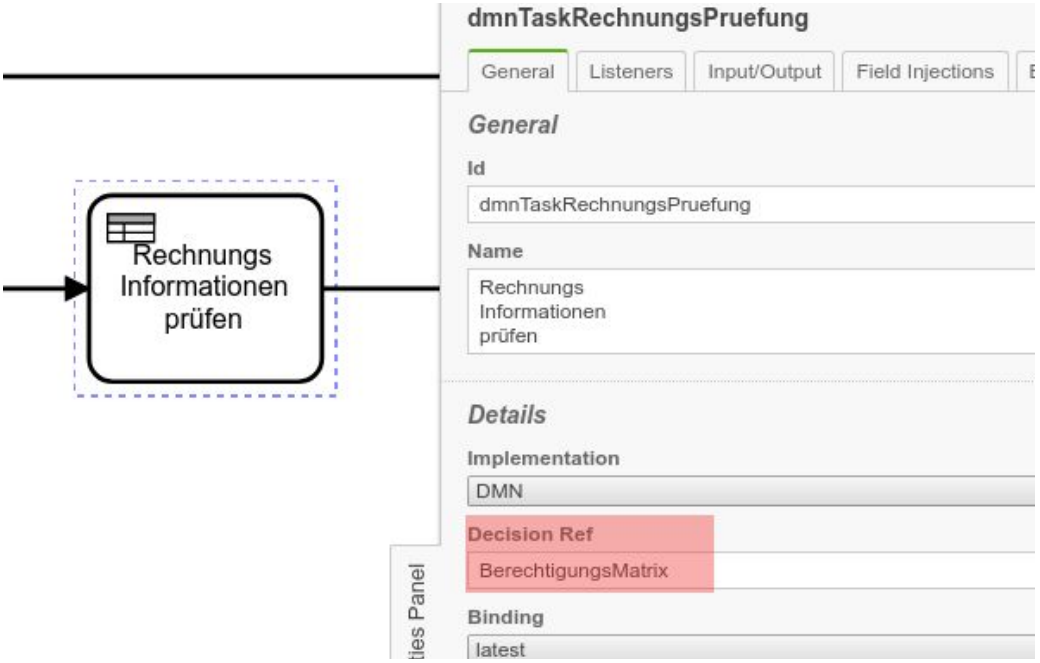
Die Rechnungen werden in der Fachabteilung von Mitarbeitern erstellt, die die Verträge aushandeln.
Die Rechnungen werden im Dokumentenmanagement System dann in den Rechnungsfreigabeprozess aufgenommen und die Parameter eingegeben.
Dieser Service prüft mit der DMN Tabelle die Eingabewerte und genehmigt die Rechnung entweder automatisch und leitet die Informationen über eine nexternen Service an das Rechnungswesen weiter, oder leitet diese an den entsprechenden Vorgesetzten weiter.

In der DMN Entscheidungstabelle wird geprüft in welcher Mitarbeitergruppe der Mitarbeiter eingeteilt ist und welchen Betrag die Rechnung hat.
Durch diese Werte wird die Rechnung entweder automatisch genehmigt oder an den entsprechenden Vorgesetzten weitergeleitet, der die Rechnung im Prozess freigeben muss.

In der ersten Aktivität kann der Mitarbeiter in einer Eingabemaske die Parameter für die **Mitarbeitergruppe** und den **Rechnungsbetrag** eingegeben.
Es werden über eine Form der Rechnungsbetrag und die Mitarbeitergruppe erfasst:



Die Daten werden diese durch eine Entscheidungstabelle verarbeitet:



Die BPM Aktivität für die Entscheidungstabelle sieht im XML wie folgt aus.
hier ist auch die **result** variable definiert:

```
<bpmn:businessRuleTask id="dmnTaskRechnungsPruefung" name="Rechnungs Informationen prüfen"
    camunda:decisionRef="BerechtigungsMatrix"
    camunda:mapDecisionResult="singleEntry"
    camunda:resultVariable="result" >
  <bpmn:incoming>SequenceFlow_0r08tj1</bpmn:incoming>
  <bpmn:outgoing>SequenceFlow_097nb1v</bpmn:outgoing>
</bpmn:businessRuleTask>
```

Hier gibt es den Parameter `decisionRef` mit dem Verweis auf unsere Entscheidungstabelle und den Parameter `resultVariable` der das Ergebnis aus der Entscheidungstabelle bekommt.

Die Entscheidungstabelle hat als Ausgang ein **result**, das sich durch die Parameter aus der Mitarbeiter Form der **Mitarbeitergruppe** und dem **Rechnungsbetrag** entscheidet.
In der BPM Regel Aktivität haben wir das DMN mit der ID “BerechtigungsMatrix” verlinkt, diese ID geben wir nun unserer DMN Entscheidungstabelle:

