Govern de les Illes Balears
Vicepresidència i Conselleria
d'Innovació, Recerca i Turisme
Direcció General de Desenvolupament Tecnològic

Estàndards de la DGMAD relatius a la Plataforma Corporativa de Seguiment d'Expedients "Helium". Requisits i recomenacions.

versió 5 – gener 2022

Índex de continguts

1 . Introducció.

En línia amb el seu objectiu d'incorporar eines d'Administració Electrònica en el sí del Govern de les Illes Balears, la Direcció General de Modernització i Administració Digital, ve impulsant des de fa anys la implantació del producte "Helium" com a plataforma de referència de suport a la gestió d'expedients administratius de totes les tipologies.

Helium afegeix una capa de components com són variables, formularis, documents, utilitats d'enllaç (dominis, firma simple, remissió al portasignatures, notificacions telemàtiques) al producte de software lliure "jBPM" (Java Business Process Management), que aporta un disseny gràfic del workflow i programari associat ("handlers" i "scripts"). Tot junt ha constituït un instrument per, d'una forma relativament senzilla, dissenyar i implantar tramitacions amb un abast funcional en expansió.

És precisament l'expansió de l'Helium el que ha facilitat l'actual detecció en la DGMAD per part del personal encarregat de l'administració del producte, d'una sèrie de qüestions relacionades d'una banda amb el disseny de les tramitacions -el desplegament i pas a producció de les quals s'ens sol·licita- i d'altra amb operatives dels gestors de cada entorn que -per acció o omissió- penalitzen el rendiment del producte i fins i tot provoquen eventuals caigudes de tota la plataforma.

La pretensió d'aquesta publicació no és en absolut forçar la modificació en les implantacions ja fetes per tal d'adaptar-les a les recomenacions, sinó evitar en el futur el que ens sembla siguin operatives de disseny que dificultin el seu anàlisi aquí en la DGMAD i/o que puguin incidir en el rendiment de la plataforma o operatives d'administració amb el mateix efecte negatiu.

Per això, **aquest document es veurà contínuament revisat i actualitzat**.

2 . Definicions de procés en la capa jBPM.

a) Homogeneïtzació de nomenclatures.

Per les definicions de procés que es vulguin desplegar en tipologies d'expedient noves **serà necessari que tots els codis vagin en MINÚSCULA (per distinguir-los dels corresponents codis de tipus d'expedient que aniran en MAJÚSCULA com s'indica en el punt 3.b)**, amb **grups de caràcters separats per GUIONS BAIXOS**, i seguint el següent criteri de format: **prefixe de 3 caràcters indicatiu de la Direcció General del Govern o òrgan gestor, o de lliure elecció si es tracta d'una implantació de caire corporatiu o transversal, seguit d'un màxim de 10 caràcters indicatius de la funcionalitat del procés o subprocés**.

Per exemple:

sfs_invalidesa (Entorn Salut, Família i Serveis Socials, expedients de seguiment d'ajudes

a la invalidesa)

edu_accgenp1 (Entorn Educació, Personal Docent, Universitats i Recerca, expedients de seguiment d'accions generals, procés principal)

edu_accgenp2 (Entorn Educació, Personal Docent, Universitats i Recerca, expedients de seguiment d'accions generals, subprocés de notificacions).

b) Ús de l'eina gràfica.

La no utilització de l'eina gràfica associada als fluxes de tramitació de les deficions de procés jBPM té com a conseqüència que els diagrames que genera la pròpia eina resulten il·legibles. Això és conseqüència de l'operativa de treballar directament sobre els arxius "source" XML, que eventualment podrà facilitar la introducció directa de codi relatiu a components com events, accions ("handlers" i "scripts"), assignements, pas de paràmetres (mapeig de variables) a subprocessos, etc.., però incrementa també el perill de deixar nodes inconnexes i fa impossible un eventual repàs per part de la DGMAD. Atès això, **no s'admetrà el desplegament de definicions de procés que no presentin una raonable claredat en el seu diagrama**.

c) Ús dels subprocessos.

JBPM implementa l'execució de tramitacions en subprocesos, utilitat molt convenient en fluxes complexes per simplificar els diagrames i en execucions de blocs de tramitació recursiva. És fins i tot útil a l'usuari final per informar-lo de la fase procedimental en què es troba l'expedient que tramita. Es disposa també del handler predefinit "ProcesTitolModificarHandler" per tal de fer que l'usuari final visualitzi una descripció més completa del procés en què està tramitant. Recomenem la seva utilització. Si no es justifica la impossibilitat d'utilitzar subprocessos, en la DGMAD **ens reservem sol·licitar la revisió dels quaderns de càrrega de desplegament de definicions de procés de dimensions excessives.**

Per la cridada a subprocessos, jBPM defineix els nodes de procés ("process state") que traslladen el fluxe al subprocés amb mapeig de variables. No s'entén, en principi, la necessitat de forçar l'entrada a un procés mitjançant l'execució d'un handler o programari java específic des d'un altre procés. Si no es justifica la impossibilitat d'utilitzar la forma nadiva de fer-ho, en la DGMAD **ens reservem sol·licitar la revisió dels quaderns de càrrega de desplegament amb definicions de procés que s'invoquin fora de fluxe, mitjançant programari extern.**

d) Codificació de tasques.

Malgrat pugui semblar còmode escriure el codi de tasca com si fos ja pràcticament una descripció, cosa que estalvia haver de re-escriure descripcions després de desplegar la definició de procés dins Helium en les components de la pipella de disseny "Tasques", aquesta praxis comporta que hi hagi caràcters com espais en blanc, accents, etc.. (propis d'una descripció) en cadenes preparades per rebre un codi. Això pot dur problemes, per la

qual cosa **ens reservem sol·licitar la revisió dels quaderns de càrrega de desplegament amb definicions de procés que incloguin "Task Nodes" amb codis de tasca no definits en cadenes inferiors a 30 caràcters, amb grups separats per GUIONS BAIXOS, i caràcters en MAJÚSCULA.** Es recomana, així mateix, que hi hagi un prefixe de 3 caràcters que permeti la ordenació alfanumèrica de la llista de tasques de la definició de procés desplegada dins Helium.

e) Execució del procés d'esborrat d'històric de tasques en la finalització d'expedients.

