CAPÍTULO 4
Herramientas BPMS

Anahí Rodríguez y Patricia Bazán

La gestión de los procesos de negocio se realiza a través de un sistema de gestión de procesos de negocio (BPMS). Según Weske [Weske M., 2008] un BPMS puede definirse como "un sistema de software genérico que se basa en diseños de procesos explícitos para promulgar y administrar procesos de negocios".

Las herramientas BPMS dan soporte a todo el ciclo de vida de los procesos de negocio y proporcionan mecanismos para mantener el ciclo de vida de los procesos de negocio y obtener mejora continua. Por ejemplo, en la etapa de configuración se puede elegir un BPMS para dar soporte a la implementación y despliegue del proceso de negocio. Luego con ayuda del BPMS se puede monitorear, realizar un seguimiento de la ejecución de cada una de las instancias del proceso de negocio y poder recolectar rastros de ejecución para retroalimentar el ciclo de vida del proceso y así tener una mejora continua.

El papel de los analistas de negocio y el personal técnico de TI queda bien determinado dado que los analistas desarrollan sus análisis y construyen los modelos de procesos sin saber cómo se realiza su ejecución. Luego el personal de TI son los encargados de diseñar y ejecutar los sistemas de información. Contar con una herramienta informática para dicho ciclo de modelado, despliegue y monitorización permite recolectar los rastros de ejecución para retroalimentar el ciclo de vida del proceso y así tener una mejora continua.

La elección de una herramienta BPMS no es una tarea sencilla, dado que la oferta de dichas herramientas es muy variada. Existen varias propuestas para la evaluación de software algunas generales, como ser la evaluación de características de calidad de software definido en la ISO/IEC 9126. Esta evaluación no es específica de los BPMS por eso es indispensable una definición de características propias de estas herramientas para poder realizar una evaluación más correcta o acertada con las necesidades de la organización. En el trabajo propuesto en [Delgado A. et al 2015] propone un total de 94 aspectos a ser evaluados entre los cuales se definen características técnicas y no técnicas.

Otros criterios bien fundamentados para evaluar tecnologías en particular, y BPMS en particular, lo constituyen Gartner, TEC y Forester.

En el caso de Gartner, se trata de una empresa pública estadounidense fundada en 1979 que se dedica a la investigación y análisis de TI para profesionales, empresas de tecnología y la comunidad de la inversión. Sus resultados los genera a partir de varios formatos como

reuniones informativas, servicios de pares en red (*peer networking service*) y programas de socios diseñados explícitamente para CEOs.

Uno de los resultados más esperados en torno a las herramientas de software generadas por Gartner, es el "cuadrante mágico" que consta de una serie de informes de investigación de mercado que se basan en métodos de análisis de datos cualitativos patentados para demostrar las tendencias del mercado, como la dirección, la madurez y los participantes. En [Gartner, 2019] se encuentra el cuadrante mágico 2019 para los BPMS.

En el caso de TEC, Technology Evaluation Centers, es una empresa imparcial de análisis de software que no representa, implementa ni comercializa soluciones de software específicas. Su misión es colaborar con organizaciones de todos los tamaños para identificar las soluciones particulares de software empresarial que mejor se alinean con sus requisitos individuales En [TEC, 2020], TEC realiza una evaluación de BPMS.

Por último, Forrester es una empresa estadounidense de investigación de mercado que brinda asesoramiento sobre el impacto actual y potencial de la tecnología, a sus clientes y al público. En particular, en torno a los BPMS, Forrester produjo el informe que se encuentra en [Forrester, 2013]

4.1. Aspectos funcionales de un BPMS

Además del trabajo de Delgado A. en [Delgado A. et al 2015] y de los aportes mencionados por TEC, Gartner y Forrester, es posible evaluar y elegir herramientas BPMS según aspectos funcionales clásicos y también no clásicos, como se propone en [Rodríguez, A. S., Bazan, P., & Díaz, F. J., 2015].