BerechtigungsMatrixDocManagement

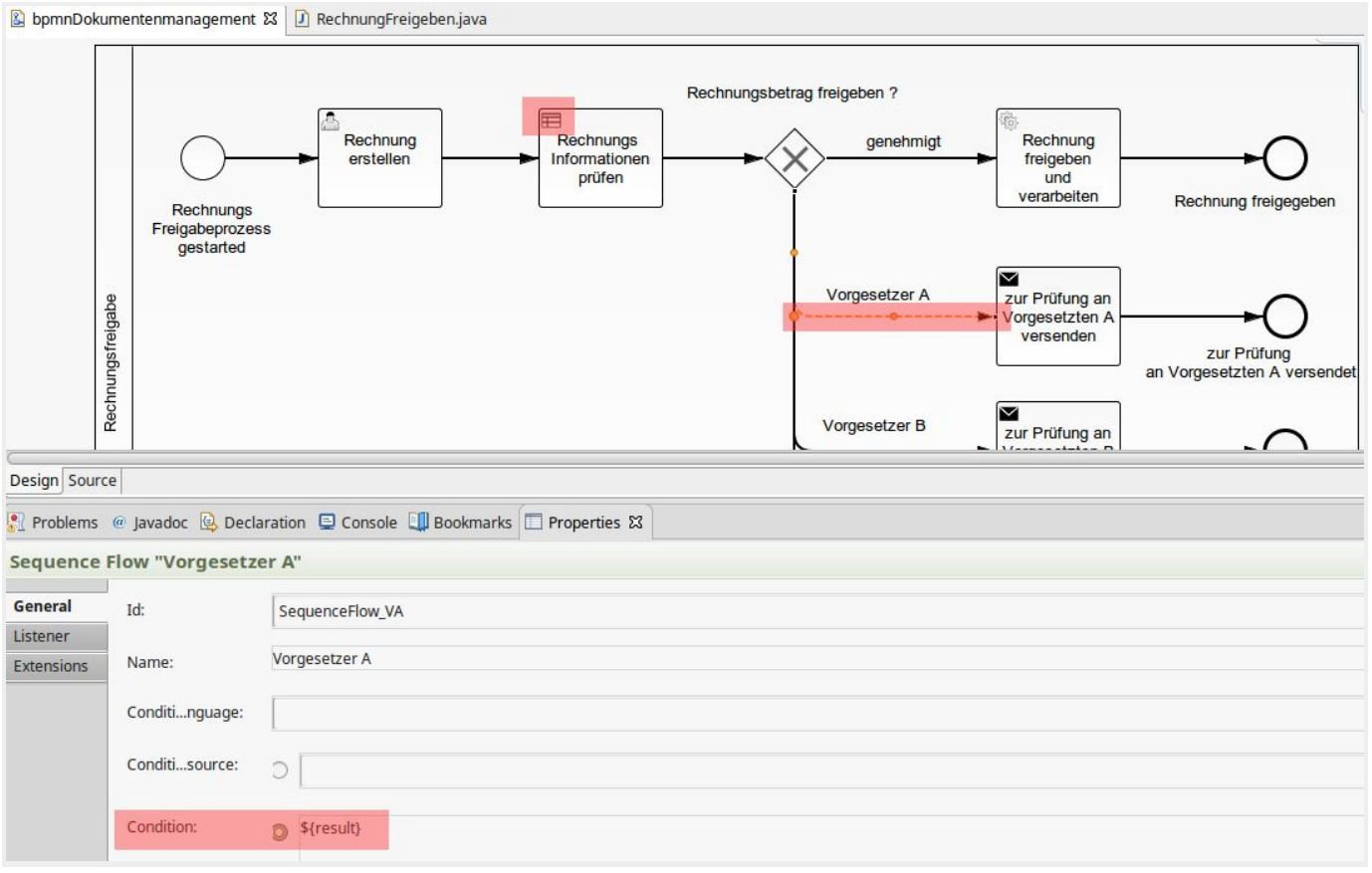
BerechtigungsMatrix

F	Input		Output	Annotation
	Mitarbeitergruppe	Rechnungsbetrag	result	
	string	integer	string	
1	"Mitarbeiter Gruppe A"	<= 1000	"genehmigt"	wird automatisch genehmigt
2	"Mitarbeiter Gruppe A"	> 1000	"Vorgesetzer A"	-
3	"Mitarbeiter Gruppe B"	<= 500	"genehmigt"	wird automatisch genehmigt
4	"Mitarbeiter Gruppe B"	> 500	"Vorgesetzter B"	-
5	"Mitarbeiter Gruppe C", "Mitarbeiter Gruppe D"	> 0	"Vorgesetzter C"	-

In der Camunda Dokumentation zur Integration von DMN Entscheidungstabellen in den BPM Prozess
https://docs.camunda.org/manual/develop/user-guide/process-engine/decisions/bpmn-cmmn
steht dann, dass die **result** variable so benutzt werden kann:

```
Object value = decisionResult
        .getSingleResult()
        .getEntry("result");
```

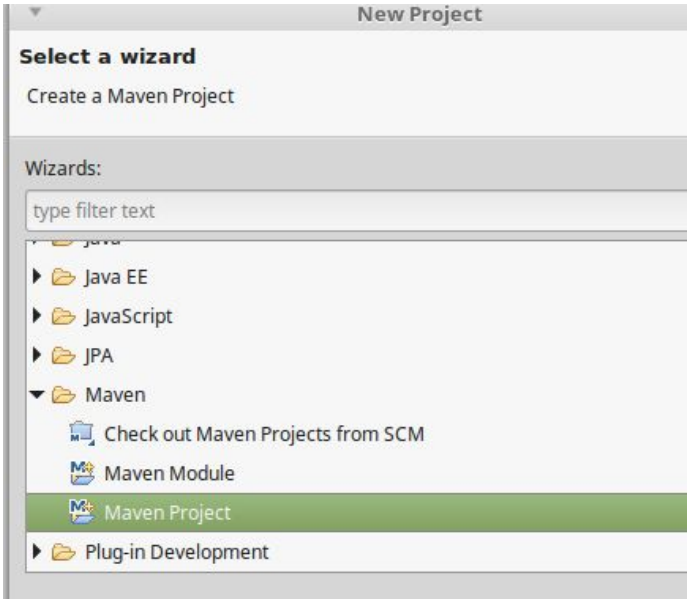
Das Result wird dann im Gateway ausgewertet:



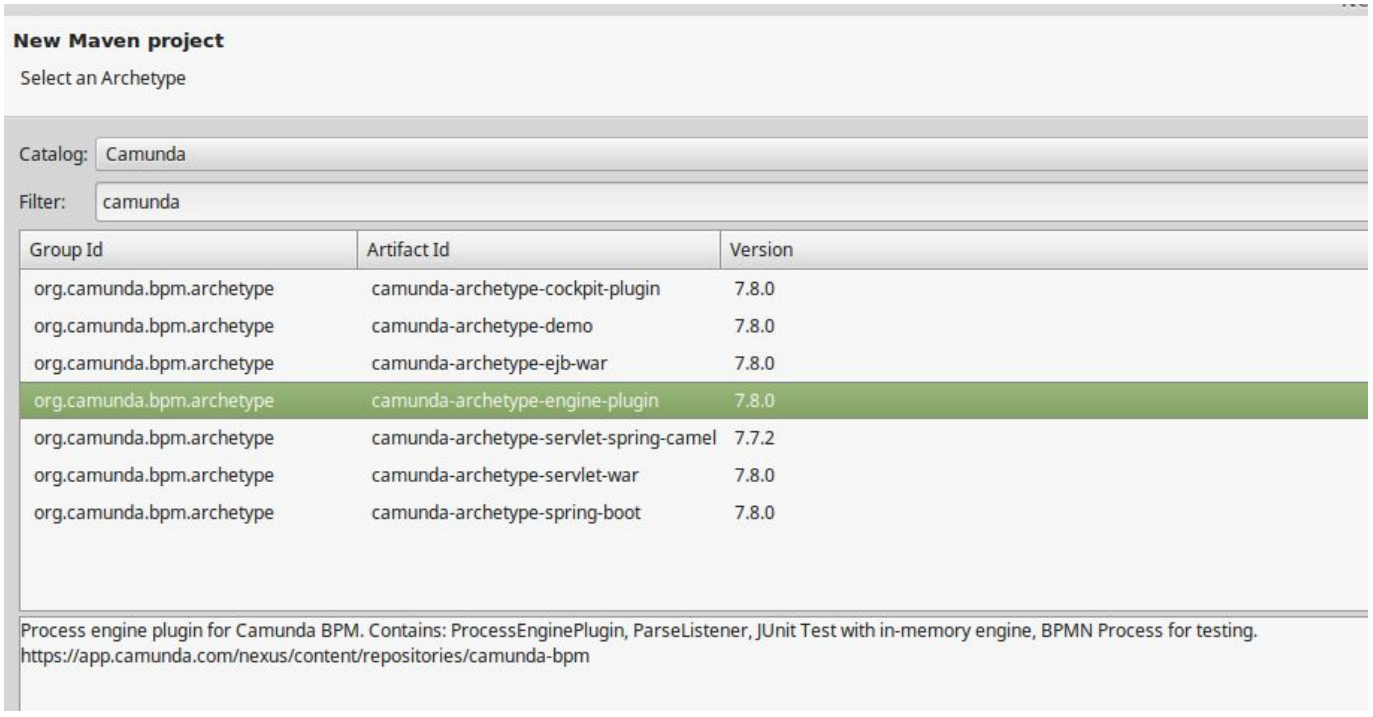
Mit Eclipse und Maven zum ausführbaren Prozess

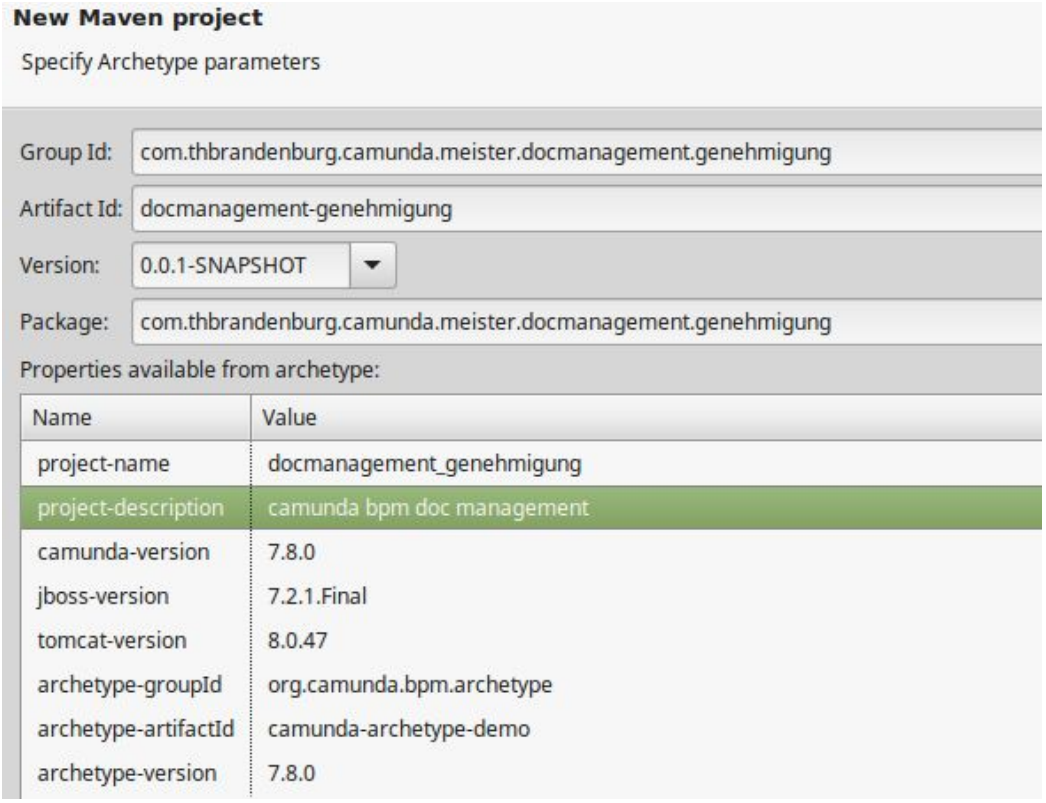
Zur Realisierung wird die BPM Engine von Camunda verwendet, welche Möglichkeiten der BPM 2.0 Spezifikation benutzt. Das erstellte BPM wird in Eclipse importiert und es werden die Camunda Bibliotheken als Plugin installiert und dann können wir die Lane aus dem Prozessmodell ausführbar einstellen und über die Programmiersprache Java Services aufrufen und automatische Logik in den Prozess integrieren.