S'ha de dissenyar l'execució, en algun dels nodes o transicions finals del fluxe tramitador, del handler predefinit "ExpedientBuidaLogHandler". Aquest handler esborra tota l'estructura de l'històric de variables de l'expedient, necessària principalment per executar retroaccions -que queden sense sentit quan l'expedient ha finalitzat- i que carrega sensiblement el volum d'informació a tractar en els processos de consulta. En la DGMAD **ens reservem sol·licitar la revisió dels quaderns de càrrega de desplegament amb definicions de procés en què no ens quedi clara la incorporació d'aquest procés.**

f) Ús del fork/join.

Es recomana repassar els dissenys per tal d'evitar definicions de procés que continguin un node tipus “fork” amb una sola branca posterior.

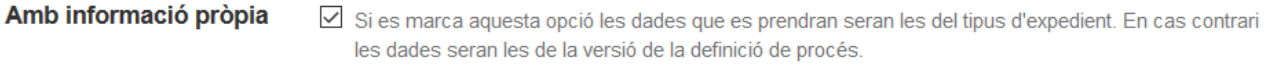
g) Integracions amb sistemes externs.

Per desenvolupar integracions amb fonts de dades externes o amb formularis externs es recomana que s'utilizin “serveis web”.

3 . Incorporació de components d'aplicació (capa "Helium").

a) Necessitat de la configuració de la tipologia amb variables pròpies.

Per tal d'independitzar les components de disseny de cada tipologia de les associades al motor de workflow (jBPM), i facilitar així una futura migració cap a altres motors més moderns o actualitzats, és necessari marcar que en el disseny les components vàlides seran les definides a nivell de tipus d'expedient:



b) Homogeneïtzació de nomenclatures.

Fent ja referència als components d'aplicació dins "Helium", un cop desplegades les definicions de procés des del jBPM, considerem necessari fixar un estàndard que homogeneïtzi una mica les nomenclatures de tipologies i grups, atenent al fets:

- Que el **codi de tipus d'expedient (tipologia)** és el que donarà lloc al **codi de l'aplicació en l'inventari de la DGMAD**, i que tenim distribuïts els entorns -o grups de tipologies- de la plataforma més o menys **d'acord a les direccions generals del Govern.**

- Que el nom dels grups d'usuaris a què s'assignen els **"roles" interns d'Helium** han de coincidir amb els **"roles" definits en el SEYCON** i per tant han de seguir la nomenclatura fixada pels mateixos en els estàndards de la DGMAD.

A partir d'ara per les tipologies noves que es vulguin desplegar serà necessari que tots els **codis vagin en MAJÚSCULA**, amb **grups de caràcters separats per GUIONS BAIXOS**, i seguint els següents criteris de format:

- Pel codi de tipus d'expedient (=codi d'aplicació en l'inventari de la DGMAD): **prefixe de 3 caràcters indicatiu de la Direcció General del Govern o òrgan gestor, o de lliure elecció si es tracta d'una implantació de caire corporatiu o transversal, seguit d'un màxim de 10 caràcters indicatius de l'objecte d'aquests expedients.** Per exemple:

SFS_INVALIDESA (Entorn Salut, Família i Serveis Socials, expedients de seguiment d'ajudes a la invalidesa)
EDU_ACCGEN (Entorn Educació, Personal Docent, Universitats i Recerca, expedients de seguiment d'accions generals)

- Pel codi dels grups d'usuaris (= "roles" que s'hauran de definir en el sistema SEYCON): El formaran el codi del tipus d'expedient definit amb el criteri anterior (abreviant en lo possible els caràcters que indiquen l'objecte de l'expedient) seguit d'un màxim de **15 caràcters indicatius de la significació genèrica d'aquest "role" o funcions** del grup d'usuaris adscrits. Per exemple:

SFS_INV_INSP ("Role" o grup d'inspectors dels expedients de seguiment d'ajudes a la invalidesa del entorn Salut, Família i Serveis Socials, amb permisos bàsics de tramitació, i permisos per cancel·lar, reassignar, suspendre i reprendre tasques. Podria així tenir assignats els roles interns d'Helium CREAR, LLEGIR, TASCA GESTIONAR)

EDU_ACCGEN_AVAL ("Role" o grup d'avaluadors expedients de seguiment d'accions generals de l'entorn Educació, Personal Docent, Universitats i Recerca, amb permisos bàsics de tramitació, permisos per cancel·lar, reassignar, suspendre i reprendre tasques, permisos per veure les tasques de qualsevol usuari de l'expedient i permisos per retrocedir una tasca. Podria així tenir assignats els roles interns d'Helium CREAR, LLEGIR, TASCA GESTIONAR, TASCA SUPERVISAR, REGISTRE GESTIONAR)

EDU_ACCGEN_ADMIN ("Role" o grup d'administradors expedients de seguiment

d'accions generals Educació, Personal Docent, Universitats i Recerca, amb permisos bàsics de tramitació, permisos per cancel·lar, reassignar, suspendre i reprendre tasques, permisos per veure les tasques de qualsevol usuari de l'expedient, permisos per retrocedir una tasca, permisos per gestionar documents i permisos per modificar ordre i etiqueta dels valors de les enumeracions. Podria així tenir assignats els roles interns d'Helium CREAR, LLEGIR, TASCA GESTIONAR, TASCA SUPERVISAR, REGISTRE GESTIONAR, DOCUMENT GESTIONAR, DISSENY DELEGAT)

NO és recomenable que els codis de tasca, procedents del disseny jBPM, siguin els mateixos que les descripcions. Cal renombrar les descripcions de les tasques un cop desplegada una definició de procés, éssent aquí on el dissenyador pot qualificar completament la descripció, i fins i tot fer-la dependre del valor/ d'una/vàries variable/s.

La recomenació ja esmentada relativa als codis de tasca, consistent en què vagin en MAJÚSCULA i amb grups de caràcters separats per GUIONS BAIXOS, es fa extensiva a la codificació de la resta de components de disseny de components dins Helium a nivell de definició de procés (documents, terminis, agrupacions, accions...) i de tipus d'expedient (estats, enumeracions, dominis, consultes...).

c) Roles d'administració en producció.

