Tallinna Ülikool

Digitehnoloogiate Instituut

Camunda protsessimootori tutvustus

Seminaritöö

Autor: Keio Arula

Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „2015

Juhendaja: ..... „ ..... „2015

Instituudi direktor: ..... „ ..... „2015

Tallinn 2015

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....	.....
(kuupäev)	(autor)

Sisukord

Sissejuhatus

Antud seminaritöö eesmärgiks on tutvustada Camunda protsessimootorit ning selle võimalusi. Ühtlasi viiakse seminaritöö käigus läbi ka Camunda paigaldamine, et anda protsessimootorist täpsem ülevaade. Lisaks kirjeldatakse ära ka pisike laenu kinnitamise protsess, et anda lugejale natukene praktilist kogemust. Näited tehakse läbi kasutades Eclipse.

Autorit motiveeris antud teemal kirjutama, kuna Camunda on protsessimootorina võimekas ning kättesaadavad materjalid on inglise keelsed ja eesti keelseid väljaandeid polegi saada. Lisaks kasutatakse tänapäevases arenduses üha enam protsessimootoreid, mis teevad äriprotsesside arendamise palju lihtsamaks ja võimekamaks. Kuigi protsessimootoreid on palju siis Camunda sai valitud seetõttu, kuna autoril on selle protsessimootoriga töö alaselt kogemusi.

Lisamärkusena peab autor vajalikuks lisada, et protsessimootorist antakse ülevaatlik pilt ning kogu funktsionaalsust ei kirjeldata. Näiteks räägitakse siinses seminaritöös äriprotsesside juhtimisest, kuid jäetakse kõrvale juhtumipõhine juhtimine. Põhjus, miks selline otsus sündis on see, et Camunda juhtumipõhine juhtimine on liialt uus teema, mille kohta pole veel piisavalt infot ning seminaritöös ei jõuaks kõike täpselt lahti selgitada.

.

Võõrkeelsete lühendite loetelu

API – Application Programming Interface, rakendusliides ehk programmiliides. (vallaste.ee, 2011)

BPM – Business Process Management, äriprotsesside juhtimine.

BPMS – Business Process Management System, äriprotsesside juhtimise süsteem.

BPMN – Business Process Modeling Notation, äriprotsesside modelleerimise standard.

JUEL – Java Unified Expression Language, Java ühtne väljendus keel.

JVM – Java Virtual Machine, Java virtuaalne masin.

REST – Representational State Transfer, esinduslik seisungi ülekanne.

XML – Extensible Markup Language, laiendatav markeerimiskeel

Java – Programming language, programmeerimise keel.

Eclipse – Integrated development environment, integreeritud arendus keskkond.

User task – Kasutaja ülesanne.

Service task – Teenuse ülesanne.

Maven – Apache build manager, Apache ehituse haldur.