In Eclipse erstellen wird dazu ein Maven Projekt:



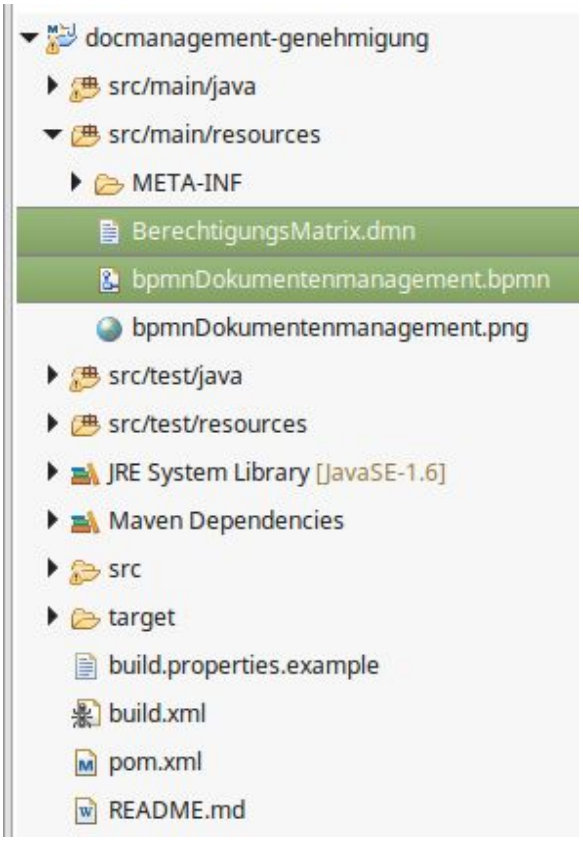
und nutzen die Camunda Plugins um die Entsprechende Projektstruktur automatisch zu erstellen:



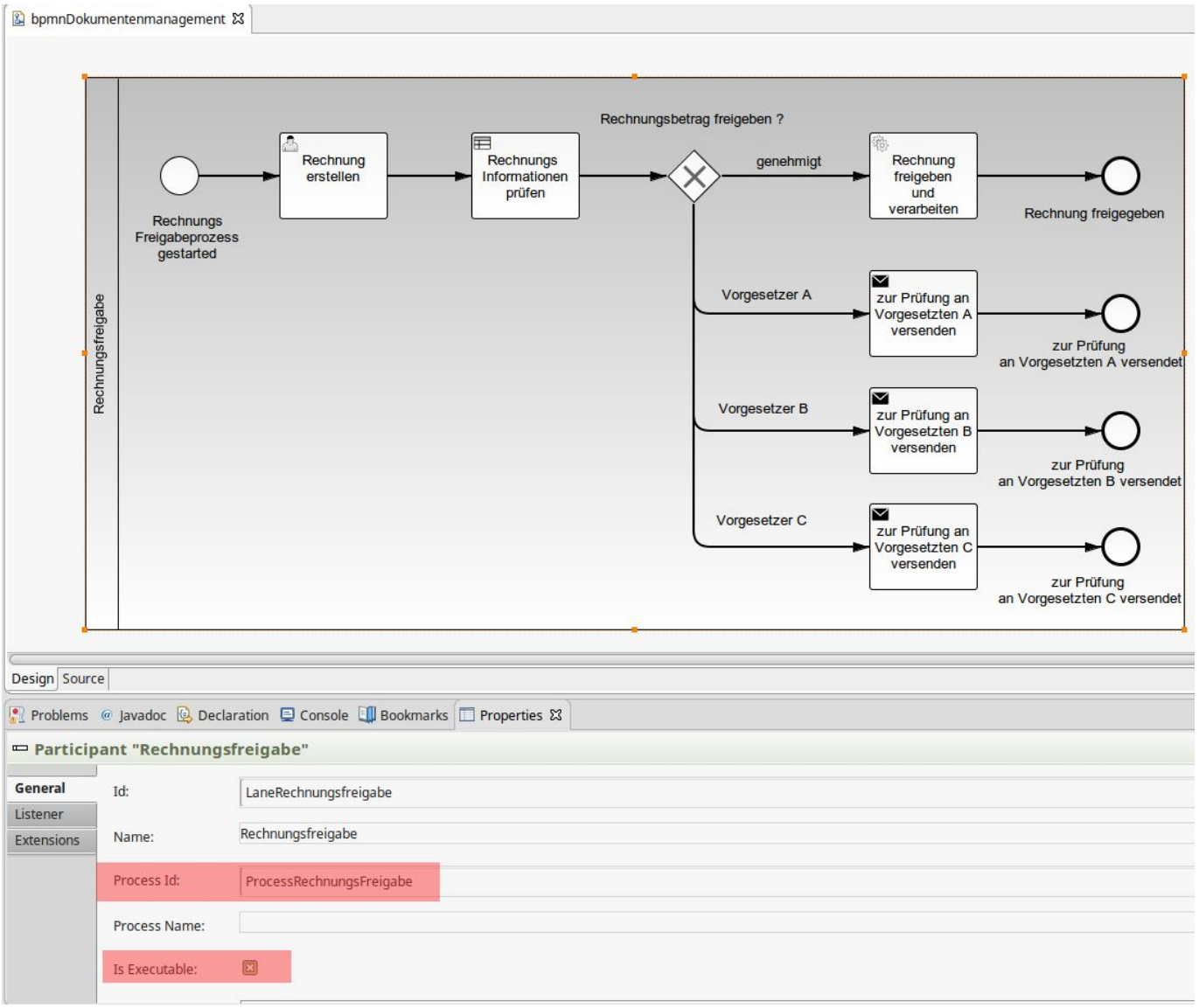


Die Camunda Maven Archetypes können hier bezogen werden, diese können dann in Eclipse als Software installiert werden:
https://docs.camunda.org/manual/7.4/user-guide/process-applications/maven-archetypes/

Dadurch wird die Projektstruktur automatisch erstellt und wir können im nächsten Schritt unser BPM aus Camunda importieren:



Nun kommen die Eclipse spezifischen Einstellungen zum Tragen, durch die wir die Lane automatisch ausführbar machen.

