**MANUAL D’INSTAL.Lació i configuració de Directives**

**ÍNDEX**

[1. Sigles i definicions 3](#_Toc533343951)

[1.1 Sigles 3](#_Toc533343952)

[1.2 Definicions 3](#_Toc533343953)

[2. Introducció 4](#_Toc533343954)

[3. Instal·lació d’Activiti 5](#_Toc533343955)

[3.1 Requisits 5](#_Toc533343956)

[3.1.1 Instal·lació JDK 5](#_Toc533343957)

[3.1.2 Instal·lació Apache Tomcat 6](#_Toc533343958)

[3.2 Instal·lació d’Activiti BPM 6](#_Toc533343959)

[4. Configuració de la base de dades 7](#_Toc533343960)

[4.1 Instal·lació i configuració de MariaDB 7](#_Toc533343961)

[5. Creació d’un flux de treball 10](#_Toc533343962)

[5.1 Activiti Modeler 10](#_Toc533343963)

[5.2 Aplicació Java 11](#_Toc533343964)

[6. Components disponibles 14](#_Toc533343965)

[6.1 Inici d’esdeveniments 14](#_Toc533343966)

[6.2 Gateways 14](#_Toc533343967)

[6.3 Tasques 15](#_Toc533343968)

[7. Exemple de flux de treball genèric 16](#_Toc533343969)

[7.1 Creació del diagrama 16](#_Toc533343970)

[7.2 Desplegament del diagrama 18](#_Toc533343971)

[7.3 Iniciar flux 20](#_Toc533343972)

[7.4 Gestió de fluxes de treball 21](#_Toc533343973)

[8. REST API 22](#_Toc533343974)

[8.1 Instal·lació 22](#_Toc533343975)

[8.2 Mètodes 22](#_Toc533343976)

[8.3 Recursos 22](#_Toc533343977)

[9. Exemple de fluxos de treball tècnics 25](#_Toc533343978)

[9.1 Explicació del diagrama 25](#_Toc533343979)

[10. Exemples de fluxos de treball funcionals 28](#_Toc533343980)

[10.1 Enviament d’emails 28](#_Toc533343981)

[10.2 Generació d’informes 29](#_Toc533343982)

[10.3 Enregistrament d’execucions 30](#_Toc533343983)

[10.4 Report diari 31](#_Toc533343984)

# Sigles i definicions

## Sigles

A continuació, trobareu una llista ordenada alfabèticament de les sigles que s'utilitzen en aquest document.

| **Terme** | **Definició** |
| --- | --- |
| BPM | Business Process Management (Gestió del procés de negoci) |
| BPML | Business Process Management Language (Idioma del procés de negoci) |
| JDBC | Java Database Connectivity |
| JDK | Java Development Kit |
| WAR | Web Application Archive |

## Definicions

A continuació, trobareu una llista ordenada alfabèticament dels termes que s'utilitzen en aquest document.

| **Terme** | **Definició** |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Introducció

El present document fa referència al mòdul Directives i inclou la definició i implementació de les mateixes. El seu abast és el següent:

* Instal·lació de l’eina open source Activiti BPMN 2.0.
* Configuració d’una base de dades externa.
* Creació d’un diagrama referent a un diagrama de flux de treball.
* Components disponibles.
* Exemple de flux de treball genèric.
* Exemple de flux de treball funcional aplicat al CityOS.

La primera part del document tracta de documentar els passos a seguir per tal de començar a treballar amb l’eina escollida en un entorn de producció.

A continuació es fa un breu resum de com es pot crear una nova directiva i les principals eines que s’ofereixen dins d’Activiti.

Per últim, es realitza un exemple de diagrama a on s’integren els diferents components que formen el flux de treball i es presenten diversos exemples de fluxos de treball que generen directives aplicades a un projecte.

# Instal·lació d’Activiti

## Requisits

Activiti BPM és una eina desenvolupada amb Java que es desplega en qualsevol servidor d’aplicacions.

Segons la web del fabricant, els requisits són els següents:

* Servidor d’aplicacions Apache Tomcat o similar.
* Oracle JDK igual o superior a la versió 7.

## Instal·lació JDK

La instal·lació del JDK 7/+ es realitza mitjançant “alternatives” de Centos, descarregant el binari corresponent des d’Oracle.com.

<https://wiki.centos.org/HowTos/JavaRuntimeEnvironment>

# alternatives --install /usr/bin/java /usr/java/latest/bin/java 5

# alternatives --config java

There are 5 programs which provide 'java'.

Selection Command

-----------------------------------------------

1 /usr/lib/jvm/jre-1.4.2-gcj/bin/java

2 /usr/java/jre1.6.0\_13/bin/java

3 /usr/java/jre1.6.0\_18/bin/java

4 /usr/lib/jvm/jre-1.6.0-openjdk.x86\_64/bin/java

5 /usr/java/latest/bin/java

Enter to keep the current selection[+], or type selection number: 5

# java -version

java version "1.6.0\_26"

Java(TM) SE Runtime Environment (build 1.6.0\_30)

Java HotSpot(TM) 64-Bit Server VM (build 20.1-b02, mixed mode)

## Instal·lació Apache Tomcat

La instal·lació d’Apache Tomcat s’ha realitzat mitjançant el repositori oficial de Centos i usant l’última versió disponible d’aquest.

## Instal·lació d’Activiti BPM

La instal·lació d’Activiti BPM consisteix en la descàrrega, descompressió del paquet subministrat i còpia dels fitxers WAR en el directori “webapps” d’Apahe Tomcat. Arribat a aquest punt, per tal d’iniciar l’aplicació podem reiniciar el servidor d’Apache Tomcat o executar-la directament des del manager.

# Configuració de la base de dades

Per defecte, Activiti ve configurada per treballar en un entorn de demostració del producte.

Per realitzar les proves, s’utilitza una base de dades H2 en memòria; és a dir, que les dades són temporals.

Per a un correcte funcionament productiu, s’ha de configurar una base de dades que no desi les dades a la memòria, ja sigui amb una instal·lació standalone H2 o amb qualsevol altre base de dades suportades per Activiti.

La llista de tipus de base de dades suportades pot variar entre versions, per la qual cosa es recomana consultar la guia de l’usuari de la versió corresponent. Per a la versió 5.22.0, que és la que hem utilitzat, els tipus de base de dades suportats són els següents:

* H2
* Mysql
* Oracle
* Postgre
* DB2
* Mssql

## Instal·lació i configuració de MariaDB

La instal·lació de MariaDB es realitza des del repositori oficial de Centos 7:

yum install mariadb mariadb-server

Posteriorment s’ha de configurar mitjançant la següent comanda:

mysql\_secureinstall

Existeixen dues maneres de configurar la base de dades que utilitzarà el motors d’Activiti:

* La primera opció, és definint les propietats JDBC de la base de dades. Per això s’ha de modificar el fitxer “db.propierties” que es troba a la ruta *{TOMCAT\_HOME}/webapps/activiti-explorer/WEB-INF/classes/.*

db=alfresco

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/alfresco

jdbc.username=root

jdbc.password=C1ty0s

Imatge 1. Fitxer db.propierties

* La segona opció és mitjançant la implementació d’un javax.sql.Datasource.

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" >

<property name="driverClassName" value="com.mysql.jdbc.Driver" />

<property name="url" value="jdbc:mysql://localhost:3306/activiti" />

<property name="username" value="activiti" />

<property name="password" value="activiti" />

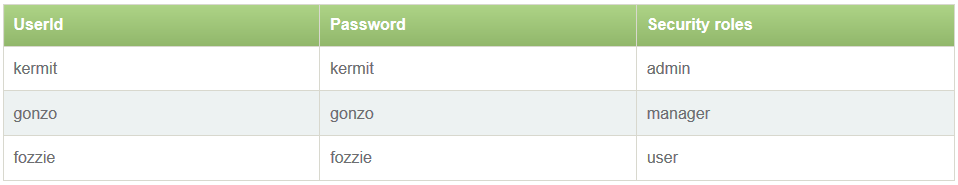
<property name="defaultAutoCommit" value="false" />

</bean>

Imatge 2. Exemple extret de la guia d'usuari

Un cop configurada la connexió, es pot iniciar l’aplicació per a què es creïn de forma automàtica les taules de la base de dades.