Los aspectos clásicos son aquellos se pueden vincular de manera directa con las distintas etapas del ciclo de vida de los procesos y que las herramientas BPMS deberían cubrir de manera evidente.

Los aspectos no clásicos se refieren a prestaciones funcionales que no están presentes en muchos BPMS y que aportan una ventaja adicional a los mismos.

Entre los aspectos funcionales clásicos provistos por los BPMS se pueden mencionar:

1. Obtener un gráfico que puede ser de utilidad para los dueños del negocio, como a los analistas para conocer el flujo de trabajo.
2. Simular el proceso de negocio, pudiendo utilizarlo como prueba con datos actuales e históricos.
3. Proporcionar la facilidad de crear automáticamente interfaces y reportes.
4. Proporcionar la facilidad de crear reglas de negocio, pudiendo ser utilizadas para conducir el flujo del proceso y la toma de decisiones.

5. La capacidad para la integración con otras herramientas externas, dado que muchas veces el BPMS no provee todo lo necesario para el ciclo de vida de los procesos de negocio.
6. La capacidad de enviar o recibir mensajes de eventos del sistema o del negocio.
7. La capacidad de reconocer indicadores de desempeño, que se pueden obtener de la ejecución de los procesos de negocio.

Considerando las componentes de un BPMS que llevan a cabo las características funcionales antes mencionadas, se puede concluir que los BPMS deben contemplar, al menos, las siguientes fases en la ejecución del proceso de negocio [BPEL User's Guide, 2007]

- **Modelado del proceso** - El modelado de proceso implica la definición de tareas, ordenarlas, su ramificación, definición de recursos, y otros aspectos del proceso. Con esto podemos tener un modelado del proceso. Este modelo es representado por una gráfica de un flujo y un ordenamiento de tareas involucradas.
- **Instanciación del proceso** - La instancia de un proceso de negocio es cada despliegue del modelo de procesos en tiempo de ejecución. Cada modelo puede tener diferentes instancias una por cada vez que el proceso es iniciado con un caso en particular.
- **Ejecución del proceso** – La ejecución del proceso se lleva a cabo dentro del entorno de ejecución donde se interpreta el modelo del proceso, lo inicializa y lo ejecuta de acuerdo a la información de entrada. Generalmente el entorno de ejecución incluye un motor de procesos, y posiblemente un repositorio para los mismos.
- **Monitorización del proceso** – Al monitorizar el proceso de negocio permite recolectar rastros de ejecución para proponer una mejora en el rendimiento del proceso de negocio.

Entre los aspectos no clásicos que se abordan en [Rodríguez, A. S., Bazan, P., & Díaz, F. J., 2015], se encuentran:

1. **Cobertura de las Etapas del Ciclo de Vida de los Procesos:** para el análisis de los componentes de las herramientas tendremos en cuenta las etapas de un ciclo de vida clásico cíclico, considerando las siguientes fases: Diseño y análisis - Configuración (Definición) - Promulgación (Ejecución) - Evaluación (Monitorización) [Weske M., 2008] [Bazán P.,2009].
2. **Mecanismos de Actualización del Modelo de Procesos de Negocio**: sería útil que la herramienta tenga un mecanismo colaborativo durante el modelado por parte de varios usuarios para el cambio de algún proceso o flujo de trabajo. Si la herramienta utiliza una tecnología WEB 2.0 podría mejorar el trabajo colaborativo independientemente del lugar donde se encuentre el usuario, en esto hay que tener en cuenta algunos riesgos

que debemos afrontar con respecto a la seguridad de los archivos compartidos y el acceso a los mismos [Documentación Bizagi, 2015].