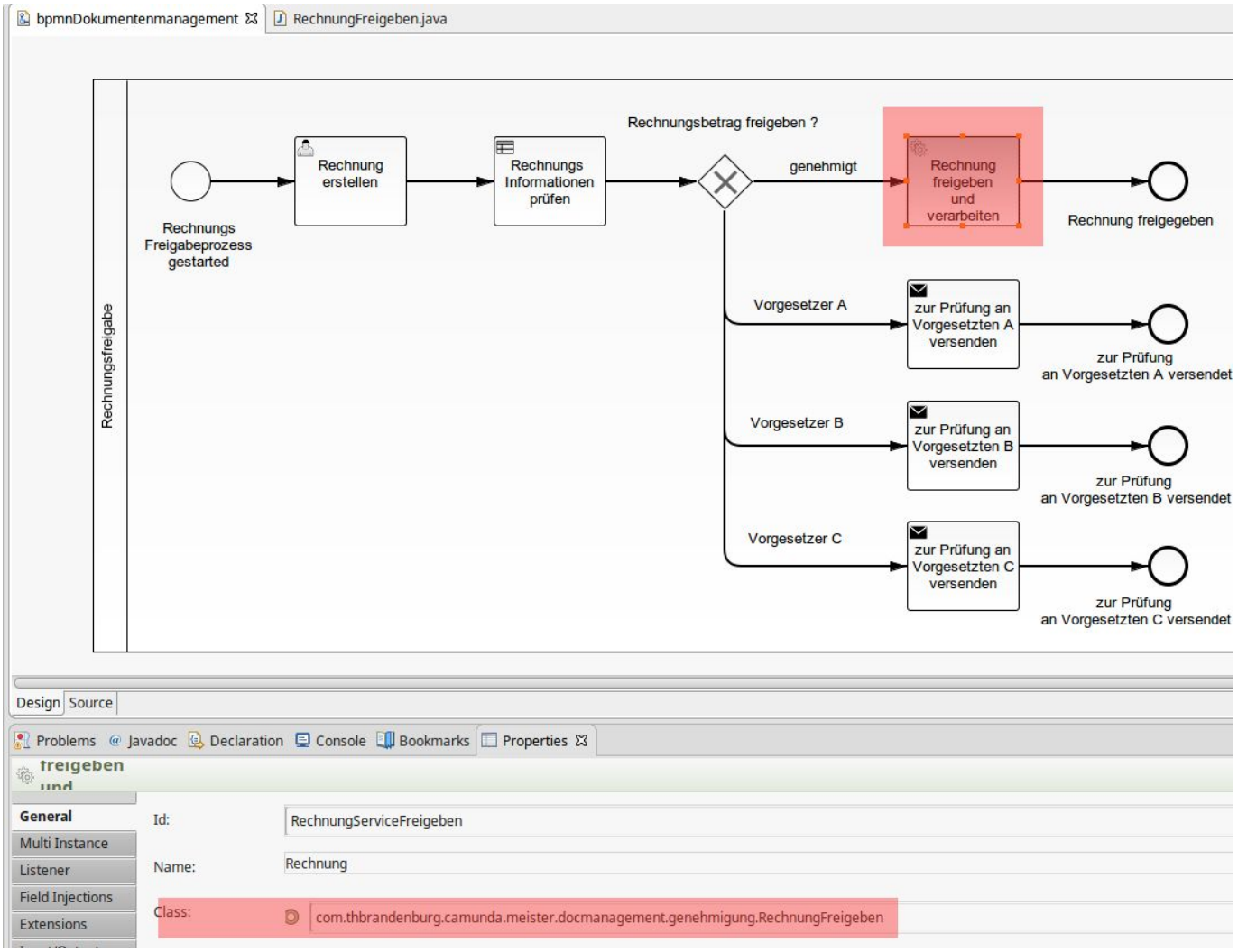


Die Lane wird als “ausführbar” markiert und die Prozess ID ist wichtig, da wir diese nun im Java Code benutzen, um eine Instanz der prozess Engine zu erstellen.

Prozess ID: ProcessRechnungsFreigabe
BPM Datei: bpmnDokumentenmanagement.bpmn

Diese Informationen werden benutzt um nun in einem Test zu schauen, ob das BPM okay ist und eine Prozess Engine daraus erstellt werden kann. In Eclipse wird ein sogenannter Unit Test ausgeführt, der auf das BPM verweist und unsere Prozess Id benutzt um die Engine zu erstellen.

Darüber hinaus müssen allen Service Aktivitäten im Prozess direkte Java Klassen zugeordnet werden, die dann bei der Aktivität aufgerufen werden und Java Logik ausführen können.