Com hem comentat anteriorment, Activiti està preparat per treballar en un entorn de demo per defecte, per la qual cosa, en crear les taules, es crearan també processos, usuaris i grups d’exemple.



Imatge 3. Usuaris creats en l'entorn de demostració

Si és volgués cancel·lar la creació dels elements de l'entorn d'exemple, s’hauria de modificar el fitxer “engine.propierties” de la ruta *{TOMCAT\_HOME}/webapps/activiti-explorer/WEB-INF/classes/* i modificar els valors de les variables a fals per tots aquells elements que no vulguem crear per defecte.

Imatge 4. Fitxer engine.propierties

# demo data properties

create.demo.users=true

create.demo.definitions=true

create.demo.models=true

create.demo.reports=true

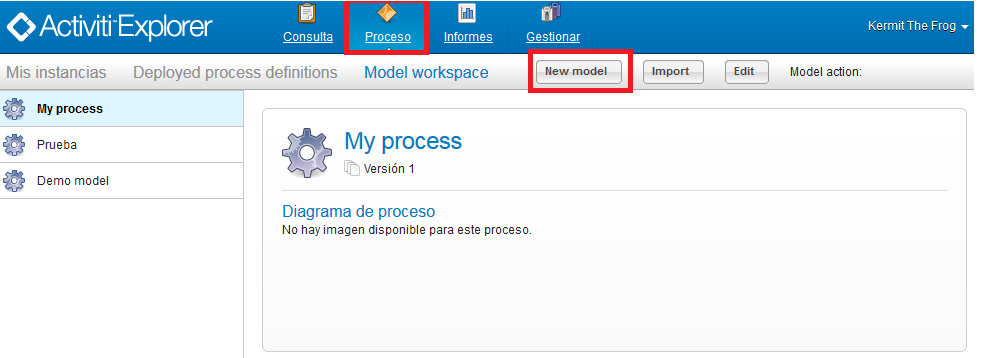
# Creació d’un flux de treball

Existeixen dues maneres de crear nous fluxos de treball. Una manera és des de la mateixa interfície gràfica que existeix dins de l’aplicació web i l’altre, és mitjançant la creació d’una aplicació Java.

## Activiti Modeler

Des de la interfície gràfica ens permet definir un flux de treball usant l’eina de modelatge o mitjançant la definició d’una taula de tasques.

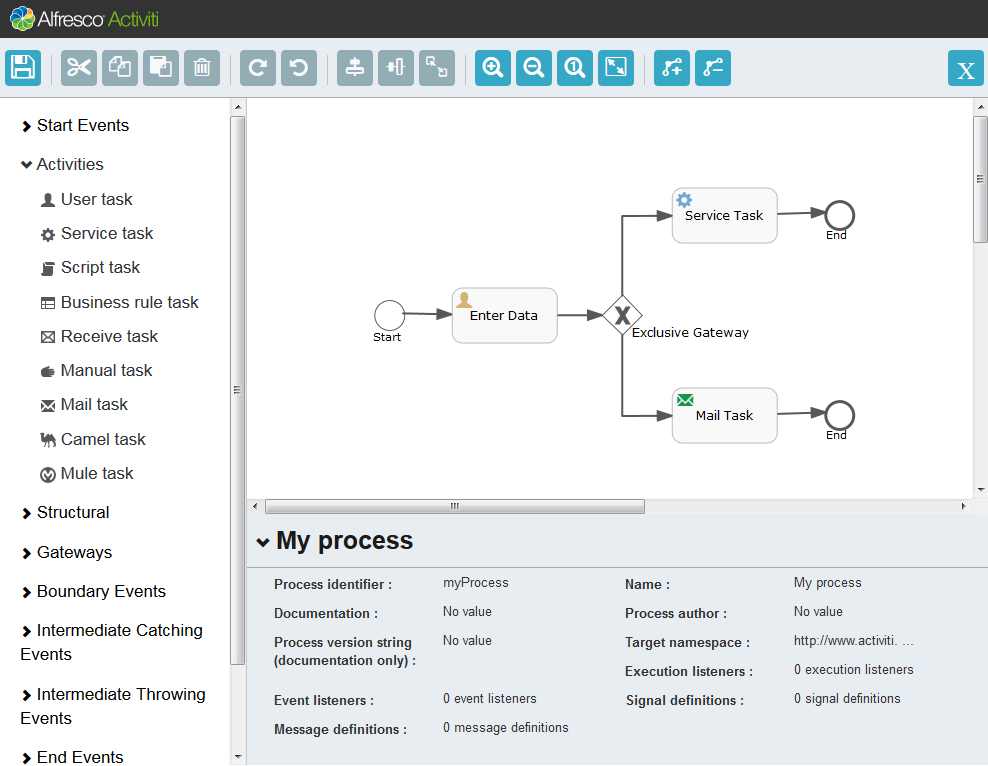
La forma d’accedir a aquestes eines és diferent segons la versió d’Activiti que s’estigui fent ús. En el nostre cas, l’accés es fa mitjançant la url *{Host}:{Port}/activiti-explorer* per accedir a la pàgina principal de l’aplicació i des d’aquí haurem de fer clic sobre el botó “New Model” de la pestanya “Proces” per iniciar el procés de creació d’un nou flux de treball.



Imatge 5. Creació d'un nou model

Es poden distingir quatre seccions diferents dins de la finestra que apareix:

* En la part superior, s’hi troben les eines típiques d’edició, tals com desar, copiar, enganxar, etc.
* En el lateral esquerre, hi ha els diferents components que es poden utilitzar per definir el diagrama.
* En la part central, tenim l’àrea de treball del diagrama.
* En la part inferior, hi apareixen les diferents propietats del component que tinguem seleccionat de l’àrea de treball.

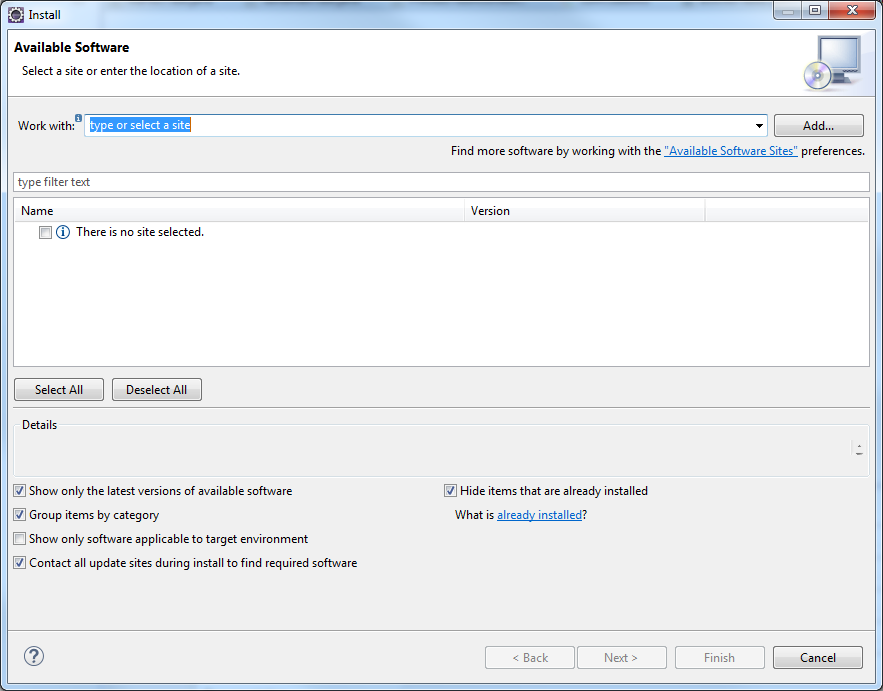


Imatge 6. Activiti Modeler versió web

## Aplicació Java

Per a la realització d’aquest apartat hem utilitzat Eclipse, juntament amb la col·lecció de paquets Mars2 i els JDK 6 i 8.