3. **Mecanismos de Actualización de Instancias de Proceso**: ante las distintas instancias que se pueden crear de la ejecución del proceso de negocio, del mismo pueden surgir diferentes subconjuntos de tareas y pueden tener diferentes caminos de ejecución, por lo que los cambios en el mismo pueden ocasionar inconsistencias en el mismo. En los cambios del esquema de flujos de trabajo, hay dos cuestiones a tener en cuenta: se deben analizar la nueva situación excepcional y ver si los cambios se deben propagar hacia abajo en la implementación del proceso de negocio. Los procesos pueden respetar una estructura determinada o variar con el tiempo, ya sea porque la definición de los mismos presenta cambios por naturaleza, o los requerimientos aún no están tan definidos completamente o simplemente porque modelan una realidad que presenta cambios continuos. Los cambios a realizarse en las instancias del proceso, por cambios en el modelo por ejemplo, se pueden realizar una vez terminada la ejecución del mismo, o bien durante la ejecución de las mismas. Hacer los cambios una vez que termine la ejecución de los casos puede ser un inconveniente cuando el tiempo de ejecución de las instancias es muy largo. En ese caso, es importante considerar cuál sería el mecanismo de actualización más adecuado. Los cambios en la ejecución del proceso, en las instancias del mismo, pueden ser causados por eventos no planificados, excepciones, o información errónea en la entrada a las tareas. Ante los cambios en las instancias, la herramienta BPMS debe tener en cuenta: ¿Cómo se resuelven los cambios? ¿Qué estrategias de cambios siguen? Existen patrones de diseño que tienen en cuenta cambios en las instancias como se muestran en [Documentación Bizagi, 2015]
4. **Capacidad de Distribución de Procesos en Varios Motores**: hoy en día es muy común la colaboración entre procesos de negocio como servicios, lo que se lleva a pensar en flujos de trabajo más dinámicos y flexibles, así como la distribución de los procesos de negocio. La necesidad de escalar y la naturaleza netamente colaborativa que subyace a la ejecución de procesos de negocio, ponen en juego la necesidad de contar con entornos descentralizados que permitan optimizar el uso de la tecnología y dar respuesta a la necesidad de crecimiento de las organizaciones. En la actualidad hay varios métodos para descomponer un proceso de negocio en múltiples procesos que se despliegan en las instalaciones propias y en la nube en función del rendimiento y requisitos de sensibilidad, expresándose los mismos como anotaciones sobre las actividades y los datos. Este modelo de distribución da cuenta de la factibilidad de contar con un modelo de distribución de procesos.
5. **Integración con Portales de Autenticación SSO (*Single Sign On*):** Los sistemas SSO permiten a los usuarios utilizar los servicios de varios sitios web en diversas aplicaciones o servicios web sin identificación y contraseña si los usuarios se autentican en un sitio web. Por lo que SSO ofrece la reducción de costos y comodidad a los administradores de sitios web y a los usuarios [Dae-Hee Seo et al., 2003].

6. **Cumplimiento del Estándar BPMN**: BPMN ha sido desarrollado para proveer a los usuarios de una notación estándar, fue definido por *Object Management Group* (OMG). BPMN está dirigido a usuarios y proveedores de servicios que requieren comunicar los procesos de negocio de una forma estándar [BPEL Tutorial, 2020]. El estándar BPMN v.1.x se definió en el año 2010 luego tuvo mejoras con la versión v.2.0., algunas mejoras son [BPMN, 2019]: 1- formaliza la semántica de la ejecución para todos los elementos de BPMN, 2- agrega nuevos eventos que no interrumpen (*Non-interrupting Events*) y eventos subprocesos entre procesos, 3- genera un metamodelo formal mostrado mediante diagramas de clases, 4- define el modelo de colaboración entre procesos y 5- define el modelo de coreografía.

4.2. Comparación de BPMS

Como hemos visto, las herramientas BPMS cuentan con características funcionales particulares y constituyen un recurso muy importante en el ciclo de vida de los procesos de negocio.

En este capítulo seleccionamos un conjunto de herramientas para compararlas en cuanto al tipo de licencia que utilizan y también en cuanto a los componentes funcionales que componen su arquitectura.

La selección fue realizada teniendo en cuenta el grado de popularidad y el volumen de la comunidad que conforman sus usuarios, al momento de la redacción de este libro. También se tuvieron en cuenta herramientas de software libre o que, al menos, cuenten con una edición con licencia abierta.