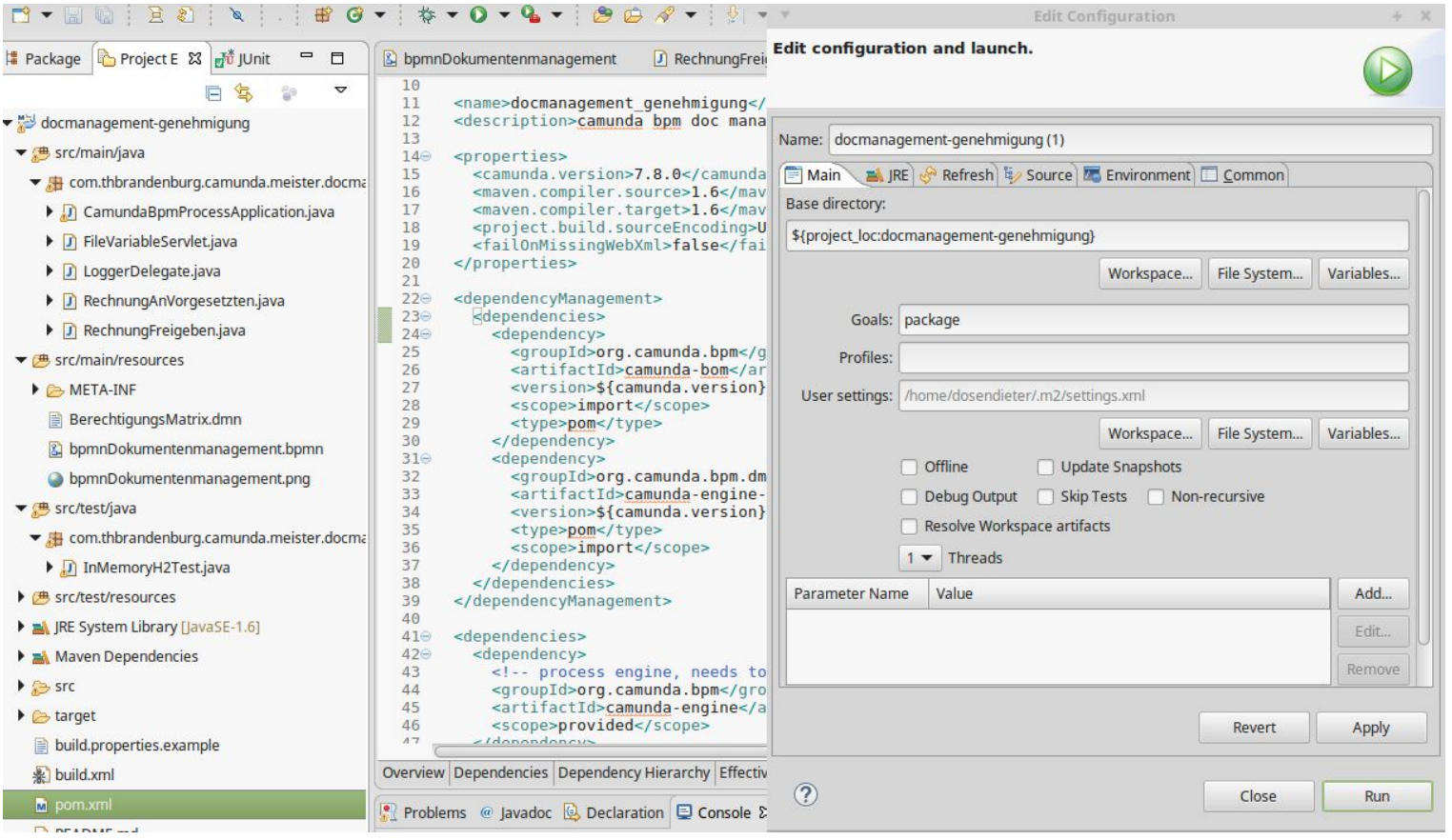
Die verknüpfte Klasse muss die Camunda Bibliotheken importieren, von der Camunda Klasse JavaDelegate erben und die “public void” Methode execute mit dem Parameter für die Ausführung haben:

Die Methode wird dann von der Prozess Engine aufgerufen.

Für unsere Tests benutzen wir die Java Unit Test Methoden, die prüfen, ob alles okay ist.

```java
package com.thbrandenburg.camunda.meister.docmanagement.genehmigung;

import java.sql.SQLException;

/**
 * Test case starting an in-memory database-backed Process Engine.
 */
public class InMemoryH2Test {

  @ClassRule
  @Rule
  public static ProcessEngineRule rule = TestCoverageProcessEngineRuleBuilder.create().build();

  private static final String PROCESS_DEFINITION_KEY = "ProcessRechnungsFreigabe";

  static {
    LogFactory.useSlf4jLogging(); // MyBatis
  }

  @Before
  public void setup() {
    init(rule.getProcessEngine());
  }

  /**
   * Just tests if the process definition is deployable.
   */
  @Test
  @Deployment(resources = "bpmnDokumentenmanagement.bpmn")
  public void testParsingAndDeployment() throws SQLException {
    // nothing is done here, as we just want to check for exceptions during deployment

      ProcessInstance processInstance = processEngine().getRuntimeService().startProcessInstanceByKey(PROCESS_DEFINITION_KEY);

```

Nachdem das BPM entsprechend angepasst wurde und die Java Klassen hinterlegt sind, kann der Unit test gestartet werden, um zu prüfen ob das deployen funktioniert.
Wenn die Tests okay sind, kann das Maven Projekt deployed werden:



Beim Deployed wird der Test durchgeführt:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.thbrandenburg.camunda.meister.docmanagement.genehmigung.InMemoryH2Test
22:29:30.891 [main] DEBUG org.camunda.bpm.engine.test - ==== BUILDING PROCESS ENGINE ========================================================================
22:29:30.948 [main] INFO  o.s.b.f.xml.XmlBeanDefinitionReader - Loading XML bean definitions from class path resource [camunda.cfg.xml]
22:29:31.335 [main] INFO  org.camunda.bpm.engine.cfg - ENGINE-12003 Plugin 'SpinProcessEnginePlugin' activated on process engine 'default'
22:29:31.339 [main] INFO  org.camunda.spin - SPIN-01010 Discovered Spin data format provider: org.camunda.spin.impl.json.jackson.format.JacksonJsonDat
22:29:31.440 [main] INFO  org.camunda.spin - SPIN-01010 Discovered Spin data format provider: org.camunda.spin.impl.xml.dom.format.DomXmlDataFormatPro
22:29:31.446 [main] INFO  org.camunda.spin - SPIN-01009 Discovered Spin data format: org.camunda.spin.impl.xml.dom.format.DomXmlDataFormat[name = appl
22:29:31.446 [main] INFO  org.camunda.spin - SPIN-01009 Discovered Spin data format: org.camunda.spin.impl.json.jackson.format.JacksonJsonDataFormat[n
22:29:31.446 [main] INFO  org.camunda.bpm.engine.cfg - ENGINE-12003 Plugin 'ConnectProcessEnginePlugin' activated on process engine 'default'
22:29:31.667 [main] INFO  org.camunda.bpm.connect - CNCT-01004 Discovered provider for connector id 'http-connector' and class 'org.camunda.connect.ht
22:29:31.669 [main] INFO  org.camunda.bpm.connect - CNCT-01004 Discovered provider for connector id 'soap-http-connector' and class 'org.camunda.conne
22:29:33.727 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03016 Performing database operation 'create' on component 'engine' with resource
22:29:33.734 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03016 Performing database operation 'create' on component 'history' with resourc
22:29:33.738 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03016 Performing database operation 'create' on component 'identity' with resour
22:29:33.745 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03016 Performing database operation 'create' on component 'case.engine' with res
22:29:33.746 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03016 Performing database operation 'create' on component 'case.history' with re
22:29:33.748 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03016 Performing database operation 'create' on component 'decision.engine' with
22:29:33.750 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03016 Performing database operation 'create' on component 'decision.history' wit
22:29:33.777 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03067 No history level property found in database
22:29:33.778 [main] INFO  org.camunda.bpm.engine.persistence - ENGINE-03065 Creating historyLevel property in database for level: HistoryLevelFull(nam
22:29:33.781 [main] INFO  org.camunda.bpm.engine - ENGINE-00001 Process Engine default created.
22:29:33.788 [main] DEBUG org.camunda.bpm.engine.test - ==== PROCESS ENGINE CREATED ========================================================================
22:29:33.791 [main] DEBUG org.camunda.bpm.engine.test - annotation @Deployment creates deployment for InMemoryH2Test.testParsingAndDeployment
Dez 30, 2017 10:29:34 PM org.camunda.bpm.extension.process_test_coverage.junit.rules.TestCoverageProcessEngineRule succeeded
INFORMATION: testParsingAndDeployment(com.thbrandenburg.camunda.meister.docmanagement.genehmigung.InMemoryH2Test) succeeded.
Dez 30, 2017 10:29:34 PM org.camunda.bpm.extension.process_test_coverage.junit.rules.TestCoverageProcessEngineRule handleTestMethodCoverage
INFORMATION: testParsingAndDeployment test method coverage is 0.13043478260869565
22:29:34.314 [main] DEBUG org.camunda.bpm.engine.test - annotation @Deployment deletes deployment for InMemoryH2Test.testParsingAndDeployment
Dez 30, 2017 10:29:34 PM org.camunda.bpm.extension.process_test_coverage.junit.rules.TestCoverageProcessEngineRule succeeded
INFORMATION: com.thbrandenburg.camunda.meister.docmanagement.genehmigung.InMemoryH2Test succeeded.
Dez 30, 2017 10:29:34 PM org.camunda.bpm.extension.process_test_coverage.junit.rules.TestCoverageProcessEngineRule handleClassCoverage
INFORMATION: com.thbrandenburg.camunda.meister.docmanagement.genehmigung.InMemoryH2Test test class coverage is: 0.13043478260869565
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.667 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

Und die WAR Datei für den Server erstellt:

```
[INFO] Copying webapp resources [/home/doren/workspace/docmanagement-genehmigung/src/main/webapp]
[INFO] Webapp assembled in [24 msecs]
[INFO] Building war: /home/doren/workspace/docmanagement-genehmigung/target/docmanagement-genehmigung.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 10.282 s
[INFO] Finished at: 2017-12-30T22:29:36+01:00
[INFO] Final Memory: 23M/248M
[INFO] ------------------------------------------------------------------------
```

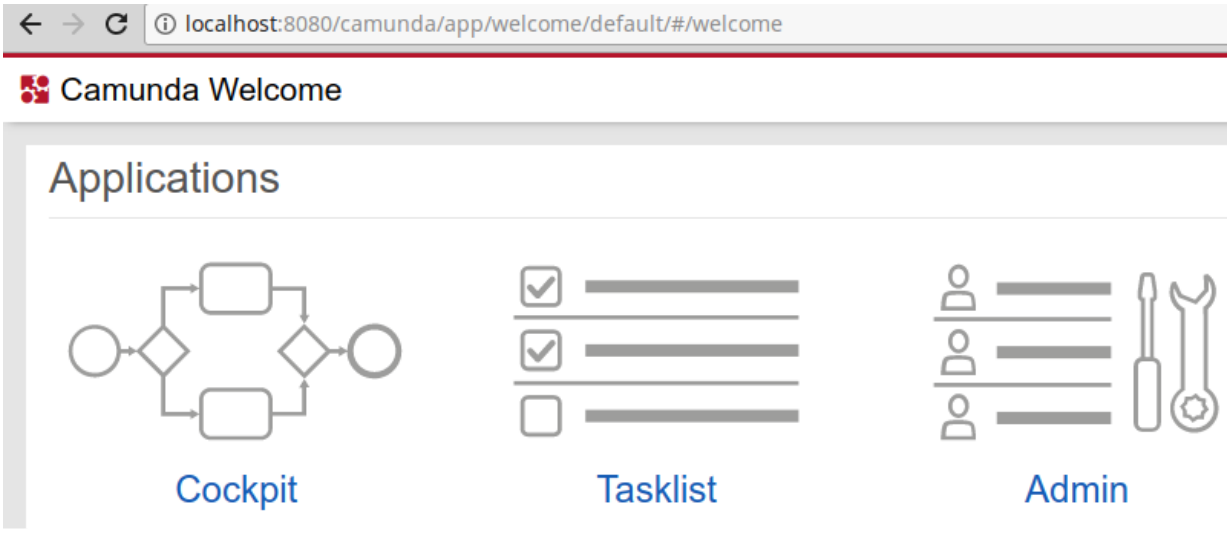
Camunda BPM Plattform

Um lokal unseren BPM Prozess und die Entscheidungstabelle zu testen ,habe wir die Camunda Plattform geladen:
https://camunda.org/download/