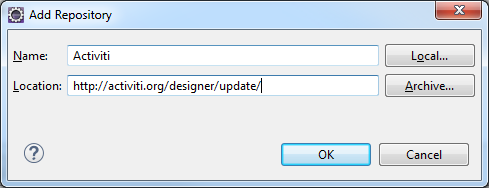
Una vegada hàgim instal·lat Eclipse, s'ha d'afegir el pluguin d’Activiti. Per fer-ho, haurem d’anar a la pestanya “Help” 🡪 “Install Software”, i una vegada aparegui la finestra emergent, prèmer sobre el botó “Add...”.



Imatge 7. Install software d'Eclipse

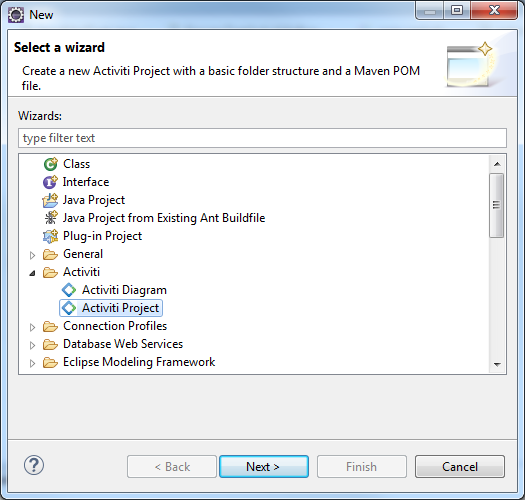
A continuació, haurem d’assignar un nom al nou repositori i afegir la següent url:

<http://activiti.org/designer/update/>



Imatge 8. Afegint el repositori Activiti

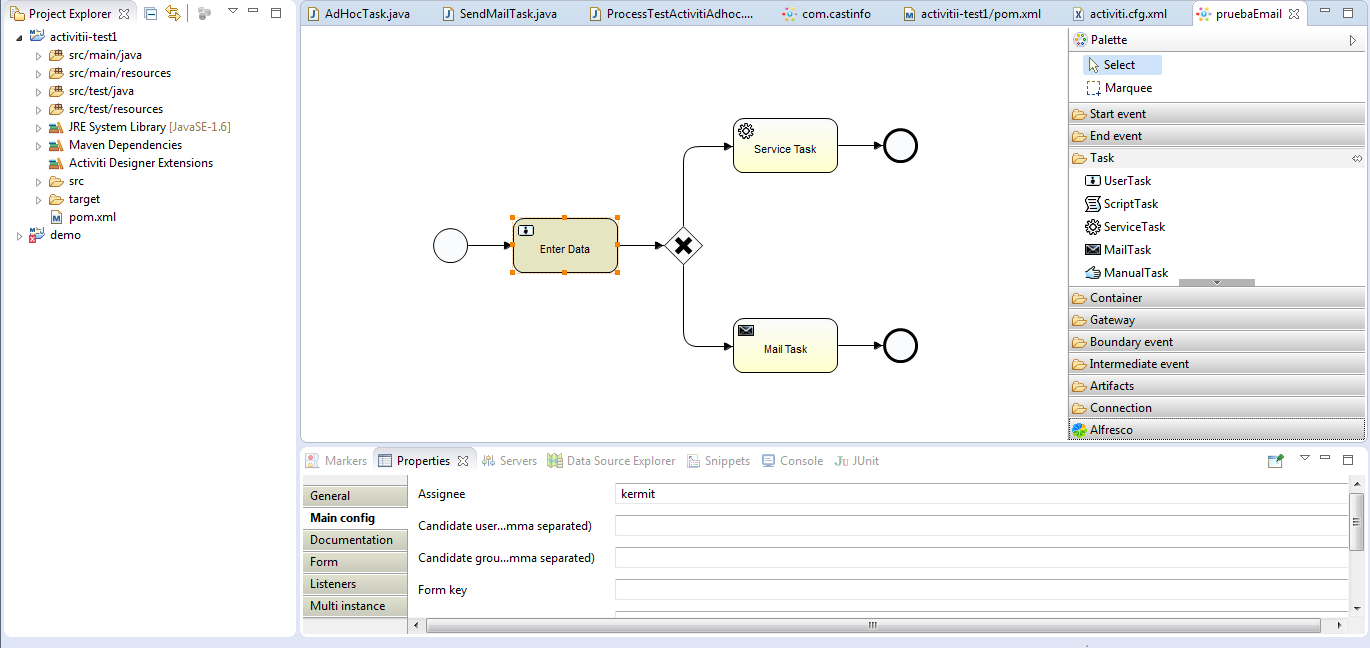
Una vegada hàgim finalitzat el procés, podrem crear nous projectes Acitivi mitjançant el menú principal: “File” 🡪 “New” 🡪 “Other” i escollir l’opció “Activiti Project” del directori “Activiti”.



Imatge 9. Nou projecte Activiti

Aquesta opció crearà l’arbre de directoris d’una aplicació Activiti. Es recomana crear els diagrames dels fluxos dins del directori */src/main/java/resources/diagrams.*

Un cop creats, obtindrem una visió molt similar a la que s’ofereix a la versió web, amb accés als diferents components i a les propietats de cadascun d’aquests.



Imatge 10. Definició d'un diagrama de flux des d'Eclipse

# Components disponibles

Activiti ofereix una àmplia diversitat de components per definir un flux de treball, dels quals explicarem els que considerem que tenen més rellevància.

## Inici d’esdeveniments

Existeixen diferents iniciadors d’esdeveniments, els quals s’utilitzen per indicar què o qui és l’encarregat d’iniciar el procés.

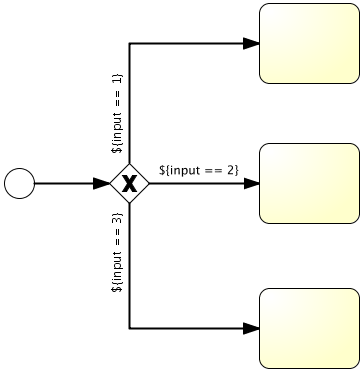
**StartEvent**: S’utilitza per indicar que el disparador de l’esdeveniment s’iniciarà després de l’acció d’un usuari.

**TimerEvent**: Aquest tipus de disparador es pot emprar tant per iniciar un procés com en un procés intermedi. Les principals característiques d’aquest component són que se li pot assignar una hora d’inici, un compte enrere i un interval de repetició.

**MessageEvent** i **Signal Event**: Són dos tipus de disparadors que s’executen al rebre un missatge o un senyal respectivament. La diferència entre ells és que el senyal s’envia a tots els components que estiguin esperant rebre-la; i en canvi, el missatge té un destinatari concret.

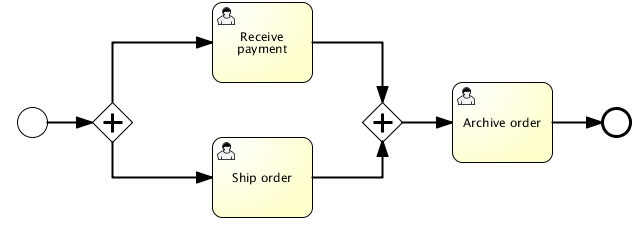
## Gateways

 **Exclusive Gateway**: Serveix per prendre una decisió del camí a seguir.



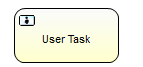
Imatge 11. Exclusive Gateway

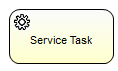
**Parallel Gateway**: Permet l’execució de diferents fils de manera concurrent obrint diferents camins d’execució. També té la possibilitat d’agrupar diferents fluxos en un de sol.