En la Tabla 7 se muestran una selección de herramientas sus licencias de uso.

Tabla 7. Herramientas seleccionadas para analizar

Nombre de la herramienta	URL	Licencia
Bonita BPM	https://es.bonitasoft.com/	La edición Community tiene licencia GNU GPL v2
BizAgi	https://www.bizagi.com/	*Módulo Modeler*: FreeSoftware, *Módulo Studio:* FreeSoftware, solo permite un máximo de veinte (20) Usuarios en ambientes de desarrollo *Módulo Engine:* Propietaria
Activiti	https://www.activiti.org/	Apache 2.0
jBPM	https://www.jbpm.org/	Apache 2.0
ProcessMaker	https://www.processmaker.com/	ProcessMaker (GNU General Public License version 3.0) Nayra (Apache 2.0)

BonitaBPM

Es una herramienta con una comunidad muy activa, con contribuciones de códigos fuentes, ayudas en el foro, documentación muy completa, videos de tutoriales, etc. Además, es una herramienta muy popular y con constantes actualizaciones. Esta herramienta está desarrollada en el lenguaje JAVA Consta de cuatro ediciones:

- *Community:* Es la edición Open Source, la cual permite entre las funcionalidades más importantes: modelado, conectores, generación de aplicaciones, manejos de los procesos, etc.
- *Teamwork*: esta edición está basada en los ambientes colaborativos, y entre las funcionalidades más importantes: un repositorio compartido, productividad avanzada para desarrolladores y personas de negocio, etc.
- *Efficiency:* esta edición presenta un ambiente avanzado, además de contar con las características de la edición anterior, también presenta plantillas de procesos y perfiles personalizados, entre otras características.
- *Performance:* esta edición es para un ambiente más crítico, monitorización, tareas de gestión, entre otras características.

BizAgi

Es una herramienta muy popular también. En su sitio web se pueden encontrar cursos, foros y tutoriales para poder utilizar la herramienta de una manera fácil y guiada. Consta de una suite con tres componentes:

- *Modeler:* permite el desarrollo de modelos de procesos de manera visual (*drag and drop*) minimizando la brecha digital entre los grupos de TI y los diseñadores de procesos, colaboración en la nube del diseño. Se rige por el estándar BPMN 2.0.
- *Studio:* permite el despliegue del proceso de negocio modelado, sin necesidad de programación. Incluye toda la información necesaria para realizar la ejecución del proceso, dado permite la definición de los datos de proceso, interfaz de usuario, reglas de negocio, etc.
- *Automation:* Es el componente pago de la suite. Permite la ejecución de los procesos creados con Bizagi Studio. Bizagi Engine permite generar un portal de trabajo web. Contiene varios motores tales como Motor de Workflow, Motor de Reglas de Negocio, Motor de Asignaciones entre otros. Los usuarios finales acceden a cada uno de ellos a través de una aplicación web.

Activiti

Es una herramienta liviana, escrita en lenguaje Java. Permite la extensión de sus funcionalidades dado que el código está disponible por su licencia Apache 2.0, fácilmente integrable con aplicaciones externas. El motor de procesos se puede utilizar como servicio externo dado que tiene una API REST, pudiendo extenderlo con nuevas operaciones.

jBPM

Es una herramienta para el desarrollo y despliegue de un proceso de negocio escrita íntegramente en lenguaje JAVA, se ejecuta en cualquier JVM, posee un editor en Eclipse para la definición y creación del proceso de negocio de manera gráfica. Provee una consola desde la cual permite la gestión de las instancias del proceso, listas de tareas, y gestión de formularios. JBPM puede usarse como servicio. La herramienta realiza un registro de historial, lo que permite luego realizar consultas, monitoreos y análisis de los resultados de la ejecución

ProcessMaker

Es un proyecto que provee tanto módulos de código abierto como ediciones pagas, los productos de código abierto que proporcionan son:

- ProcessMaker BPM 4: es un paquete de software BPM compatible con BPMN 2.0. Provee una RESTful APIs para poder acceder de manera más flexible y escalable a las funcionalidades de ProcessMaker y poder realizar una integración con otras aplicaciones ya existentes.
- *Nayra:* Es un motor de flujo de trabajo BPMN 2.0 construido en PHP.
- *Conectores PM*: provee conectores ya predefinidos con los cuales se puede realizar una integración para lograr una plataforma de integración como servicio.