Els "roles" interns d'Helium "DISSENY ADMIN" i "ADMINISTRAR", que donen a l'usuari que els té atorgats la possibilitat d'esborrar components (variables, estats, definicions de procés), executar processos d'importació i instruccions o scripts "BEANSHELL" que afectin fins i tot a BD, **NO és recomanable que es sol·licitin en PRODUCCIÓ**. L'operativa adequada és treballar en PRE i exportar a nivell de definició de procés jBPM (arxius .PAR), a nivell de components Helium de definició de procés (en arxius .EXP) o a nivell de tipus d'expedient (workflow jBPM i components Helium de totes les definicions de procés en arxius .EXP), fent després una sol·licitud de desplegament o d'importació als administradors en la DGMAD, adjuntant els respectius arxius .par o .exp adients.

En la DGMAD **ens reservem sol·licitar la revisió dels quaderns de càrrega de desplegament amb assignació d'aquests roles d'administració en producció.**

d) Control de tipus de dades en el seu traspàs.

En cas de realitzar traspàs de dades de Java a Helium (integracions, handlers, etc), és necessari utilitzar en Java el tipus corresponent definit en Helium sense forçar el "càsting". Per exemple, el tipus "INTEGER" d'Helium es correspon amb un tipus java.lang.Long.

e) Aprofitament de les variables tipus "Registre".

En cas de haver d'utilitzar estructures de dades de "n" valors es recomana utilitzar el tipus "REGISTRE" d'Helium en lloc d'emmagatzemar "n" variables senzilles.

4 . Operatives d'administració dels gestors de cada entorn.

a) Traspàs d'estructures i components.

Pel que fa al traspàs d'estructures i components entre els diferents entorns de feina que estiguin validats, hi ha que tenir en compte el que s'indica en el manual de disseny de la Plataforma. Al quadern de càrrega cal indicar l'entorn on s'ha de fer la instal·lació, el codi concret de la definició de procés o tipus d'expedient amb què s'ha de treballar i el tipus d'operativa amb l'arxiu subministrat:

- desplegament per definició de procés a nivell jBPM (fitxer .PAR)
- importació per definició de procés en la capa de components Helium (fitxer .EXP)
- importació per tipus d'expedient en la capa de components Helium (fitxer .EXP)

En el primer cas, desplegament per definició de procés jBPM, cal indicar si cal actualitzar els expedients actius. En el segon i tercer cas, cal indicar si s'han de sobreescriure dades existents amb el mateix codi. I en el tercer cas, cal indicar també si s'han d'incloure dades bàsiques (títol, opcions, seqüències). **Ens reservem sol·licitar la revisió dels quaderns de càrrega que no tinguin clares aquestes indicacions.**

b) Subministrament de les fonts Java.

En sol·licitar el desplegament d'un arxiu .PAR, que conté tots els components jBPM (workflow), amb la possibilitat d'incorporar handlers generats des de codi Java independent, és necessari incloure tant les clases compilades (.class) com les fonts Java. Atès que el codi font Java dels handlers no queda incorporat al generar l'arxiu .PAR des del jBPM, s'haurà de lliurar en un arxiu comprimit denominat “sources” per a la seva revisió. **Ens reservem sol·licitar la revisió dels quaderns de càrrega de desplegament amb definicions de procés que no incloguin aquest arxius font.**

c) Execució d'accions massives.

Es recomana executar el mínim d'accions massives, malgrat sigui en segon pla. Es penalitza greument el sistema, i han de ser utilitzades per **a tasques d'administració i no de tramitació**. Cas d'haver d'utilitzar-se han de llançar-se fora de l'horari de treball dels usuaris interactius (a partir de les 15:00 hores).

d) Finalització d'expedients (usuaris finals).

Els expedients que pasen definitivament a una fase d'arxiu definitiu s'han de finalitzar, es a dir, han d'assolir el "End State" del jBPM en tots els processos, el principal i els eventuals subprocessos. No poden quedar "Task Nodes" oberts, malgrat aparentment s'hagi finalitzat l'expedient (us recordem que un procés pot tenir més d'un “End State”). Això és una **operativa a què s'haurien d'habituar als usuaris finals, per recomenació dels administradors de cada tipologia**.

e) Esborrat d'informació de registre (esborrat de "logs").

La utilitat de retroacció, que és configurable a nivell de tipus d'expedient, permet que els usuaris amb el permís adient ("REGISTRE GESTIONAR"), puguin desfer tràmits amb retorn a l'estat anterior de tots els components (variables i documents) configurades per fer-ho. Aquesta utilitat, desenvolupada sobre la capa Helium per ampliar i millorar l'eina nadiva dels "tokens", lligada al jBPM, té l'inconvenient que, quan està habilitada, carrega una taula de "logs" amb el valors de les variables en cada tràmit que és ja d'un tamany quasi insostenible, la qual cosa afecta negativament sobre el rendiment del producte.

Per això, és necessari

- que els dissenyadors habilitin els "logs" (indicador "amb retroacció"a nivell de tipus d'expedient) només quan sigui imprescindible desfer tràmits amb retorn als valors previs dels components. Si això no és un requisit per cap component es recomenable desactivar o comprovar que està desactivat el corresponent indicador a nivell de tipus d'expedient (disseny -> tipus d'expedient -> informació), i procedir amb la utilitat dels "tokens" per eventuals bots de fluxe a tràmits anteriors.

- si no es pot desactivar la retroacció a nivell de tota la tipologia, és preceptiu associar el handler predefinit "ExpedientBuidaLogHandler" a les transicions finals per tal de minimitzar l'impacte de la taula de "logs" en el rendiment del producte.

f) Esborrat de definicions de procés obsoletes i no utilitzades.

Durant el període d'implantació d'un tipus d'expedient, és habitual desplegar, fins i tot en l'entorn de producció, moltes versions de les definicions de procés en poc temps, a les quals s'actualitzen els expedients. Això genera molta estructura "inútil" en BD, que ha d'esborrar-se, amb la corresponent utilitat d'administració habilitada a nivell de disseny de tipus d'expedient.

g) Indexació asíncrona.

Quan es finalitza cada tràmit o tasca, s'executa una reindexació del "Lucene" per actualitzar les recerques. Aquest procés, si s'executa intercativament, pot penalitzar eventualment el temps d'espera per a l'usuari. Per això, convé activar l'indicador de "Reindexació asíncrona" a nivell de tipus d'expedient, que passa l'execució d'aquest procés a segon pla, alliberant més ràpidament la tasca de cara a l'usuari.

h) Finalització de tasques en segon pla.