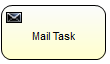
Imatge 12. Parallel Gateway

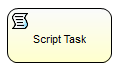
## Tasques

**UserTask**: Aquest component fa referència a un tipus d’activitat que necessita ser realitzada per interacció humana. Existeixen tres maneres d’assignar aquesta tasca: a través del nom específic de l’usuari, mitjançant el nom de possibles candidats o bé, mitjançant el nom d’un grup.

 **ServiceTask**: Aquest component s’utilitza per fer crides a funcions de Java externes. Existeixen quatre formes d’invocar-les:

* Especificant una classe que implementi JavaDelegate o ActivityBehavior;
* Avaluant una expressió que es resolgui en un objecte de delegació;
* Invocant una expressió de mètode;
* Avaluant una expressió de valor.

 **MailTask**: Activiti permet l’enviament d’emails a un o més destinataris amb les opcions típiques tals com CC, BCC, contingut HTML, etc. Per poder fer ús d’aquesta opció, s’ha de disposar d’un servidor de correus extern amb capacitat SMTP i per configurar-lo, s’han de modificar les propietats del fitxer “activiti.cfg.xml”.

**ScriptTask**: És un procés automàtic que s’encarrega de llençar un script en un format conegut. Per defecte, Javascript està inclòs en cada JDK, per la qual cosa, no és necessari afegir cap .jar addicional. Si es vol utilitzar un altre tipus de motor de scripts, només cal afegir el .jar corresponent.

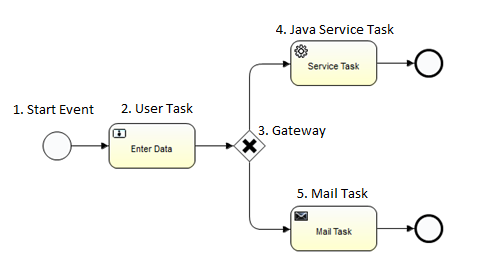
# Exemple de flux de treball genèric

## Creació del diagrama

Un exemple de directiva podria ser l’enviament d’una alerta via correu electronic en el moment que es detecta un error en les dades noves que arriben a la plataforma.

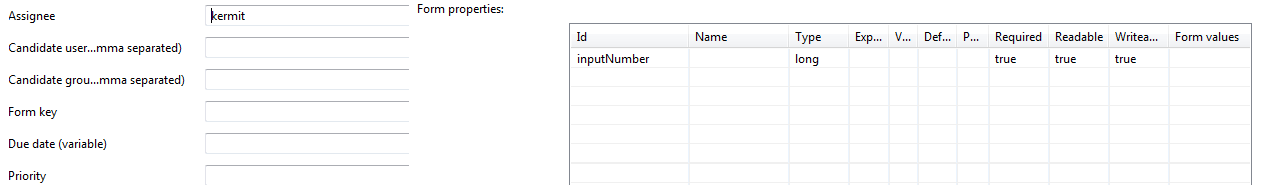
S’ha realitzat un exemple a on després de rebre la informació de l'usuari, es prèn la decisió de fer la crida a una funció Java o fer l'enviament del correu.

Una altra alternativa, hauria estat posar el procés Java abans del gateway i que s’encarregués de fer la validació de les dades.



Imatge 13. Exemple de flux de treball

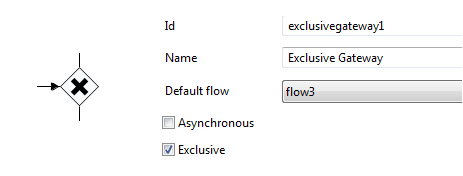
* **Start Event:**  S’ha configurat aquest exemple per a què l’encarregat d’iniciar el procés sigui per interacció de l’usuari.
* **User Task:** En aquest pas, arribarà una nova consulta a l’usuari o usuaris candidats escollits. La consulta consisteix en un formulari sol·licitant les dades especificades a les propietats del component. En el cas de l’exemple, s’ha assignat la tasca a l’usuari “Kermit” i se li ha sol·licitat que entri un número que es desarà a la variable “inputNumber”.



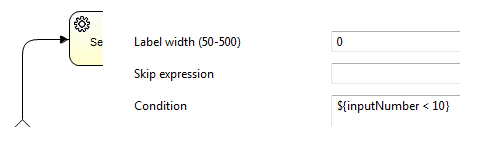
Imatge 14. Propietats User Task

* **Gateway:** Per realitzar la prova dels encaminadors, s’ha implementat una lògica molt senzilla que permet comprovar el funcionament d'aquests. En arribar a aquest punt, es comprovarà si a l'acció anterior s’hi ha inserit un número inferior a 10, o si pel contrari, s’hi ha inserit un número igual o superior a 10. En el primer cas, s'enviarà un email informant del valor introduït. En el segon cas, es realitzarà una crida a una funció Java que s'encarregarà de llegir i escriure sobre la variable "inputNumber".

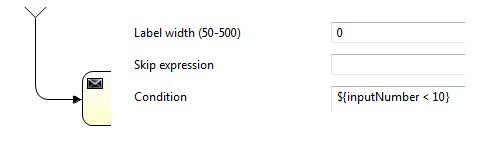
El component en sí no permet aplicar cap lògica del camí a seguir, sinó que s'encarrega d'agrupar tots els possibles camins i declarar qui hauria de ser el camí per defecte. La lògica s'ha d'aplicar directament als enllaços que uneixen el component Gateway amb el següent component:



Imatge 15. Opcions del Gateway



Imatge 16. Opcions del camí cap al servei java



Imatge 17. Opcions cap a l'enviament de l’email

* **Java Service Task:** En aquest procés es veu com es fa la crida a una funció Java que s'ha creat per realitzar aquest exemple, amb la qual, es recull el valor de la variable que ha introduït l'usuari, es treu per pantalla i s'assigna un valor nou.

package com.[project\_name].bpmn;

import org.activiti.engine.delegate.BpmnError;

import org.activiti.engine.delegate.DelegateExecution;

import org.activiti.engine.delegate.Expression;

import org.activiti.engine.delegate.JavaDelegate;

public class ReadNumberTask implements JavaDelegate {

private Expression number;

public Expression getNumber() {

return this.number;

}

public void setNumber(final Expression number) {

this.number = number;

}

@Override

public void execute(final DelegateExecution execution) throws Exception {

try {

String sAux = this.number.getValue(execution).toString();

System.out.println(sAux);

execution.setVariable("numberInput", sAux);

} catch (Throwable e) {

throw new BpmnError("BusinessExceptionOccurred");

}

}

}

Un cop creada la classe, s’accedeix a les propietats de l’element i s’assignen els atributs “Task Type” i “Class Name”

* **Email Task:** Per últim, es realitza l'enviament de l’email. Per a realitzar aquest pas, és necessari haver realitzat la configuració prèvia, tal com s’ha comentat en l'apartat anterior. Les opcions d'aquest component fan referència a propietats clàssiques d'un email.

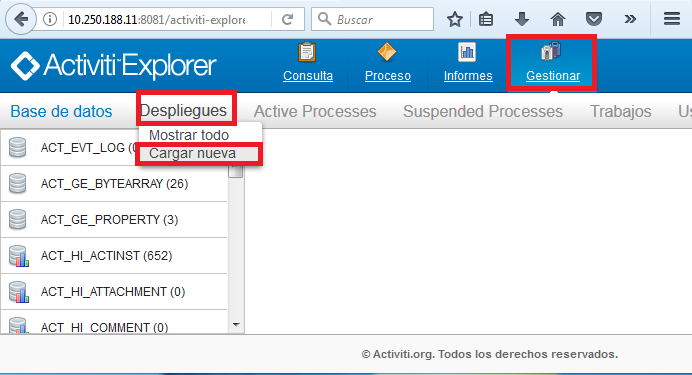
## Desplegament del diagrama

Abans de realitzar el desplegament a l’aplicació web, s’ha d’empaquetar el servei Java i copiar-lo a la carpeta de llibreries de l’aplicació que es troba a la ruta *{TOMCAT\_HOME/webapps/activiti-explorer/WEB-INF/lib/*.