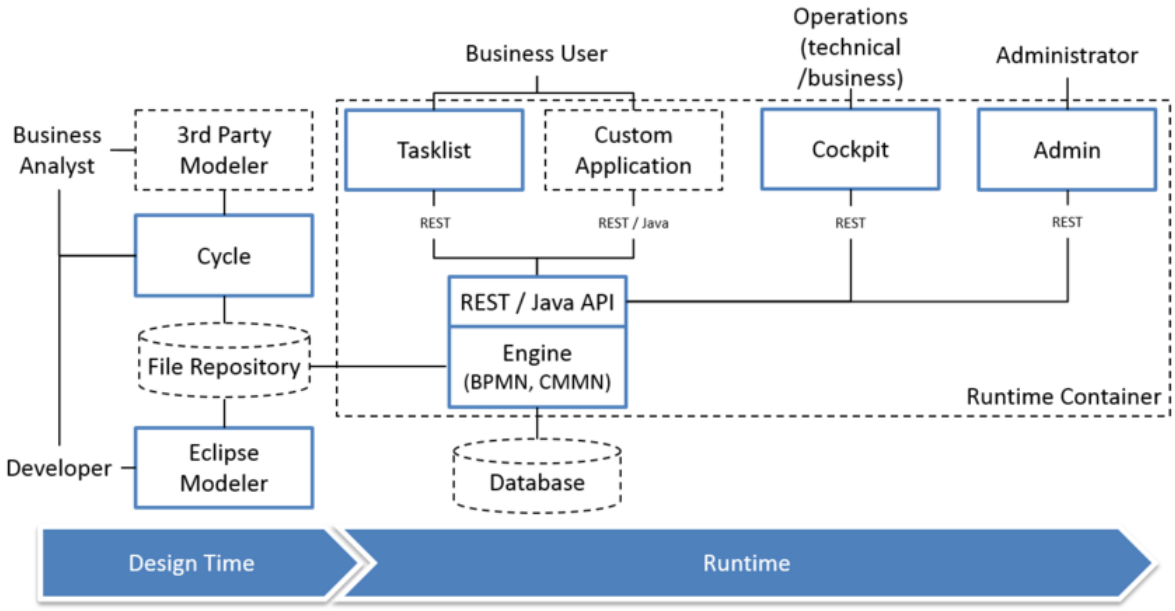
1. Camunda

Camunda BPM on Java'l põhinev avatud lähtekoodiga raamistik. Kuigi Camunda on küllaltki väike ja lihtne võimaldab see siiski täita ka kõige keerukamaid ärilisi nõudeid, nii neid disainides, kui ka käivitades.

Kõik põhikomponendid on kirjutatud Java's, kuna põhiliseks kasutajaskonnaks on mõeldud Java arendajad. Tänu sellele saavad Java arendajad kasutada omale tuttavaks saanud tööriistu, mis on vajalikud äriprotsesside disainimiseks, implementeerimiseks ning jooksutamiseks JVM'l. Näiteks, kui arendaja kasutab oma töövahendina Eclipse siis kõige mugavam on tal kasutada Eclipse BPMN pluginat, millega saab kirjeldada äriprotsesse. Tänu sellele võimalusele kaob ära ajaline kadu, mis kuluks uue tööriista installimiseks ja tundma õppimiseks. (Camunda Services GmbH, 2015: 2)

Nagu eelnevalt mainitud, siis põhiliseks sihtgrupiks on Java arendajad, kuid mõeldud on ka teistes keeltes arendajate peale, et kasutajaskond ning kasutusvõimalused protsessimootoril oleksid veegi suuremad. Selle tarbeks on loodud REST API tugi, mille abil ei pea protsessimootor olema osa arendatavast rakendusest. Lühidalt öeldes saab protsessimootor olla nii Java rakenduse üks osadest, kui ka eraldiseisvalt kuskil serveris, mille poole pöördutakse REST API teenuste kaudu.

2. Protsessimootori arhitektuur



Joonis 1. Protsessimootori arhitektuur.

Joonisel 1 on ära näidatud peamised komponendid Camunda BPM'il koos tüüpilisemate kasutajarollidega.

- **Rest / Java Api** – REST API võimaldab protsessimootorit kasutada kõrvalisest rakendusest või JavaScripti rakendusest. Tänu REST API'le ei pea protsessimootor olema osa rakendusest. (Camunda Services GmbH, 2015: 3)
- **Tasklist** – Veebirakendus User Task'ega töötamiseks. Kasutaja saab töid enda nimele võtta (claim) ning töö ise ära täita, et protsess edasi läheks või siis töö kellelegi teisele suunata (assign), kes saab siis valida, kas võtab töö endale ja teeb ise ära või keeldub tööst, misjärel antakse töö suunajale sellest teada. Lisaks User Taski'dega töötamisele sisaldab tasklist veebirakendus endas ka tööde filtreerimise võimalust, et oleks võimalik töödel paremini silma peal hoida, kuna aktiivseid ning täitmist vajavaid töid võib olla väga palju käsil. Kindlasti tuleb silmas pidada, et iga kasutaja näeb ja saab filtreerida vaid töid, millele on tal õigus olemas, kas siis

kasutaja või kasutajagrupi põhiselt. Kasutaja saab endale ise vastavad filtrid tekitada ning neid vastavalt oma soovidele nimetades ja vajalikke kriteeriumeid paika pannes. Uue kriteeriumi sisestamiseks on vajalik sisestada võti ja väärtus. Võtmete jaoks on vaikimisi valikud, kuskohast saab kategrooria põhiselt omale sobiv võti valida. Põhilisteks kategooriateks, mida valida saab on protsessi-, juhtumi-, ülesandepõhised kategooriad ning ühtlasi saab kriteeriumeid paika panna ka kasutajate, gruppide ning igasuguste protsessiga seotud aegade põhjal. Väärtuse väljale saab lisaks konkreetsele väärtusele sisestada ka mõne tingimuse, mis tuleks kirja panna JUEL keeles.

```
${ dateTime().plusDays(2) }
```