Quan es finalitza cada tràmit o tasca, es poden executar -segons el disseny de cada tipologia- un o varis processos o accions (handlers, scripts, etc..). Pot interessar, sobre tot en aquells òrgans que tramitin pocs moviments però en molts d'expedients alhora, que aquests processos s'executin en segon pla. Per això, convé activar l'indicador de "Fiinalització en segon pla" a nivell de tasca, alliberant més ràpidament la tasca cara a l'usuari.

5 . Restricció d'accions dels handlers mitjançant una nova API d'accés a la funcionalitat de HELIUM/jBPM.

a) Objecte.

Controlar l'accés dels handlers a la funcionalitat que ofereix HELIUM i la llibreria jBPM amb la intenció d'assegurar unes bones pràctiques de programació.
La versió actual de l'aplicació permet una total llibertat a l'hora d'implementar els handlers jBPM en l'accés a l'estructura de dades interna d'un expedient. Això provoca que de vegades, els handlers implementin determinada funcionalitat de forma no gaire eficient, la qual cosa pot afectar negativament al rendiment de tot el producte, o que modifiquin el funcionament normal del fluxograma definit, poguent provocar efectes no disitjats en la tramitació.

b) Afectació als nous handlers.

S'ha definit una nova interfície que hauran d'implementar de forma obligatòria tots els handlers de l'aplicació redissenyant l'API d'accés a la funcionalitat de jBPM/Helium. Tots els accessos a la funcionalitat de jBPM que no estiguin contemplats a dins aquesta nova API estaran restringits per a evitar males pràctiques en la programació dels handlers i possibles problemes de rendiment.

L'accés a aquesta nova API es farà mitjançant un canvi en la implementació dels handlers. Fins ara els handlers havien d'implementar la següent interfície:

```
package org.jbpm.graph.def;
public interface ActionHandler extends Serializable {
      void execute(ExecutionContext executionContext) throws Exception;
}
```

Per a utilitzar la nova API els handlers hauran d'estendre de la classe abstracta "net.conselldemallorca.helium.jbpm3.api.HeliumActionHandler":

```
package net.conselldemallorca.helium.jbpm3.api;
public abstract class HeliumActionHandler implements ActionHandler {
      public abstract void execute(HeliumApi heliumApi)
                  throws HeliumHandlerException;
      public abstract void retrocedir(
            HeliumApi heliumApi,
            List<String> parametres) throws HeliumHandlerException;
}
```

Al estendre aquesta classe estem obligats a implementar dues funcions:

- public void execute(HeliumApi api) throws HeliumHandlerException: codi que s'executarà al executar el handler, tant sigui en la tramitació d'un expedient, com si es crida des de una acció definida a Helium.

- public void retrocedir(HeliumApi api, List<String> args) throws Exception: codi que s'executarà durant el retrocés de l'expedient, allà on s'havia executat el handler.

Implementant els handlers amb aquesta classe abstracta, es limita l'accés a la

funcionalitat jBPM al que permet la classe HeliumApi. Aquesta classe força a que els accessos a la funcionalitat jBPM es facin mitjançant la classe "HeliumApi" en lloc de l'habitual "ExecutionContext" de jBPM.

Els mètodes disponibles a la nova classe "HeliumApi" seran equivalents a l'antic "jBPMContext" per a la consulta d'informació. La modificació d'informació, en canvi, estarà restringida. Tota la funcionalitat de modificació d'informació de jBPM o d' Helium s'accedirà mitjançant els mètodes de la nova "HeliumApi".
A continuació es mostra un exemple d'implementació d'un nou handler de Helium que empra la instància de la nova API per a modificar la variable «var_1» del procés:

```
public class SampleHeliumHandler extends HeliumActionHandler {
	public void execute(HeliumApi heliumApi) throws HeliumHandlerException {
		// Modificació de variable de procés
		api.setVariableInstanciaProces(
			"var_1",
			"valor_1");
		}
	}
	public void retrocedir(
		HeliumApi heliumApi,
		List<String> parametres) throws HeliumHandlerException {
		// No executam cap acció al retrocedir
	}
}
```

En un primer moment, podran coexistir les dues formes d'implementació dels handlers per assegurar la compatibilitat amb els tipus d'expedient ja existents. Aquesta coexistència **es mantindrà durant un temps per donar temps a migrar tota la funcionalitat existent cap als nous handlers**.

c) Definició de la nova API.Mètodes de consulta d'informació del jBPM (Execution context).

A continuació es llisten els mètodes que ofereix la interfície HeliumApi per a accedir a la informació del jBPM.
En els següents apartats, s'informa dels objectes retornats, ara bé, si es desitja més informació del significat de cada camp, s'haurà de consultar la documentació del jBPM:

http://docs.jboss.org/jbpm/v3.2/

- Informació de Token actual

<table>
<tr><td>Mètode</td><td colspan="2">public TokenInfo getToken();</td></tr>
<tr><td>Descripció</td><td colspan="2">Obté la informació del token on es troba el node actual de jBPM.</td></tr>
<tr><td rowspan="4">Valor retornat</td><td colspan="2">Objecte TokenInfo, format pels següents camps:</td></tr>
<tr><td>Camp</td><td>Tipus</td></tr>
<tr><td>id</td><td>long</td></tr>
<tr><td>name</td><td>String</td></tr>
<tr><td>start</td><td>Date</td></tr>
</table>

	end	Date
	node	NodeInfo
	nodeEnter	Date
	processInstance	ProcessInstanceInfo
	parent	TokenInfo
	children	Map<String, TokenInfo>
	subProcessInstance	ProcessInstanceInfo
	isAbleToReactivateParent	boolean
	isTerminationImplicit	boolean
	isSuspended	boolean
	lock	String

- Informació del Node actual

Mètode	public TokenInfo getNode();	
Descripció	Obté la informació del node actual de jBPM.	
Valor retornat	Objecte NodeInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	name	String
	description	String
	processDefinition	ProcessDefinitionInfo
	leavingTransitions	List<TransitionInfo>
	arrivingTransitions	List<TransitionInfo>
	action	ActionInfo
	isAsync	boolean
	isAsyncExclusive	boolean