Aquest procés es pot automatitzar afegint un control de versions com pot ser el Git i configurant que el repositori es trobi a la carpeta d’Activiti.

Per altra banda, s’ha de tenir en compte que en fer-se el desplegament d’un nou paquet, es reinicia també l’aplicació d’Activiti. En cas de produïr-se algun error a l’hora d’executar el paquet, Activiti deixa de funcionar.

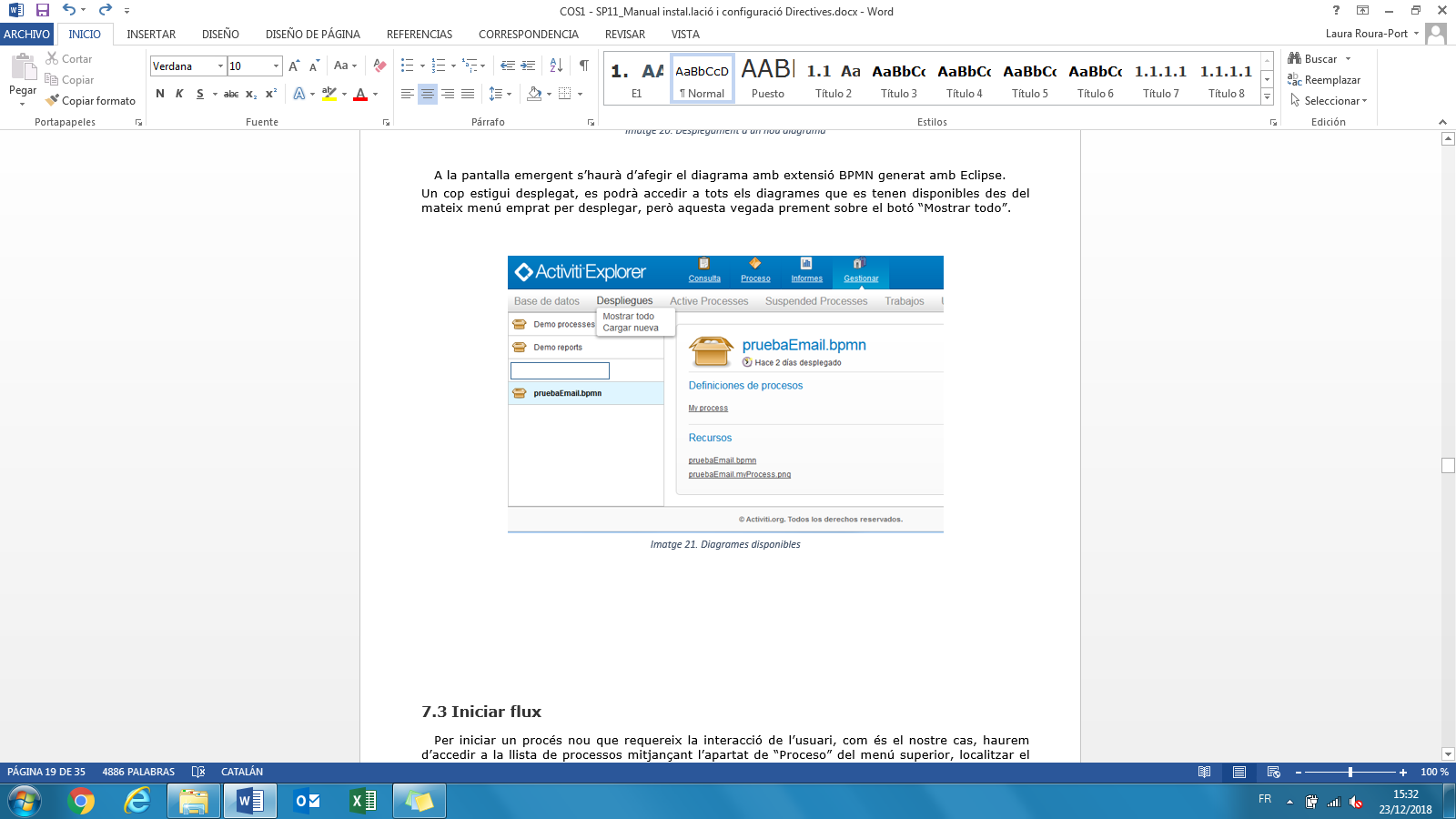
Per realitzar el desplegament del diagrama des de l’aplicació web, s’ha d’accedir a l’apartat “Gestionar” 🡪 “Despliegues” i fer clic sobre el botó “Cargar Nueva”.



Imatge 20. Desplegament d'un nou diagrama

A la pantalla emergent s’haurà d’afegir el diagrama amb extensió BPMN generat amb Eclipse.

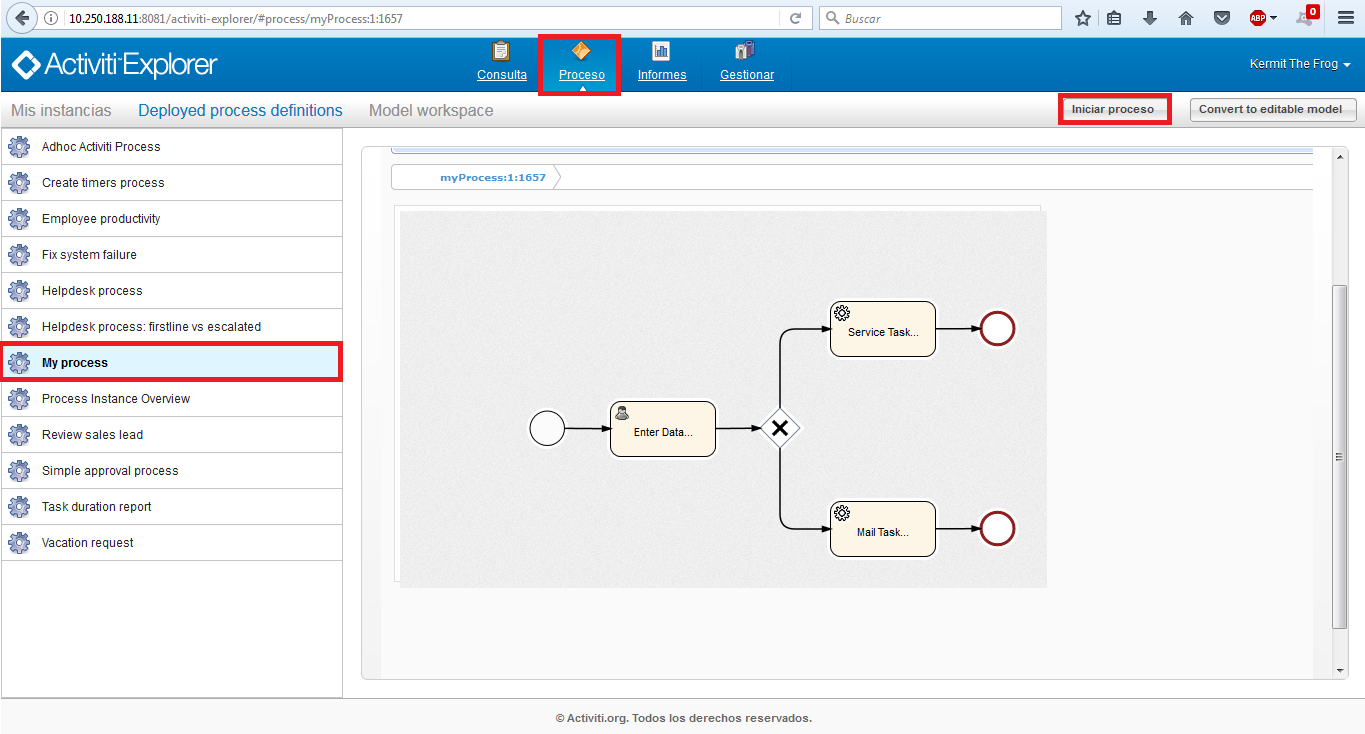
Un cop estigui desplegat, es podrà accedir a tots els diagrames que es tenen disponibles des del mateix menú emprat per desplegar, però aquesta vegada prement sobre el botó “Mostrar todo”.