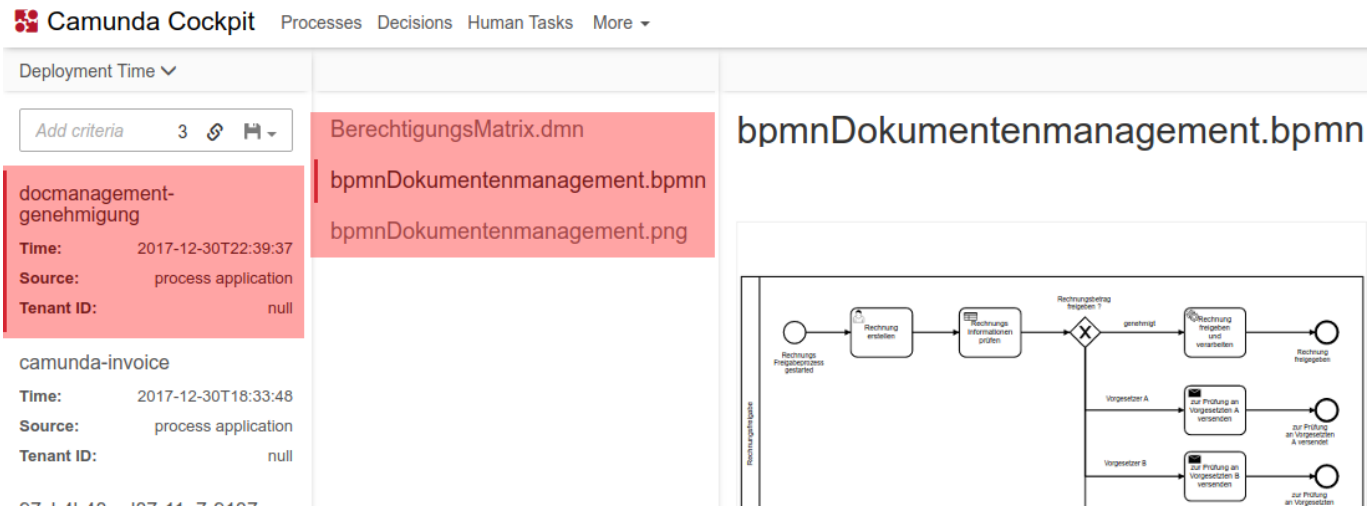
Webserver mit Camunda Plattform starten:

```
~/Camunda/tomCat $ ./start-camunda.sh
starting camunda BPM platform on Tomcat Application Server
Using CATALINA_BASE:   ./server/apache-tomcat-8.0.47
Using CATALINA_HOME:   ./server/apache-tomcat-8.0.47
Using CATALINA_TMPDIR: ./server/apache-tomcat-8.0.47/temp
Using JRE_HOME:        /usr/lib/jvm/java-8-oracle
Using CLASSPATH:
./server/apache-tomcat-8.0.47/bin/bootstrap.jar:./server/apache-tomcat-8.0.47/bin/tomcat-juli.jar
Tomcat started.
```

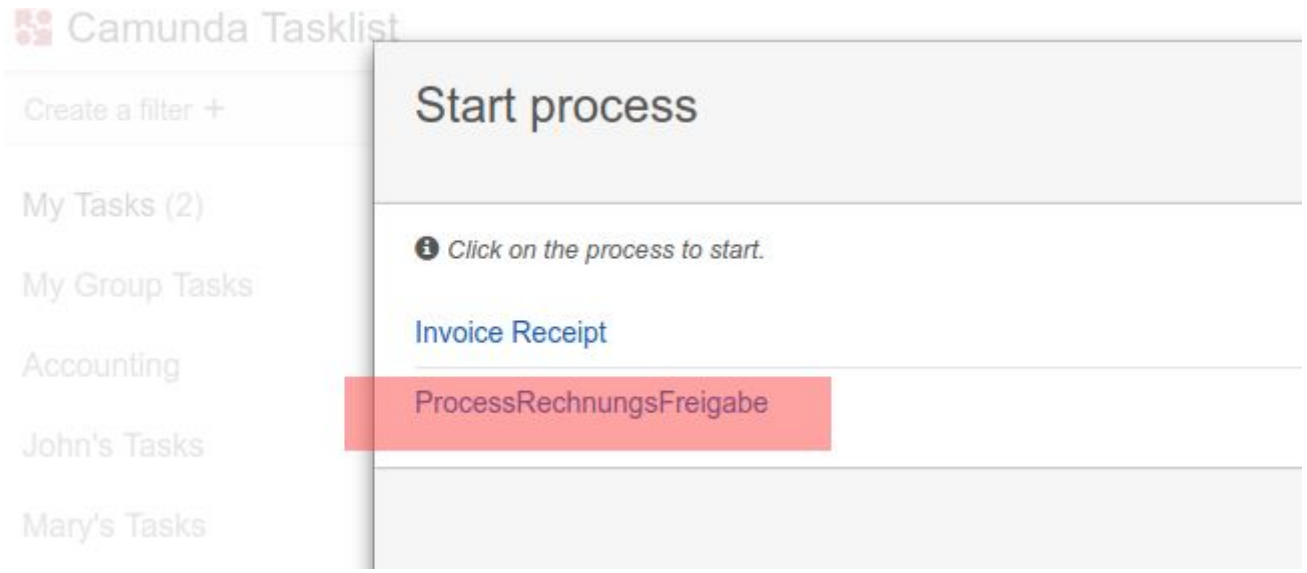
Und lokal läuft nun die Camunda Plattform zum Testen:



Hier können wir das Eclipse Projekt importieren:



In der Taskliste kann nun der Prozess gestartet werden:



Und dadurch werden Instances zu den Prozess aktiv:

2 process definitions deployed

State	Incidents	Running Instances	Name
✓	0	3	Invoice Receipt
✓	0	1	ProcessRechnungsFreigabe