4.3. Mecanismos de Integración de los BPMS

Como se ha desarrollado extensamente en el Capítulo 3, la integración de aplicaciones constituye un concepto ampliamente abordado por la literatura y de gran utilidad, debido a que es muy frecuente que las organizaciones cuenten con software de distinto origen y requieran ser integrados.

Los BPMS, como herramientas de software, también tienen que poder integrarse, más aún por la función que desempeñan: administran los procesos de negocio que, como se mencionó

también en el Capítulo 3, conforman un mecanismo altamente recomendado para orquestar servicios e integrar funcionalidades preexistentes.

Entre los distintos mecanismos o tecnologías que los BPMS utilizan para resolver la integración se encuentran:

- **Conectores Nativos.** Algunas herramientas BPMS proveen conectores nativos, tales como conectores a servidores de bases de datos o a servidores de mail, con los cuales se pueden realizar las conexiones definiendo simples parámetros de configuración. Además, permiten la posibilidad invocar comandos a ejecutar en dichos servidores con la posibilidad de guardar los resultados en variables de proceso para luego su manipulación dentro del proceso de negocio.
- **Web Services.** En el Capítulo 2 se introdujo el concepto de servicio como componente funcional y se define un web service como una aplicación con la cual se puede interactuar a través de una red. Este tipo de tecnología puede ser utilizada para poder conectar distintas aplicaciones existentes. Esto permite gran interoperabilidad entre distintos sistemas dado que productor y consumidor del servicio solo requieren conocer su interface de comunicación y no la manera en que está implementado.

  En el caso de los BMPS, la integración por web service permite invocar componentes funcionales que den respuesta a la ejecución de las actividades, aunque las mismas estén desarrolladas en otra plataforma o tecnología.
- **API (Application Programming Interface).** Una interfaz de programación de aplicaciones es un conjunto de acciones o simplemente mensajes en los cuales se definen las operaciones que se pueden acceder del software o herramienta. Crear una interfaz de comunicación permite que los productos y servicios establezcan una comunicación, sin importar cómo está implementado. También se lo puede considerar como un “contrato” entre aplicaciones, dado que se establece un acuerdo entre las partes en las cuales se determina como es la comunicación.

  Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos. Las API públicas representan un valor comercial único porque simplifican y amplían la forma en que se conecta con sus partners y, además, pueden rentabilizar sus datos (un ejemplo conocido es la API de Google Maps)[2].

  Actualmente los BPMS, presentan una API de programación con la cual se puede acceder externamente a distintas funcionalidades, como por ejemplo, iniciar procesos o acceder a las variables, entre otros.

[2] https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

4.4. Uso de API en BonitaBPM. Un ejemplo

El motor de procesos de BonitaBPM proporciona una API REST Web. Esta API permite el acceso a todos los objetos del motor, como por ejemplo, procesos, tareas, usuarios o conectores, entre otros, y permiten ejecutar operaciones con ellos - crear, recuperar, actualizar, eliminar-.

El componente de Bonita Engine es el encargado de ejecutar la lógica del flujo de trabajo. Los usuarios pueden gestionar procesos y tareas, y realizar actividades administrativas. La Figura 26 muestra la comunicación entre aplicaciones externas y el Bonita Engine, mediante API REST.