Imatge 21. Diagrames disponibles

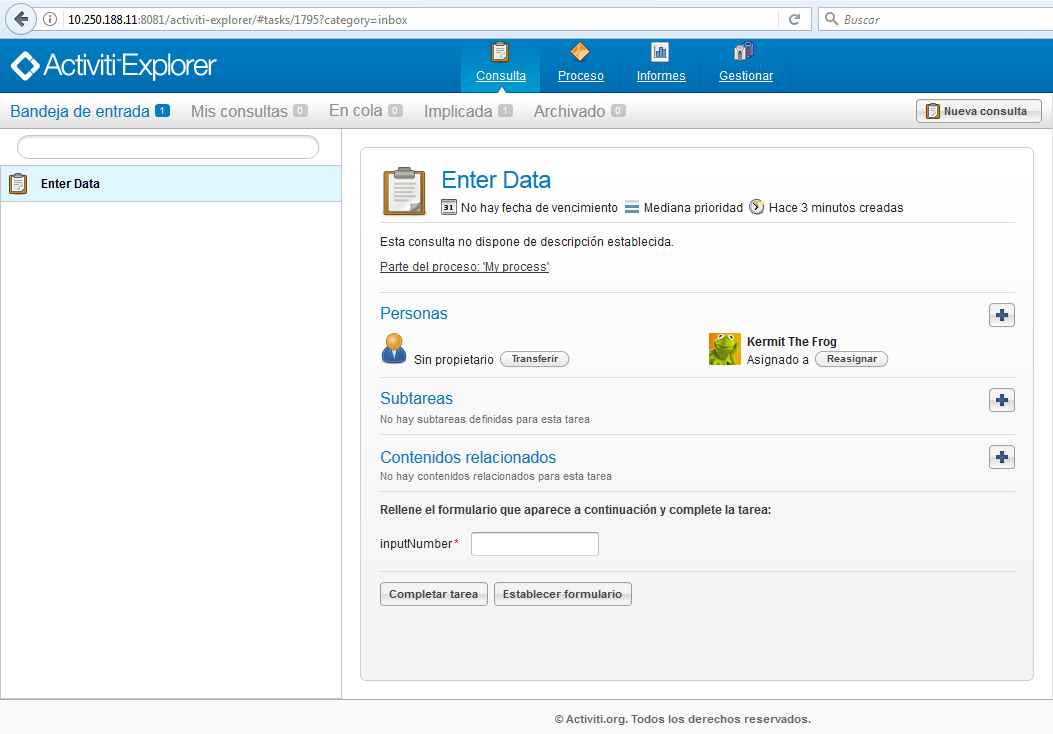
## Iniciar flux

Per iniciar un procés nou que requereix la interacció de l’usuari, com és el cas d’aquest exemple, s’haurà d’accedir a la llista de processos mitjançant l’apartat de “Proceso” del menú superior, localitzar el procés que es vol executar i fer clic sobre el botó “Iniciar proceso”.



Imatge 22. Inici del procés My process

Si s’accedeix amb l'usuari "Kermit" després d'haver-hi iniciat el procés, es tindrà un nou missatge a la bústia d'entrada. El títol que apareix al missatge és el mateix que se li ha assignat a la tasca.  
  
 En obrir el missatge, es podràn veure totes les propietats de la tasca, tals com: propietari de la tasca, usuaris candidats als quals se'ls hi ha assignat la tasca, si té alguna subtasca associada i el formulari amb les dades sol·licitades.



Imatge 23. Exemple de petició

Després d'introduir el valor, s'escollirà el camí a seguir depenent de si el numero és menor a 10 o no, tal com s’ha explicat prèviament. Després, es procedirà a l'enviament de l’email, en cas de ser inferior, o a modificar el valor de la variable, en cas de ser igual o superior.

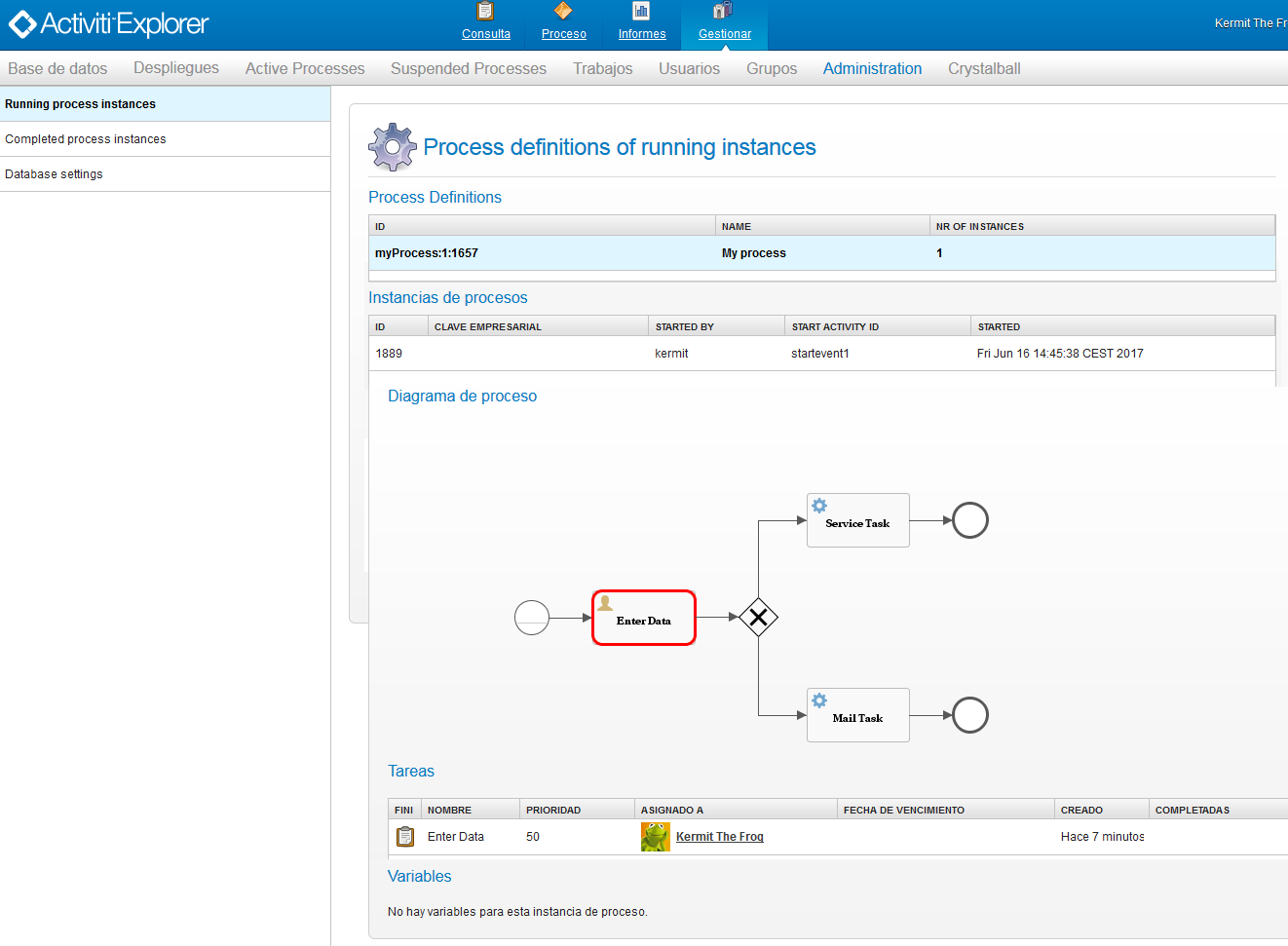
## Gestió de fluxes de treball

Si es disposen de permisos d’administrador, a més de poder gestionar els usuaris i els grups, es poden gestionar els processos que s’estan executant i veure l’històric dels que es van completant.