- Informació de la definició de procés actual

Mètode	public ProcessDefinitionInfo getProcessDefinition();	
Descripció	Obté la informació de la definició de procés on es troba el node actual de jBPM.	
Valor retornat	Objecte ProcessDefinitionInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	name	String
	description	String
	version	int
	isTerminationImplicit	boolean

	startState	NodeInfo
	nodes	List<NodeInfo>
	actions	Map<String, ActionInfo>

- Informació de la instància de procés actual

Mètode	public ProcessInstanceInfo getProcessInstance();	
Descripció	Obté la informació de la instància de procés on es troba el node actual de jBPM.	
Valor retornat	Objecte ProcessInstanceInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	versio	int
	key	String
	start	Date
	end	Date
	processDefinition	ProcessDefinitionInfo
	rootToken	TokenInfo
	superProcessToken	TokenInfo
	isSuspended	boolean

- Informació de la acció actual

Mètode	public ActionInfo getAction();	
Descripció	Obté la informació de la acció que s'està executant.	
Valor retornat	Objecte ActionInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	name	String
	isPropagationAllowed	Date
	isAsync	boolean
	isAsyncExclusive	boolean
	actionExpression	String
	processDefinition	ProcessDefinitionInfo

- Informació de l'event actual

Mètode	public EventInfo getEvent();
Descripció	Obté la informació de l'event on es troba la acció que s'està executant actualment.
Valor retornat	Objecte EventInfo, format pels següents camps:

	Camp	Tipus
	id	long
	eventType	String
	actions	List<ActionInfo>

- Informació de la transició actual

Mètode	public TransitionInfo getTransition();	
Descripció	Obté la informació de la transició on es troba l'event actual de jBPM.	
Valor retornat	Objecte TransitionInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	name	String
	description	String
	from	NodeInfo
	to	NodeInfo
	condition	String

- Informació de la tasca actual

Mètode	public TaskInfo getTask();	
Descripció	Obté la informació de la definició de la tasca actual.	
Valor retornat	Objecte TaskInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	name	String
	description	String
	processDefinition	ProcessDefinitionInfo
	isBlocking	boolean
	isSignalling	boolean
	condition	String
	dueDate	String
	priority	int
	taskNode	TaskNodeInfo
	startState	NodeInfo
	swimlane	SwimlaneInfo
	actorIdExpression	String
	pooledActorsExpression	String

- Informació de la instància de procés del subprocés

Mètode	public ProcessInstanceInfo getSubProcessInstance();
Descripció	Obté la informació del token on es troba el node actual de jBPM.
Valor retornat	Objecte ProcessInstanceInfo

- Informació de la instància de tasca actual

Mètode	public TaskInstanceInfo getTaskInstance();	
Descripció	Obté la informació de la instància de la tasca actual.	
Valor retornat	Objecte TaskInstanceInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	name	String
	description	String
	actorId	String
	create	Date
	start	Date
	end	Date

	dueDate	Date
	priority	int
	isCancelled	boolean
	isSuspended	boolean
	isOpen	boolean
	isSignalling	boolean
	isBlocking	boolean
	task	TaskInfo
	token	TokenInfo
	swimlaneInstance	SwimlaneInstanceInfo
	taskMgmtInstance	taskMgmtInstanceInfo
	processInstance	ProcessInstanceInfo
	pooledActors	Set<PooledActorInfo>
	variableInstances	Map<String, VariableInstanceInfo>

- Informació del timer actual

Mètode	public TimerInfo getTimer();	
Descripció	Obté la informació del timer on s'està executant la acció actual.	
Valor retornat	Objecte TimerInfo, format pels següents camps:	
	Camp	Tipus
	id	long
	version	int
	dueDate	Date
	processInstance	ProcessInstanceInfo
	token	TokenInfo
	taskInstance	TaskInstanceInfo
	isSuspended	boolean
	isExclusive	boolean
	lockOwner	String
	lockTime	Date
	exception	String
	retries	int
	configuration	String
	name	String
	repeat	String
	transitionName	String
	action	ActionInfo

Els objectes d'informació retornats poden contenir camps que fan referència a altres objectes de

jBPM, i que s'han convertit a classes pròpies informatives. Les classes que no s'han especificat com a objectes retornats per els mètodes anteriors, i que s'utilitzen en alguns d'ells son:

- TaskNodeInfo

Camp	Tipus
id	long
name	String
description	String
processDefinition	ProcessDefinitionInfo
leavingTransitions	List<TransitionInfo>
arrivingTransitions	List<TransitionInfo>
action	ActionInfo
isAsync	boolean
isAsyncExclusive	boolean
tasks	Set<TaskInfo>
signal	int
createTasks	boolean
endTasks	boolean

- TaskMgmtInstanceInfo

Camp	Tipus
id	long
processInstance	ProcessInstanceInfo
swimlaneInstances	Map<String, SwimlaneInstanceInfo>
taskInstances	Set<TaskInstanceInfo>

- VariableInstanceInfo

Camp	Tipus
name	String
processInstance	ProcessInstanceInfo
token	TokenInfo
Value	Object

- SwimlaneInfo

Camp	Tipus
id	long
name	String
actorIdExpression	String
pooledActorsExpression	String

tasks	Set<TaskInfo>

- SwimlaneInstanceInfo

Camp	Tipus
id	long
name	String
actorId	String
pooledActors	Set<PooledActorInfo>
swimlane	SwimlaneInfo
taskMgmtInstance	TaskMgmtInstanceInfo

- PooledActorInfo

  private String actorId;
  private Set<TaskInstance> taskInstances;
  private SwimlaneInstance swimlaneInstance;

Camp	Tipus
id	long
actorId	String
taskInstances	Set<TaskInstanceInfo>
swimlaneInstance	SwimlaneInstanceInfo

d) Definició de la nova API. Funcions per a interectuar amb l'expedient.

- Obtenir valor d'una variable

Mètode	public Object getVariable(String varCodi);
Descripció	Obté el valor d'una variable de jBPM. El context a on es consultarà la variable dependrà del lloc a on s'executi el handler (Si s'executa a dins una instància de tasca s'obtendrà el valor de la variable de la instància de tasca).
Paràmetres	• varCodi: Codi de la variable a llegir.
Valor retornat	El valor de la variable.