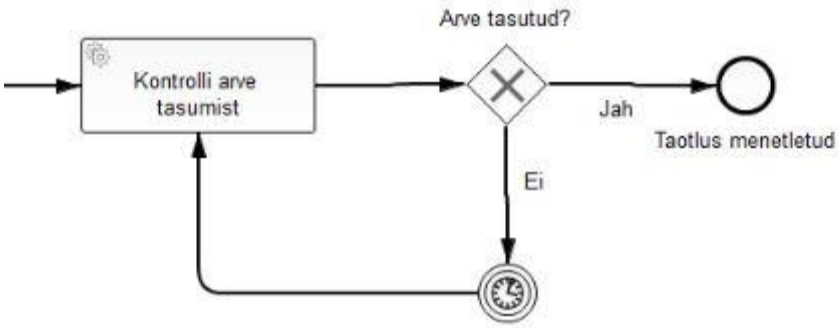
- **Cockpit** – Veebirakendus, mille eesmärgiks on monitoorida ja administreerida käimasolevaid protsesse. Näeb ära kõik käimasolevad protsessid ning nende staatused. Cockpit arhitektuur on ehitatud selliselt, et sellele saaks lisada kõiksugu lisasid suurendamaks cockpit kasutusvõimalusi ja funktsionaalsust.
- **Admin** – Administreerimiseks mõeldud veebirakendus. Siin saab hallata kasutajaid, kasutajategruppe ning nende õiguseid. Kasutajate ja kasutajagruppide haldamiseks on mootoris olemas identifitseerimise teenus (indentity service) ning õiguste tarbeks autoriseerimise teenus (authorization service). Selleks, et kasutajal oleks Camunda mõistes administreerija õigused peab ta kuuluma camunda-admin gruppi. Juhul, kui camunda-admin gruppi ei kuulu ühtegi kasutajat siis kuvatakse cockpit või tasklist veebirakendusse sisenedes vormi kasutaja tekitamiseks. Ühtlasi on administreerimise veebirakenduses olemas ka alammenüü süsteemi jaoks, kus andes ette alguse ja lõpu kuupäevad saab näha mitut protsessi on antud ajavahemikus käivitatud. Viimane alammenüü on nähtav ja kättesaadav vaid Enterprise väljalaske omanikele. Suhtlus protsessimootoriga käib läbi REST teenuste.
- **Cycle** – BPMN 2.0 protsessimudelite sünkroniseerimiseks mõeldud veebirakendus. Cycle veebirakendus võimaldab sünkroniseerida BPMN failid, millest üks on tehtud ärianalüütiku poolt tema tööriistadega ja teine on tehtud arendaja poolt, mis on tehnilisem ja sisaldab lisaks näiteks igasuguseid klasse, mis kutsutakse User või Service task'de puhul välja.

- **Modeler** – Põhiliselt kasutatakse äriprotsesside disainimiseks ja kirjeldamiseks kahte võimalust, millest üks on Eclipse plugin ja teine veebipõhine bpmn.io. Neist viimane tööriist sobib pigem ärianalüütikutele, kuna käesoleva seminaritöö kirjutamise ajal oli bpmn.io alles arendamisel ja seal ei olnud sel hetkel kuigi palju võimalusi. Näiteks puudub võimalus viidata Service Task puhul, et millise Java klassiga on see seotud. Kuid sellegipoolest saab ärianalüütik enda töö seal tehtud. Arendaja peaks kindlasti praegusel hetkel suuna võtma Eclipse plugina peale. Antud plugin võimaldab väga täpselt protsessimudelit kirjeldada.

3. BPMN

Camunda kasutab protsesside mudelina BPMN, mis on globaalne standard protsesside modelleerimisel. BPMN võimaldab luua äriprotsesse algusest lõpuni ning selle struktuurielemendid võimaldavad äriprotsessimudeli vaatajal teha kergesti vahet erinevate BPMN diagrammi osade vahel. Aina rohkem ja rohkem organisatsioone on BPMN'i kasutusele võtnud ning tänu sellele õpetatakse seda ka ülikoolides üha enam. Põhjuseid selleks, miks BPMN nii populaarseks on saanud on mitmeid.