Aquest apartat és molt útil per saber en quin punt es troba un procés que no ha finalitzat encara o bé per saber què ha passat en un procés en concret.

Per accedir a aquesta opció, s’ha d’anar a l’apartat “Gestionar” i fer clic sobre la pestanya “Administration”.

A la següent imatge, es pot veure com existeix un procés actiu que es troba bloquejat en una tasca d'usuari, a l'espera de l'usuari "Kermit the frog" realitzi la tasca "Enter Data".



Imatge 24. Pantalla de processos actius

# REST API

## Instal·lació

Activiti BPM inclou una API REST pel motor d’Activiti. Per realitzar la seva instal·lació només cal realitzar la descompressió del paquet i realitzar la còpia dels fitxers WAR en el directori “webapps” d’Apache Tomcat.

De la mateixa manera que Activiti-explorer, el component d’Activiti-rest ve configurat per treballar amb una base de dades H2 en memòria. Per tal d’utilitzar els mateixos fluxos tècnics que a l’Activiti-explorer, s’ha d’accedir al fitxer *db.properties* situat al directori *{TOMCAT\_HOME}/webapps/activiti-rest/WEB-INF/classes/* i afegir la mateixa configuració emprada a l’Activiti-explorer.

## Mètodes

|  |  |
| --- | --- |
| **Mètode** | **Operació** |
| GET | Obté un recurs o una col·lecció de recursos. |
| POST | Crea un nou recurs.  Emprat també per realitzar consultes més complexes. |
| PUT | Actualitza propietats d’un recurs existent.  Emprat també per invocar accions sobre un recurs existent. |
| DELETE | Esborra un recurs. |

## Recursos

Activiti-rest ofereix les mateixes possibilitats que es poden realitzar des de l’Activiti-explorer. Això comporta una gran quantitat de crides diferents, entre les quals destacarem les dues crides necessàries per cercar un procés i executar-lo.

Totes les crides requereixen l’autenticació d’un usuari d’Activiti, motiu pel qual s’ha d’incloure una capçalera HTTP d’autenticació bàsica: **http://username:password@{host}**

Per realitzar la cerca d’un procés, es fa mitjançant la següent crida:

**GET** http://{username}:{password@{hostname}/activiti-rest/service/repository/process-definitions

|  |  |  |  |
| --- | --- | --- | --- |
| **Paràmetre** | **Obligatori** | **Valor** | **Descripció** |
| version | No | integer | Retorna un llistat de processos amb la versió donada |
| name | No | String | Retorna el procés amb el nom donat. |
| nameLike | No | String | Retorna un llistat de processos que contingui el text introduït en el camp nom |
| key | No | String | Retorna el procés amb la clau donada |
| keyLike | No | String | Retorna un llistat de processos que contingui el text introduït en el camp clau |
| resourceName | No | String | Retorna el procés amb el nom del recurs donat |
| resourceNameLike | No | String | Retorna un llistat de processos que contingui el text introduït en el camp nom del recurs |
| category | No | String | Retorna el procés amb el nom de la categoria donada |
| categoryLike | No | String | Retorna un llistat de processos que contingui el text introduït en el camp categoria |
| categoryNotEquals | No | String | Retorna un llistat de processos que no siguin de la categoria introduïda |
| deploymentId | No | String | Retorna el procés amb el identificador del desplegament donat |
| startableByUser | No | String | Retorna un llistat de processos que poden ser executats per l’usuari donat |
| sort | No | *name* (default), *id*, *key*, *category*, *deploymentId* and *version* | Propietat per endreçar la cerca |

Per executar un procés, és necessari conèixer l’identificador de la definició del procés, de la clau de la definició del procés o bé d’un missatge per poder-lo executar mitjançant la següent crida:

**POST** http://{username}:{password@{hostname}/activiti-rest/service/runtime/process-instances

El cos del missatge pot ser qualsevol dels següents, depenent de l’identificador que tinguem:

* Si disposem de l’identificador de la definició del procés:

{

"processDefinitionId":"prova:1:158",

"businessKey":"provaKey",

"variables": [

{"name":"var1","value":"Valor de la variable 1"},

{"name":"var2","value":"Valor de la variable 2"}

]

}

* Si disposem de la clau de la definició del procés:

{

"processDefinitionKey":"prova",

"businessKey":" provaKey",

"tenantId": "tenant1",

"variables": [

{"name":"var1","value":"Valor de la variable 1"},

{"name":"var2","value":"Valor de la variable 2"}

]

}

* Si disposem d’un missatge:

{

"message":"nouMissatge",

"businessKey":" provaKey",

"tenantId": "tenant1",

"variables": [

{"name":"var1","value":"Valor de la variable 1"},

{"name":"var2","value":"Valor de la variable 2"}

]

}

Els paràmetres businessKey, tenantID i variables són paràmetres opcionals.

# Exemple de fluxos de treball tècnics

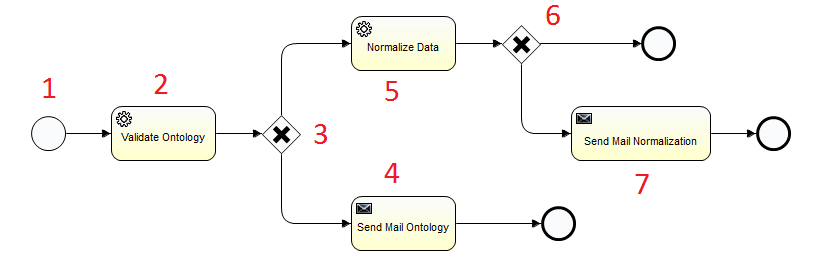
Seguint el que s’explica en el capítol 7, a continuació es descriu un possible flux de treball tècnic que es podria presentar en una plataforma.

El flux de treball comença en arribar una dada nova. Una vegada la dada entra dins de la plataforma a l’àrea de staging, s’intenta identificar la mateixa fent el mapeig amb l’ontologia.

Si la dada entrant és una dada desconeguda, s’ha decidit realitzar l’enviament d’un email informant que la dada no és coneguda i que cal revisar-la.

Si pel contrari, la dada és una dada coneguda, es pot iniciar el procés de normalització.

En aquest procés es poden produir diferents errors. Si això passés, s’enviaria un email informant que la dada no ha pogut ser normalitzada i que cal revisar-ne el format.



Imatge 25. Diagrama del flux de treball

## Explicació del diagrama

Per tal de simular el flux de treball esmentat, en iniciar-se el procés des d’Activiti (pas 1), apareixerà un formulari demanant l’entrada de les dades que han de passar pel procés. La informació introduïda es desarà a la variable *initialMessage*, la qual és de tipus String.

A continuació (pas 2), trobem l’event *Validate Ontology*, el qual s’encarrega de fer la crida a una funció Java a on es fa la validació de l’estructura de les dades. Els patrons que han de complir les dades són els següents:

* Han d’estar en format JSON.
* Han de contenir com a mínim els següents 2 elements:
  + Cod: Camp del tipus numèric. Intentar simular l’entrada d’un identificador.
  + Name: Camp que pot contenir una cadena de text.