- Obtenir el text d'una variable

Mètode	public Object getVariableText(String varCodi);
Descripció	Obté el text que es mostra a Helium d'una variable de jBPM. Depenent del tipus de variable, el valor es formata o es consulta abans de mostrar-lo a l'usuari. El context a on es consultarà la variable dependrà del lloc a on s'executi el handler (Si s'executa a dins una instància de tasca s'obtendrà el valor de la

	variable de la instància de tasca).
Paràmetres	• varCodi: Codi de la variable a llegir.
Valor retornat	El valor de la variable formatat per mostrar a l'usuari.

- Obtenir el valor d'una variable d'una tasca

Mètode	public Object getVariableInstanciaTasca(String varCodi);
Descripció	Obté el valor d'una variable de jBPM de la instància de tasca. Si l'execució d'aquest mètode es realitza a dins un handler no associat a una instància de tasca es produirà una excepció.
Paràmetres	• varCodi: Codi de la variable a llegir.
Valor retornat	El valor de la variable.

- Obtenir el text d'una variable d'una tasca

Mètode	public Object getVariableInstanciaTascaText(String varCodi);
Descripció	Obté el text que es mostra a Helium d'una variable de jBPM de la instància de tasca. Depenent del tipus de variable, el valor es formateja o es consulta abans de mostrar-lo a l'usuari. Si l'execució d'aquest mètode es realitza a dins un handler no associat a una instància de tasca es produirà una excepció.
Paràmetres	• varCodi: Codi de la variable a llegir.
Valor retornat	El valor de la variable formatat per mostrar a l'usuari.

- Crear o modificar valor d'una variable de la tasca

Mètode	public void setVariableInstanciaTasca(String varCodi, Object varValor);
Descripció	Modifica el valor d'una variable jBPM de la instància de tasca. Si l'execució d'aquest mètode es realitza a dins un handler no associat a una instància de tasca es produirà una excepció. Si la variable no existeix, aquesta es crearà.
Paràmetres	• varCodi: Codi de la variable a modificar. • varValor: Nou valor de la variable.
Valor retornat	

- Obtenir el valor d'una variable del procés

Mètode	public Object getVariableInstanciaProces(String varCodi);
Descripció	Obté el text que es mostra a Helium d'una variable de jBPM de la instància de procés.
Paràmetres	• varCodi: Codi de la variable a llegir.
Valor retornat	El valor de la variable.

- Obtenir el text d'una variable del procés

Mètode	public Object getVariableInstanciaProcesText(String varCodi);
Descripció	Obté el text que es mostra a Helium d'una variable de jBPM de la instància de procés. Depenent del tipus de variable, el valor es formateja o es consulta abans de mostrar-lo a l'usuari.
Paràmetres	• varCodi: Codi de la variable a llegir.
Valor retornat	El valor de la variable formatat per mostrar a l'usuari.

- Crear o modificar valor d'una variable del procés

Mètode	public void setVariableInstanciaProces(String varCodi, Object varValor);
Descripció	Modifica el valor d'una variable jBPM de la instància de procés.
Paràmetres	• varCodi: Codi de la variable a modificar. • varValor: Nou valor de la variable.
Valor retornat	

- Obtenir expedient actual

Mètode	public Expedient getExpedient();
Descripció	Obté la informació de l'expedient a on s'està executant el handler.
Paràmetres	
Valor retornat	Un objecte amb la informació de l'expedient.

- Llançar error de validació de tasca

Mètode	public void errorValidacioInstanciaTasca(String error);
Descripció	Llença un error de validació per a mostrar a l'usuari dins la tramitació d'una tasca. Si l'execució d'aquest mètode es realitza a dins un handler no associat a una instància de tasca es produirà una excepció.
Paràmetres	• error: El text de l'error que es vol mostrar a l'usuari.
Valor retornat	

- Consultar expedients

Mètode	public List<Expedient> consultaExpedients(String titol, String numero, Date dataInici1, Date dataInici2, String expedientTipus,

	String estat, boolean iniciat, boolean finalitzat);
Descripció	Realitza una consulta d'expedients donat un filtre i retorna el resultat. Per motius de rendiment, no es podran retornar més de 100 expedients.
Paràmetres	• titol: Part del títol de l'expedient. • numero: Número de l'expedient. • dataInici1: Valor iniciar del filtre per data d'inici. • dataInici2: Valor final del filtre per data d'inici. • expedientTipus: Retorna només expedients amb el codi de tipus d'expedient indicat. • estat: Retorna només expedients amb el codi d'estat indicat. • iniciat: Retornar només expedients en estat iniciat. • finalitzat: Retornar només expedients en estat finalitzat.
Valor retornat	Una llista amb els expedients trobats segons el criteri de consulta.

- Consultar domini (ws o sql)

Mètode	public List<FilaResultat> consultaDomini(String codiDomini, String id, Map<String, Object> parametres);
Descripció	Realitza una consulta de domini i retorna el resultat.
Paràmetres	• dominiCodi: Codi del domini a consultar. • id: Id del domini a consultar. • parametres: Paràmetres per a la consulta de domini. ◦ String: codi del paràmetre ◦ Object: valor del paràmetre
Valor retornat	Una llista amb els resultats retornats per la consulta de domini.

- Consultar domini intern

Mètode	public List<FilaResultat> consultaDominiIntern(String id, Map<String, Object> parametres);
Descripció	Realitza una consulta de domini i retorna el resultat.
Paràmetres	• id: Id del domini a consultar. • parametres: Paràmetres per a la consulta de domini. ◦ String: codi del paràmetre ◦ Object: valor del paràmetre
Valor retornat	Una llista amb els resultats retornats per la consulta de domini intern.

- Consultar enumeració

Mètode	public List<ParellaCodiValor> consultaEnumeracio(String codiEnumeracio);

Descripció	Realitza una consulta dels valors d'una enumeració.
Paràmetres	• codiEnumeracio:
Valor retornat	Una llista amb els resultats retornats per la consulta de domini.