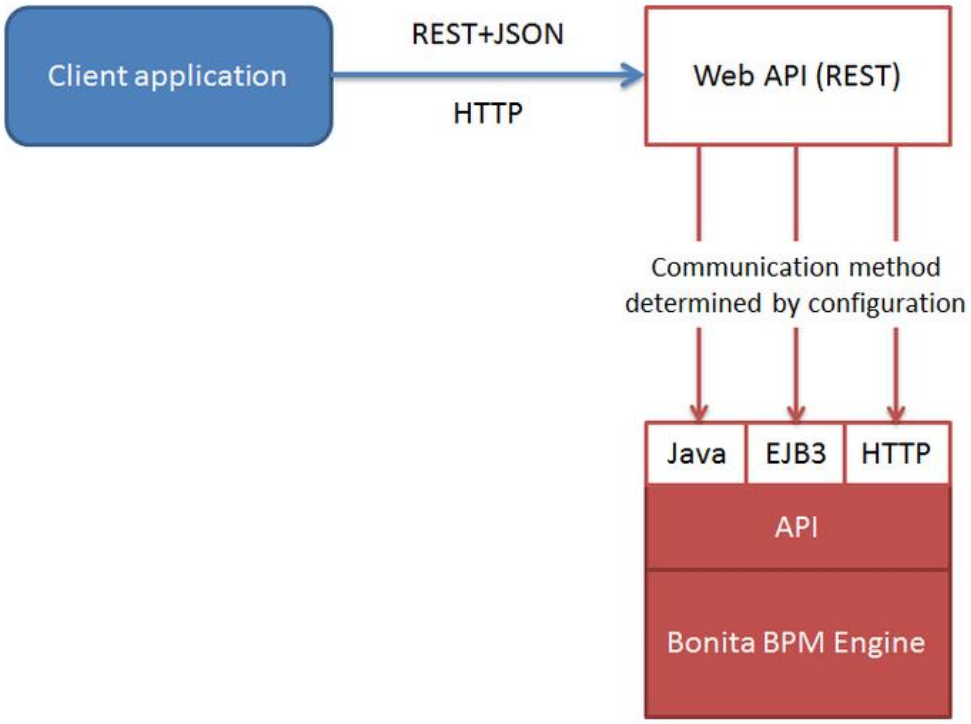


Figura 26. *Comunicación entre un cliente y Bonita Engine por API REST*

Ejemplos de peticiones y uso de API de BonitaBPM

Para poder utilizar la API REST, primero se debe iniciar sesión en el Bonita BPM Engine, de la siguiente manera:

- Request URL: http://host:port/bonita/loginservice
- Request Method: POST
- Content-Type: application/x-www-form-urlencoded
- Form Data:
  - username: a username
  - password: a password

- redirect: true or false.
- redirectURL: la URL de la página que se mostrará después de iniciar sesión

La respuesta a esta llamada genera cookies y el JSESSIONID debe transferirse con cada llamada posterior.

La creación de un Caso es de la siguiente manera:

- URL
  `/API/bpm/case`
- Método
  POST
- Cuerpo de la petición
  Id de proceso en JSON