La funció de Java desarà el resultat de la validació a la variable *ontValid*, valent 1 si el resultat de la validació compleix amb els patrons o amb valor igual a 0 en cas contrari.

public class ReadNumberTask implements JavaDelegate {

package com.[project\_name].bpmn;

import org.activiti.engine.delegate.BpmnError;

import org.activiti.engine.delegate.DelegateExecution;

import org.activiti.engine.delegate.Expression;

import org.activiti.engine.delegate.JavaDelegate;

import org.activiti.engine.impl.util.json.JSONObject;

public class ValidateOntology implements JavaDelegate {

private Expression message;

@Override

public void execute(final DelegateExecution execution) throws Exception {

try {

String phrase = this.message.getValue(execution).toString().toUpperCase();

JSONObject = new JSONObject(phrase);

try

{

int index = jsonObject.getInt("COD");

String name = jsonObject.getString("NAME");

System.out.println(index);

System.out.println(name);

execution.setVariable("ontValid", 1);

}

catch (Exception e) {

execution.setVariable("ontValid", 0);

}

} catch (Throwable e) {

execution.setVariable("ontValid ", 0);

}

}

}

Per tal de prendre una decisió segons el resultat que hagi donat la funció Java, s’ha utilitzat un gateway (pas 3). Per fer-ho, s’ha afegit la condició **${ontValid == 0}** al camí inferior i la condició

**${ontValid == 1}** al camí superior tal com s’explica a l’apartat anterior.

En cas que la variable tingui valor igual a 0, el procés anirà pel camí de l’enviament de email (pas 4), a on farà l’enviament de l’email informant que les dades no tenien el format esperat i després finalitzarà el procés.

Si el procés continua pel camí superior, es tornarà a fer una altra crida a una funció Java (pas 5). En aquest cas, es comprovarà que el codi de la dada introduïda al JSON sigui igual a 1. En cas afirmatiu, la funció Java desarà la variable *normValid* amb valor igual a 1; en cas contrari, es desarà la mateixa variable però amb valor igual a 0.

public class NormalizeData implements JavaDelegate {

private Expression message;

@Override

public void execute(final DelegateExecution execution) throws Exception {

try {

String phrase = this.message.getValue(execution).toString().toUpperCase();

JSONObject = new JSONObject(phrase);

try

{

int index = jsonObject.getInt("COD");

if (index == 1)

execution.setVariable("normValid", 1);

else

execution.setVariable("normValid", 0);

}

catch (Exception e) {

execution.setVariable("normValid", 0);

}

} catch (Throwable e) {

execution.setVariable("normValid", 0);

}

}

}

Per últim, es torna a fer una presa de decisió (pas 6) a partir de la variable *normValid*, per verificar si s’ha de procedir amb l’enviament de l’email (pas 7) informant que el procés de normalització ha fallat o bé es pot finalitzar el procés.

no llegibles.

# EXEMPLES DE FluxOs de treball funcionals

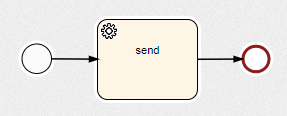
S’han desplegat diferents fluxos de treball dins de la plataforma com a exemples, per informar i enregistrar la informació obtinguda dels diferents processos que s’executen sobre la plataforma:

* Per tramitar l’enviament d’emails quan es produeix un error al procés de normalització.
* Per realitzar informes en format PDF.
* Per enregistrar el resultat de l’execució d’un procés.
* Per generar un informe diari resumint els resultats del dia.

Aquests fluxos de treball funcionals exemple es poden desplegar en els entorns de desenvolupament, preproducció i producció de la plataforma.

## Enviament d’emails

El primer dels fluxos de treball funcional exemple està pensat per ser cridat com a servei i està configurat per rebre la informació de l’últim procés de normalització i enviar un email al personal intern de l’IMI si ha succeït qualsevol error.



Imatge 31. sendEmail workflow

Crida al servei:

{

"processDefinitionId":"sendEmail:1:49198",

"variables": [

{"name": "jobName","value":

"[platform\_name]NormalizationJob - raw.[table\_origin].[table\_name]"}, vol error. {"name": "rawTableName", "value": "raw.[table\_origin].[table\_name]"},

{"name": "startTime", "value": "17/09/18 10:05:19"},

{"name": "endTime", "value": "17/09/18 10:05:38"},

{"name": "resultCode", "value": "000" },

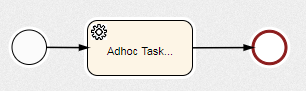
{"name": "description", "value": "[platform\_name]NormalizationJob - raw. .[table\_origin].[table\_name] process has finished successfully. Number of valid rows: 53"}

]

}

## Generació d’informes

El segon flux de treball funcional exemple es pot executar des de l’apartat gràfic d’Activiti o realitzar la crida com a servei. Rep com a paràmetre el contingut de l’informe la ruta i el nom del fitxer.



Imatge 32. Report workflow

{

"processDefinitionId":"reportWorkflow:1:153929",

"variables": [{

"name": "report",

"value": "[

{\"name\":\"jobName\",\"value\":\"[platform\_name]NormalizationJob - raw.[table\_origin].[table\_name]\"},

{\"name\":\"rawTableName\",\"value\": "raw.[table\_origin]. [table\_name]\"},

{\"name\":\"startTime\",\"value\":\"17/09/18 10:05:19\"},

{\"name\":\"endTime\",\"value\":\"17/09/18 10:05:38\"},

{\"name\":\"resultCode\",\"value\":\"000\"},

{\"name\":\"description\",\"value\":\"[platform\_name]NormalizationJob - raw. [table\_origin].[table\_name] process has finished successfully. Number of valid rows: 55\"}]"

},

{

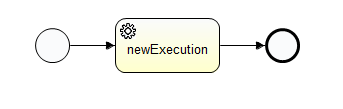
"name": "reportPath", "value": "/home/prueba/normalizacion"

}

]}

## Enregistrament d’execucions

El tercer flux de treball exemple es basa en enregistrar el resultat de les execucions dels diferents processos que formen part de la plataforma.



Imatge 33. newExecution workflow

Per realitzar la crida del flux, es pot executar des de l’apartat gràfic d’Activiti o realitzar la crida com a servei. Si s’executa des de la part gràfica, s’ofereix a l’usuari un formulari per omplir els camps necessaris. En cas de cridar-lo com a servei, és necessari enviar per paràmetre el resultat de l’execució de la següent manera:

{

"processDefinitionId":"newExecutionWorkflow:1:94178",

"variables": [

  { "name": "jobName", "value": "[platform\_name]NormalizationJob" },

  { "name": "rawTableName",  "value": "raw. [table\_origin]. [table\_name]" },

  { "name": "startTime", "value": "17/09/18 10:05:19" },

  { "name": "endTime", "value": "17/09/18 10:05:38" },

  { "name": "resultCode", "value": "003" },

{ "name": "description", "value": "Description" },

  { "name": "validRows", "value":"10" },

  { "name": "invalidRows", "value":"5" }

]}

A continuació, s’explica el significat de cadascuna de les variables:

jobname: Nom del procés executat.

rawTableName: Nom de la taula processada.

startTime: hora d’execució del procés.

endTime: hora de finalització del procés.

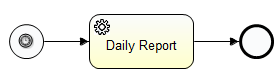
resultCode: Resultat del procés.

* 000: Execució finalitzada correctament.
* 001: La taula no existeix.
* 002: Falta definir la configuració
* 003: El procés ha finalitzat amb registres no vàlids.

El directori a on s’emmagatzemarà l’informe es pot modificar mitjançant un fitxer de configuració el procés que es troba al directori **/var/lib/tomcat/webapps/conf/workflows.conf.**

## Report diari

El quart flux de treball exemple s’encarrega de llegir el registre amb el resultat de les execucions dels diferents processos que formen part de la plataforma (ingesta, normalització, historificació, publicació) i genera un resum en format PDF de cadascun dels processos.



Imatge 34. dailyReport workflow

Aquest flux de treball es pot programar que s’excecuti de forma automàtica cada dia a l’hora que es vulgui (a les 01:00 per exemple) i s’encarrega de llegir el report generat durant el dia, el qual ha d’estar al directori especificat al fitxer de configuració **/var/lib/tomcat/webapps/conf/workflows.conf** i crea 4 informes en format PDF amb el resum de les execucions. Cadascun d’aquests 4 informes es pot desar a una ruta diferent; per fer-ho, es pot configurar mitjançant el mateix fitxer de configuració.