- Enviar correu electrònic

Mètode	public void enviarEmail(List<String> recipients, List<String> ccRecipients, List<String> bccRecipients, String subject, String text, List<String> attachments);
Descripció	Envia un correu electrònic amb la possibilitat d'adjuntar documents de l'expedient.
Paràmetres	• recipients: Llista de destinataris. • ccRecipients: Llista de destinataris CC. • bccRecipients: Llista de destinataris BCC. • subject: Assumpte del missatge. • text: Contingut del missatge. • attachments: Llista amb els codis de document a adjuntar al missatge.
Valor retornat	

- Obtenir document

Mètode	public Document getDocument(String documentCodi);
Descripció	Obté un document de l'expedient.
Paràmetres	• documentCodi: El codi del document.
Valor retornat	La informació del document.

- Realitzat registre d'entrada

Mètode	public RespostaRegistre registreEntrada(DadesRegistreEntrada dadesEntrada, List<String> documentsEntrada);
Descripció	Crea una anotació al registre d'entrada.
Paràmetres	• dadesEntrada: Objecte amb la informació per a efectuar l'anotació. • documentsEntrada: codis de documents per a adjuntar a l'anotació.
Valor retornat	La resposta a l'anotació de registre.

- Realitzar registre de sortida

Mètode	public RespostaRegistre registreSortida(DadesRegistreSortida dadesSortida, List<String> documentsSortida);
Descripció	Crea una anotació al registre de sortida.
Paràmetres	• dadesSortida: Objecte amb la informació per a efectuar l'anotació. • documentsSortida: codis de documents per a adjuntar a l'anotació.
Valor retornat	La resposta a l'anotació de registre.

- Realitzar notificació telemàtica

Mètode	public RespostaRegistre registreNotificacio(DadesRegistreNotificacio dadesNotificacio, List<String> documentsNotificacio);
Descripció	Crea una notificació telemàtica.
Paràmetres	• dadesNotificacio: Objecte amb la informació per a efectuar l'anotació. • documentsEntrada: codis de documents per a adjuntar a l'anotació.
Valor retornat	La resposta a l'anotació de registre.

- Obtenir justificant de recepció

Mètode	public JustificantRecepcio obtenirJustificantRecepcio(String registreNumero);
Descripció	Obté la el justificant de recepció d'una notificació telemàtica.
Paràmetres	• registreNumero: El número de registre de la notificació telemàtica.
Valor retornat	La informació relativa al justificant de recepció.

- Relacionar expedients

Mètode	public void expedientRelacionar(Long expedientId);
Descripció	Crea una relació entre l'expedient actual i l'expedient indicat.
Paràmetres	• expedientId: L'identificador de l'expedient a relacionar amb l'actual.
Valor retornat	

- Aturar expedient

Mètode	public void expedientAturar(String motiu);
Descripció	Atura la tramitació de l'expedient actual.
Paràmetres	• motiu: El motiu pel qual s'atura l'expedient.
Valor retornat	

- Reprendre expedient

Mètode	public void expedientReprendre();
Descripció	Reprèn lla tramitació de l'expedient actual. Si l'expedient no es troba aturat es produirà una excepció.
Paràmetres	
Valor retornat	

- Reindexar expedient

Mètode	public void expedientReindexar();
Descripció	Executa una reindexació de l'expedient actual.
Paràmetres	
Valor retornat	

- Redirigir token

Mètode	public void expedientTokenRedirigir(Token token, String nodeName, boolean cancelarTasques);
Descripció	Redirigeix un token de l'expedient cap a un node determinat.
Paràmetres	• token: El token a redirigir. • nodeName: El nom del node destí de la redirecció. • cancelarTasques: Indica si s'han de cancel·lar les tasques actives del token abans de fer la redirecció.
Valor retornat	

- Desar paràmetres pel retrocés

Mètode	public void retrocedirGuardarParametres(List<String> parametres);
Descripció	Emmagatzema els paràmetres per a la retroacció.
Paràmetres	• parametres: Els paràmetres a guardar per la retroacció.
Valor retornat	

- Desar informació de l'execució

Mètode	public void desarInformacioExecucio(String missatge);
Descripció	Permet desar missatges que es mostraran en temps real, quan la acció s'executa en la finalització d'una tasca en segon pla.
Paràmetres	• missatge: Els missatge que es vol mostrar a l'usuari.

Valor retornat	

e) Handler d'exemple.