Esimeseks põhjuseks võib kindlasti välja tuua BPMN lihtsuse. Põhimõte, kuidas BPMN on üles ehitatud on väga lihtsasti arusaadav ka neile, kes sellega varem pole kokku puutunud. Protsessimudeli disainimisel kasutatavad tähised on kiiresti õpitavad ja hästi dokumenteeritud. Heast dokumentatsioonist on kõvasti kasu, kui tekib vajadus täpsemalt aru saada, mida mingitel juhtudel kasutada ja mis on võimalikud alternatiivid. Samuti on olemas ka palju näidiseid ja seletusi, kuidas midagi teha. Näiteks käesoleva seminaritöö autoril oli oma töös headest näidetest kasu, kui tekkis olukord, mille kohta algul teadmine puudus, kuidas seda BPMN's lahendada. Antud situatsioon oli järgnev – Oli vajadus Service Task järele, mis käib teatud aja tagant kontrollimas, kas arve on tasutud ja kui ei ole siis tuleb seda kindla aja möödudes minna uuesti tegema. Seda oleks saanud ka Java's lahendada, kuid tähtsaimaks kriteeriumiks oli tingimus, et peab olema ülevaade protsessile, mis staatuses see parajasti on ning kui mitu korda on arve tasumist juba kontrollimas käidud. Seetõttu oligi vaja natukene uurida BPMN kohta, et mis võimalused seal on sellise olukorra lahendamiseks. Vastust ei pidanud kaua otsima. Võtmesõnaks oli timer, mille võimalustest autor nii täpselt ei teadnud, et seda saaks ka sealses kohas ära kasutada. Joonisel 2 on lisatud väike näide lahendusest:



Joonis 2. Tsükli kasutamise näidis.

Teiseks põhjuseks on BPMN võimekus. Nimelt võimaldab BPMN täpselt ära kirjeldada, kuidas protsess funktsioneerib. Selle kirjeldamine on küllaltki keeruline eriti, kui võtta arvesse, et põhiliseks ülesandeks on BPMN'l siiski protsessimudeli disainimine, kuid siiski tehtav.

3.1 BPMN alammudelid

Sisemised äriprotsessid (Private (internal) business processes) – Sisemised äriprotsessid on protsessid, mis on seotud konkreetse organisatsiooniga ning on protsessi tüübid, mida üldiselt nimetatakse töövooks või äriprotsesside juhtimise protsessideks.

Avalikud protsessid (Abstract (public) processes) – Avalikud protsessid on protsessid, mis kirjeldavad koostoimimist siseste äriprotsesside ja mingi muu protsessi või osaleja vahel. Avalikud tegevused sisaldavad ainult neid tegevusi, mida kasutatakse suhtlemiseks väljapoole sisemisi äriprotsesse, lisaks on kaasatud avalikesse protsessidesse sobivad voo kontrollmehhanismid. Kõiki teisi sisemisi tegevusi ei kajastata avalikes protsessides. Avalik protsess näitab väljapoole sõnumite jada, mida on vaja, et suhelda äriprotsessiga.

Koostööprotsessid (Collaboration (global) Processes) – Need protsessid kujutavad kahe või enama ettevõtte üksuste vahelist koostoimimist. Koostoimed on määratletud kui tegevuste jadad, mis kirjeldavad sõnumiedastusmustreid seotud üksuste vahel. Üksik koostööprotsess võib olla vastandatud erinevate koostöökeeltega. (Object Management Group, 2015: 6)

3.2 BPMN elemendid

BPMN struktuurielemendid jagunevad nelja kategooriasse ning need jagunevad omakorda alamkategooriatesse.

- **Voogobjektid** (Flow objects) – Voogobjektid on peamised graafilised elemendid, määratlemaks äriprotsesside käitumist. Voogobjektid jagunevad omakorda kolmeks elemendiks:
    - **Sündmused** (Events) – Sündmuseks on element, millega märgitakse millegi toimumist äriprotsessis. Sündmused jagunevad omakorda kolmeks: Algsündmus (Start), vahesündmus (Intermediate) ja lõppsündmus (End).
    - **Tegevused** (Activities) – Tegevuseks nimetatakse elementi, millega märgitakse tegevust äriprotsessis. Jagunevad samuti omakorda kolmeks: Protsess (Process), alamprotsess (Sub-Process) ja ülesanne (Task).
    - **Väravad** (Gateways) – Väravaks nimetatakse elementi, millega märgitakse voo lahknemisi ja ühinemisi.
- **Ühendavad objektid** (Connecting Objects) – Ühendatavaid objekte kasutatakse erinevate voogobjektide ühendamiseks. Voogobjektid jagunevad omakorda kolmeks vooks:

    - **Järgnevusvoog** (Sequence Flow) – Voog, millega näidatakse tegevuste sooritamise järjekorda.
    - **Sõnumvoog** (Message Flow) – Voog, millega näidatakse voogu kahe osapoole vahel, mis on valmis saatma ja vastu võtma.
    - **Ühendus** (Association) – voog, mida kasutatakse informatsiooni ühendamiseks voogobjektidega.
- **Ujumisrajad** (Swimlanes) – Ujumisradu kasutatakse modellerimiselementide grupeerimiseks. Ujumisrajad jagunevad omakorda kaheks:
    - **Basseinid** (Pool) – näitab protsessi osalejat.
    - **Rada** (Lane) – Basseini alamosa, mis võib laieneda kogu Basseini ulatuses nii vertikaalselt, kui ka horisontaalselt.
- **Tehised** (Artifacts) – Tehiseid kasutatakse protsessi kohta täiendava informatsiooni andmiseks. Tehised jagunevad omakorda kolmeks:
    - **Andmeobjekt** (Data Object) – Andmeobjektid annavad teavet, mida tegevused vajavad või toodavad.
    - **Grupp** (Group) – Kasutatakse tegevuste grupeerimiseks sama kategooria siseselt.
    - **Annotatsioon** (Annotation) – Kasutatakse lisainfo andmiseks BPMN diagrammi lugejale. (Camunda Services GmbH, 2015: 5)

4. Camunda paigaldamine

Järgnevalt vaatame, kuidas paigaldada Camunda BPM platvorm. Alustuseks tuleks kontrollida, kas kõik nõuded on selleks täidetud, et kogu protsess algusest lõpuni edukalt läheks:

- Java JDK 1.6+
- Veebibrauser – Firefox, Chrome või Internet Explorer versiooniga 9+
- Eclipse, millel oleks BPMN plugin

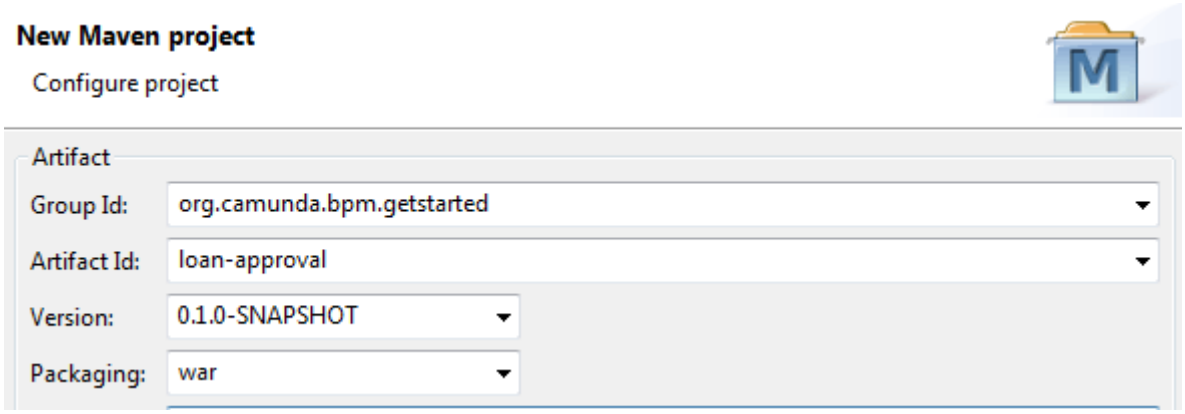
Kui kõik tööriistad on olemas ja installitud tuleks alla tõmmata Camunda BPM platvorm. Valida saab mitmete erinevate distributsioonide vahel, mis on mõeldud erinevatele serveritele, kuid antud hetkel tuleks valida Apache Tomcat'l põhinev distributsioon. Peale allalaadimist pakkida lahti omale sobivasse kohta. Kuna hiljem läheb selle kausta asukohta vaja siis kutsuks seda kausta edaspidiselt `$CAMUNDA_KODU`. Alla laaditud Camunda BPM platvorm on vajalik, et käivitada enda loodud protsesse, mis on war laienditega. Sellised failid tekivad, kui tekitada Maven projekt ning peale seadistusi ja protsessi kirjeldamist `Maven Install` teha, mis loobki vajaliku war laiendiga faili. Sellest kõigest räägime lähemalt järgmises peatükis, kus teeme näidisrakenduse nig käime selle projekti läbi.

Juhul, kui kõik on läinud edukalt siis peaks Windows operatsioonisüsteemi kasutajatel peale `start-camunda.bat` ning Linuxi kasutajatel peale `start-camunda.sh` käivitamist avanema veebibrauseris automaatselt Camunda koduleht, mis õnnitleb, et olete edukalt installinud Camunda BPM platvormi – Joonis 3. Veebileht avaneb localhost's 8080 pordil. (Camunda Services GmbH, 2015: 4)

Joonis 3. Camunda avaleht.

5. Näidis protsessi koostamine

Olles edukalt installinud Camunda BPM platvormi on aeg edasi liikuda Eclipse poole, kus tuleks minna `File / New / Other` ning avanevas `New Project Wizard` aknas valida `Maven / Maven Project` ja seejärel vajutada `Next`. Järgnevas aknas linnutada ära `Create a simple project` ja samuti vajutada `Next`. Edasi tuleb Maven projekti jaoks ära konfigureerida. Kindlasti panna rõhku, et valitud oleks `war`, kuna paneme püsti WAR projekti. Peale konfigureerimist vajutada `Finish` ning Eclipse seab püsti uue Maven projekti. Joonisel 4 on näha, kuidas peaks konfigureerimine olema.



Joonis 4. Maven konfigureerimine.

Järgmise sammuna tuleks Maven'le teada anda Camunda'st ning seda saab teha läbi Maven sõltuvuste (dependencies). Sõltuvuste lisamise tarbeks on pandud LISA1 alla kõik vajalikud sõltuvused, mis tuleb lisada `pom.xml` faili. Nüüd on võimalus teha esimene „ehitus" (build) – selleks tuleb muudetud `pom.xml` peal teha parem klikk ning seejärel `Run As / Maven install`.

Liidese tekitamiseks rakenduse ja protsessimootori on vajalik tekitada protsessi rakenduse klass (Process Application) ning selleks lisada org.camunda.bpm.getstarted.loanapproval kaust (package) ja järgnev klass:

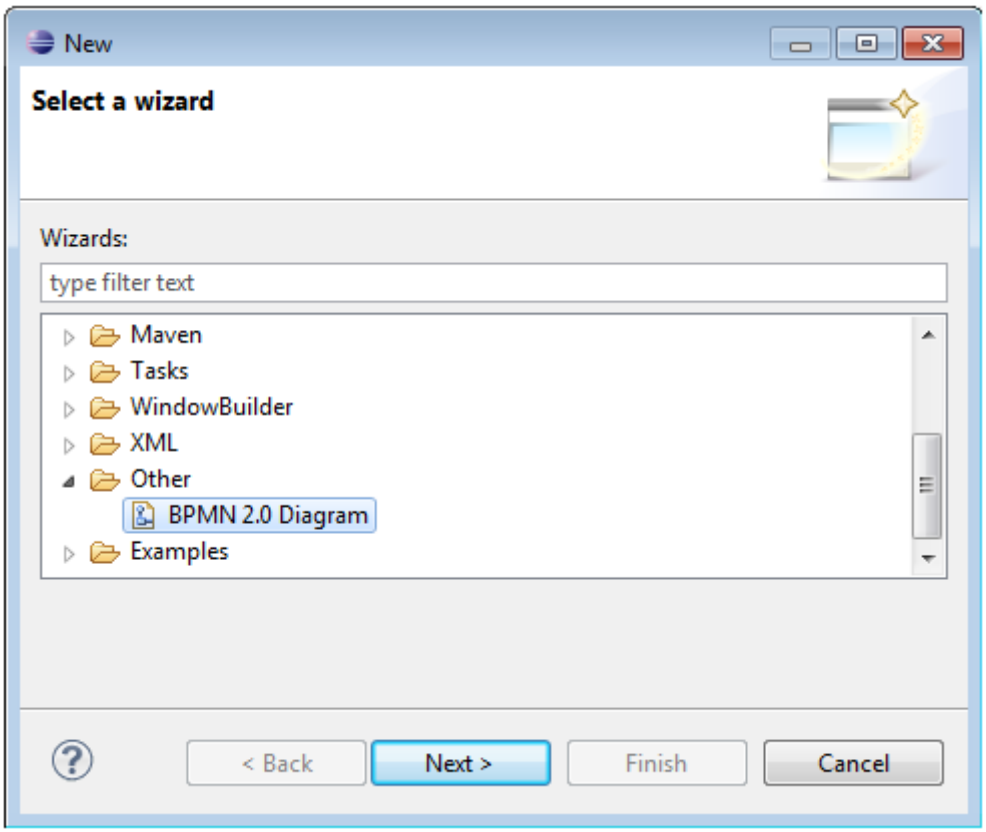
```
package org.camunda.bpm.getstarted.loanapproval;

import org.camunda.bpm.application.ProcessApplication;
```

```
import
org.camunda.bpm.application.impl.ServletProcessApplication;

@ProcessApplication("Loan Approval App")

public class LoanApprovalApplication extends
ServletProcessApplication {}
```