Per tal de veure com s'utilitzen tots els mètodes de l'HeliumApi posarem un handler de exemple amb totes els mètodes:

```
package com.sample.action;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import net.conselldemallorca.helium.jbpm3.api.HeliumActionHandler;
import net.conselldemallorca.helium.jbpm3.api.HeliumApi;
import net.conselldemallorca.helium.jbpm3.handlers.exception.HeliumHandlerException;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.ActionInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.DadesRegistreEntrada;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.DadesRegistreNotificacio;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.DadesRegistreSortida;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.DocumentInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.EventInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.ExpedientInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.FilaResultat;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.JustificantRecepcioInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.NodeInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.ParellaCodiValor;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.ProcessDefinitionInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.ProcessInstanceInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.RespostaRegistre;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.TaskInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.TaskInstanceInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.TimerInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.TokenInfo;
import net.conselldemallorca.helium.jbpm3.handlers.tipus.TransitionInfo;


public class ApiHandler extends HeliumActionHandler {

   private String msg = "";
   private String varCodi = "";

   @Override
   public void execute(HeliumApi api) throws HeliumHandlerException {
      try {

         TokenInfo t = api.getToken();
         System.out.println(">>> Token: " + t);

         NodeInfo n = api.getNode();
         System.out.println(">>> Node: " + n);

         ProcessDefinitionInfo pd = api.getProcessDefinition();
         System.out.println(">>> ProcessDefinition: " + pd);

         ProcessInstanceInfo pi = api.getProcessInstance();
         System.out.println(">>> ProcessInstance: " + pi);

         ActionInfo a = api.getAction();
         System.out.println(">>> Action: " + a);

         EventInfo e = api.getEvent();
         System.out.println(">>> Event: " + e);

         TransitionInfo tr = api.getTransition();
         System.out.println(">>> Transition: " + tr);

         TaskInfo ta = api.getTask();
         System.out.println(">>> Task: " + ta);
```

```
TaskInstanceInfo ti = api.getTaskInstance();
System.out.println(">>> TaskInstance: " + ti);

ProcessInstanceInfo spi = api.getSubProcessInstance();
System.out.println(">>> SubProcessInstance: " + spi);

TimerInfo tm = api.getTimer();
System.out.println(">>> Timer: " + tm);


api.desarInformacioExecucio("Iniciant ApiHandler: " + msg);

Object val = null;
String sval = null;

// Variables
System.out.println(">>> SetVariableInstanciaTasca " + varCodi + ": val02");
api.setVariableInstanciaTasca(varCodi, "cod_01");
System.out.println(">>> SetVariableInstanciaProces " + varCodi + ": val03");
api.setVariableInstanciaProces(varCodi, "cod 02");

val = api.getVariable(varCodi);
System.out.println(">>> GetVariable: " + varCodi + ": " + val);
sval = api.getVariableText(varCodi);
System.out.println(">>> GetVariableText: " + varCodi + ": " + sval);
val = api.getVariableInstanciaTasca(varCodi);
System.out.println(">>> GetVariableInstanciaTasca: " + varCodi + ": " + val);
sval = api.getVariableInstanciaTascaText(varCodi);
System.out.println(">>> GetVariableInstanciaTascaText: " + varCodi + ": " + sval);
val = api.getVariableInstanciaProces(varCodi);
System.out.println(">>> GetVariableInstanciaProces: " + varCodi + ": " + val);
sval = api.getVariableInstanciaProcesText(varCodi);
System.out.println(">>> GetVariableInstanciaProcesText: " + varCodi + ": " + sval);



// Expedient
ExpedientInfo exp = api.getExpedient();
System.out.println(">>> GetExpedient: " + exp.toString());

// Error de validació de tasca
api.errorValidacioInstanciaTasca("Error de validació de la tasca");

// Consultes
List<FilaResultat> resultatDomini = api.consultaDomini(
      "poblacio",
      "070010001700",
      null); //Map<String, Object> parametres);
System.out.println(">>> ConsultaDomini: " + resultatDomini.toString());

Map<String, Object> parametresDomini = new HashMap<String, Object>();
parametresDomini.put("area", "pruebas");
parametresDomini.put("entorn", "Test");
List<FilaResultat> resultatDominiIntern = api.consultaDominiIntern(
      "PERSONES_AMB_AREA",
      parametresDomini);
System.out.println(">>> ConsultaDomini: " + resultatDomini.toString());

List<ParellaCodiValor> resultatEnumeracio = api.consultaEnumeracio("enum_proves");
System.out.println(">>> ConsultaEnumeracio: " + resultatEnumeracio.toString());

List<ExpedientInfo> exps = api.consultaExpedients(
      null,
      null,
      null,
      null,
      "EXPMOD",   // Codi expedient tipus
      null,
      false,
      false);

System.out.println(">>> ConsultaExpedients: ");
for (ExpedientInfo expe: exps) {
   System.out.println(">>>>>> " + expe);
}
```

```
    List<String> recipients = new ArrayList<String>();
    List<String> adjunts = new ArrayList<String>();
    recipients.add("sandreu@limit.es");
    adjunts.add("doc1");

    api.enviarEmail(
          recipients,
          null,
          null,
          "Titol del missatge",
          "Cos del missatge",
          adjunts);

    DocumentInfo doc = api.getDocument("doc1");
    System.out.println(">>> ConsultaExpedients: " + doc.getArxiuNom());

    DadesRegistreEntrada dadesEntrada = new DadesRegistreEntrada();
    dadesEntrada.setOrganCodi("1100");
    dadesEntrada.setOficinaCodi("11-1199");
    dadesEntrada.setInteressatEntitatCodi("1100");
    dadesEntrada.setInteressatNomAmbCognoms("Limit Tecnologies");
    dadesEntrada.setInteressatMunicipiNom("Manacor");
    dadesEntrada.setAnotacioIdiomaCodi("ca");
    dadesEntrada.setAnotacioTipusAssumpte("OF");
    dadesEntrada.setAnotacioAssumpte("Prova registre entrada");
    List<String> documentsEntrada = adjunts;

    RespostaRegistre entrada = api.registreEntrada(
          dadesEntrada,
          documentsEntrada);
    System.out.println(">>> RegistreEntrada: " + entrada.getNumero() + " "
                        + entrada.getData());

    DadesRegistreSortida dadesSortida = new DadesRegistreSortida();
    dadesSortida.setOrganCodi("1100");
    dadesSortida.setOficinaCodi("11-1199");
    dadesSortida.setInteressatEntitatCodi("1100");
    dadesSortida.setInteressatNomAmbCognoms("Limit Tecnologies");
    dadesSortida.setInteressatMunicipiNom("Manacor");
    dadesSortida.setAnotacioIdiomaCodi("ca");
    dadesSortida.setAnotacioTipusAssumpte("OF");
    dadesSortida.setAnotacioAssumpte("Prova registre sortida");
    List<String> documentsSortida = adjunts;

    RespostaRegistre sortida = api.registreSortida(
          dadesSortida,
          documentsSortida);
    System.out.println(">>> RegistreSortida: " + sortida.getNumero() + " "
                        + sortida.getData());

    RespostaRegistre notificacio = api.registreNotificacio(
          new DadesRegistreNotificacio(),
          adjunts);
    System.out.println(">>> RegistreNotificacio: " + notificacio.getNumero() + " "
                        + notificacio.getData());

    JustificantRecepcioInfo justificant = api.obtenirJustificantRecepcio(entrada.getNumero());
    System.out.println(">>> ObtenirJustificantRecepcio: " + justificant.getData());

    api.expedientRelacionar(58100L);

    api.expedientAturar("Aturant expedient...");
    api.expedientReprendre();
    api.expedientReindexar();

    api.expedientTokenRedirigir(
          api.getToken().getId(),
          "task-node3",
          false);

    List<String> parametres = new ArrayList<String>();
    parametres.add("aaa");
    parametres.add("111");
    api.retrocedirGuardarParametres(parametres);

} catch (Exception e) {
    e.printStackTrace();
```

```
      }
   }

   @Override
   public void retrocedir(HeliumApi api, List<String> args) throws Exception {
      int i = 1;
      for (String arg: args) {
         api.setVariableInstanciaProces("ret" + i++, arg);
         System.out.println("Retrocedim paràmetre: " + arg);
      }
   }

   public void setMsg(String msg) {
      this.msg = msg;
   }
   public void setVarCodi(String varCodi) {
      this.varCodi = varCodi;
   }
}
```