Viimase sammuna, et seadistada protsessi rakendust on vaja lisada `src/main/resources/META-INF/processes.xml` kasutuselevõttu kirjeldav fail. Antud fail võimaldab meil ära deklareerida seadistused, mis teevad protsessi rakendust käivitades sellest protsessi mootori. Ühtlasi on võimalik `processes.xml` ka tühjaks jätta, mille käigus võetakse kõik väärtused vaikimisi.

```
<?xml version="1.0" encoding="UTF-8" ?>

<process-application

  xmlns="http://www.camunda.org/schema/1.0/ProcessApplication"

  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <process-archive name="loan-approval">

    <process-engine>default</process-engine>

    <properties>

      <property name="isDeleteUponUndeploy">false</property>

      <property
name="isScanForProcessDefinitions">true</property>

    </properties>

  </process-archive>

</process-application>
```

Nüüdseks on protsessi rakendus edukalt seadistatud ja saab edasi minna järgmise sammuga, milleks on ärimudeli tegemine ehk tekitame BPMN faili, et siis selle abil tekitada `war` laiendiga fail. Selleks minna `src/main/resources` kausta peale ning peale paremat klikki valida `New / Other` ning seejärel `BPMN 2.0 Diagram` nagu on näha ka joonisel 5. Lisatavale BPMN failile tuleks anda nimeks `loan-approval.bpmn`. Juhul, kui miskipärast ei saa valida BPMN diagrammi tuleks veenduda, et Eclipse BPMN plugin on olemas.



Joonis 5. BPMN faili loomine.

Avanenud äriprotsessi disainimise aknas on vajalik antud näite põhjal töölauale tõsta `Start-Event`, `User Task` ja `End-Event` ning need ühendada `Sequence Flow`'ga nagu joonisel 6 näha:

Joonis 6. Äriprotsessi mudel.

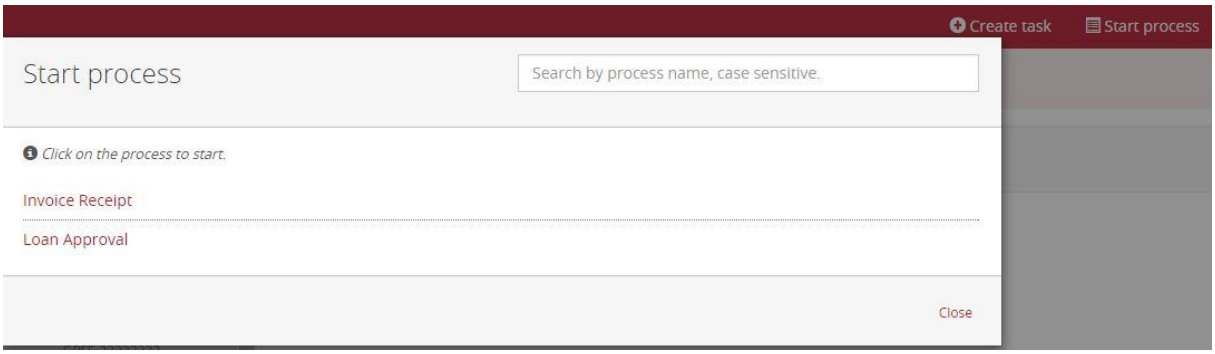
Kuna me modelleerime välja kutsutavat protsessi siis tuleks `properties` alajaotises isExecutable ära linnutada ning ühtlasi ka protsessi id määrata `approve-loan`. Vajaliku alajaotise leiab, kui vajutada kuskile tühja koha peale mis järel avanebki õige asi.

Edasi tuleks Maven'ga valmis ehitada war fail. Selleks tuleb uuesti `pom.xml` peal `Maven Install` teha, mis tekitab `target/` kausta `loan-approval-0.0.1-SNAPSHOT.war` faili. Nüüd jõuabki kätte koht, mil vajame `$CAMUNDA_KODU` kataloogi. Äsja loodud war fail tuleks panna `$CAMUNDA_KODU/server/apache-tomcat/webapps` kausta, et see sealt käivituks. Konsooli aknast, mis avanes `start-camunda.bat` käivitamisest tuleks järele vaadata, kas õnnestus.

Õnnestumise korral minnes `http://localhost:8080/camunda/app/cockpit` lehele ja logides sisse `demo / demo` kasutajaga on näha protsessi, mille tekitasime. Cockpit's on näha ühtlasi ka praegusel hetkel käimas olevaid protsesse versiooni põhiselt, kuna versioone antud protsessist võib olla erinevaid. Seda on näha joonisel 7. Näiteks mõndade protsesside muutumine on tingitud riiklike seaduste muutumisest ja seetõttu peab olema protsessist uus versioon, kuid samas ka vana peab alles jääma, et näha ajalugu, kui selleks peaks vajadus tekkima.

Joonis 7. Käimasolevad protsessid.

Tasklistile saab ligi minnes `http://localhost:8080/camunda/app/tasklist`. Seal saab uusi protsesse alustada nagu näha ka joonisel 8. Protsessi alustamiseks tuleb vajutada `Start process` ja seejärel avaneb loetelu kõikidest protsessidest, mis antud serveril olemas on.



Joonis 8. Uue protsessi alustamine.

Protsesse saab kustutada ja lisada, kui minna `$CAMUNDA_KODU/server/apache-tomcat/webapps` kausta ja seal vastavad war failid ära kustutades või siis uusi juurde lisades. Järgmises aknas vajutada lihtsalt `Start` nupule ning protsess algab. Alustatud protsess on nähtav `All Tasks` loetelus. Demo kasutajaga saab näha, mis staadiumis protsess parajasti on, kuid ei saa protsessi täita, kuna protsess on suunatud John kasutajale. Liikudes John kasutajaga samamoodi loodud protsessi peale nagu läksime enne Demo kasutajaga saab protsessi lõpetatuks nimetada vajutades `Complete` nagu joonisel 8 näha. Seejärel ongi protsess täidetuks kuulutatud ja inimene saab laenu võtta. Protsessile võib lisada iseseisvalt näiteks `gateway`, et protsess tõetruum oleks tänu tingimusele, et seal on summa alla mille saaks laenu ja üle mingi kindla summa ei saaks.

Joonis 8. Protsessi lõpetamine.

Kokkuvõte

Käesoleva seminaritöö eesmärgiks oli anda ülevaade Camunda protsessimootorist ning selle käigus teha läbi ka näidis protsess, mille käigus valmib lihtne laenu kinnitamise protsess. Protsessi olemus oli lihtne. Tekitati laenu taotlus ning seejärel sai seda kinnitada. Lisaülesandena pakuti lugejale protsessi täiendamist, et protsess võtaks arvesse laenu võtmise summa suurust ning kui summa on liialt suur siis laenu ei saaks võtta.

Kuna enne näite tegemist on hea ja samuti ka vajalik omandada mõningaid algteadmiseid protsesside olemusest siis kõigepealt räägiti algul seminaritöös Camunda'st ning tolle arhitektuurist. Läbi võeti ka BPMN alammudelid ja elemendid, mis olid vajalikud protsessi kirjeldamiseks ning disainimiseks.

Autori hinnangul on seminaritöö oma eesmärgi täitnud, kuna seminaritöö käigus jõuti püstitatud eesmärgini. Lisaks andis seminaritöö autorile ka olulisel määral uusi teadmisi juurde nii Camunda, kui ka BPMN kohta, mida autor saab oma töös kasutada. Näiteks tuli lisateadmisi juurde protsessimootori arhitektuuri kohapealt ning samuti olid ka osad BPMN elemendid võõrad.

Kasutatud kirjandus

1. Camunda Services GmbH. (2015) User guide. Loetud 24.10.2015 aadressil: https://docs.camunda.org/manual/7.3/guides/user-guide

2. Camunda Services GmbH. (2015) Process Engine. Loetud 24.10.2015 aadressil: https://docs.camunda.org/manual/7.3/guides/user-guide/#process-engine

3. Camunda Services GmbH. (2015) Architecture. Loetud 24.10.2015 aadressil: https://docs.camunda.org/manual/7.3/guides/user-guide/#introduction-architecture-overview

4. Camunda Services GmbH. (2015) Install. Loetud 25.10.2015 aadressil: https://docs.camunda.org/get-started/bpmn20/install/

5. Camunda Services GmbH. (2015) BPMN Model Api. Loetud 25.10.2015 aadressil: https://docs.camunda.org/manual/7.3/guides/user-guide/#bpmn-model-api

6. Object Management Group. (2015) BPMN. Loetud 25.10.2015 aadressil: http://www.bpmn.org/

LISA1

```
<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>7.3.0</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>
```