Obtener una variable de un caso

- URL

`/API/bpm/caseVariable/:caseId /:variableName`

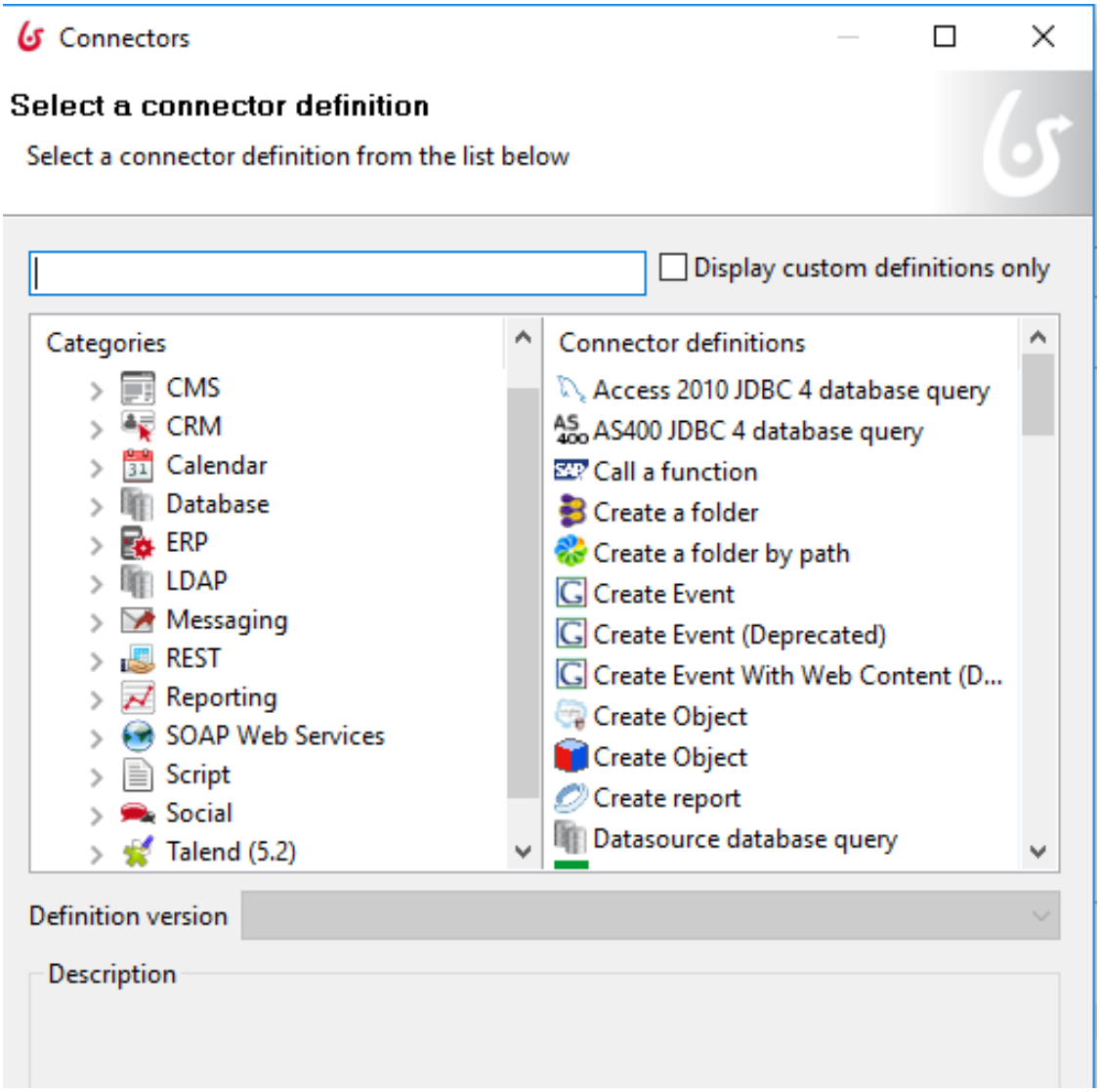
- Método
  GET
- Respuesta exitosa
  Código: 200

```
{
   "descripción":"",
   "name": "myInvoiceAmount",
   "value": "14.2",
   "case_id": "1",
   "type": "java.lang.Float"
}
```

Estos y más métodos se pueden encontrar en la página oficial de Bonita https://documentation.bonitasoft.com/bonita/

Conectores definidos en BonitaBPM

Los conectores definidos en la herramienta, presentan una forma *wizard* para la configuración de los mismos. En la siguiente imagen podemos ver los conectores que están disponibles en la herramienta:



Algunos de ellos son:

- Conectores a Bases de Datos: los cuales permiten realizar conexiones a varios motores populares de base de datos.
- Conectores para envío o recepción de mails: permite hacer conexiones POP3 a los servidores de mail que lo permitan.
- Conectores REST y SOAP: permiten realizar conexiones a servicios REST o SOAP.
- Conectores para Script: permite la creación de Script para su ejecución.

Utilización de conectores nativos - Invocando servicios REST

A continuación, se muestra el wizard que permite la configuración de una petición REST:

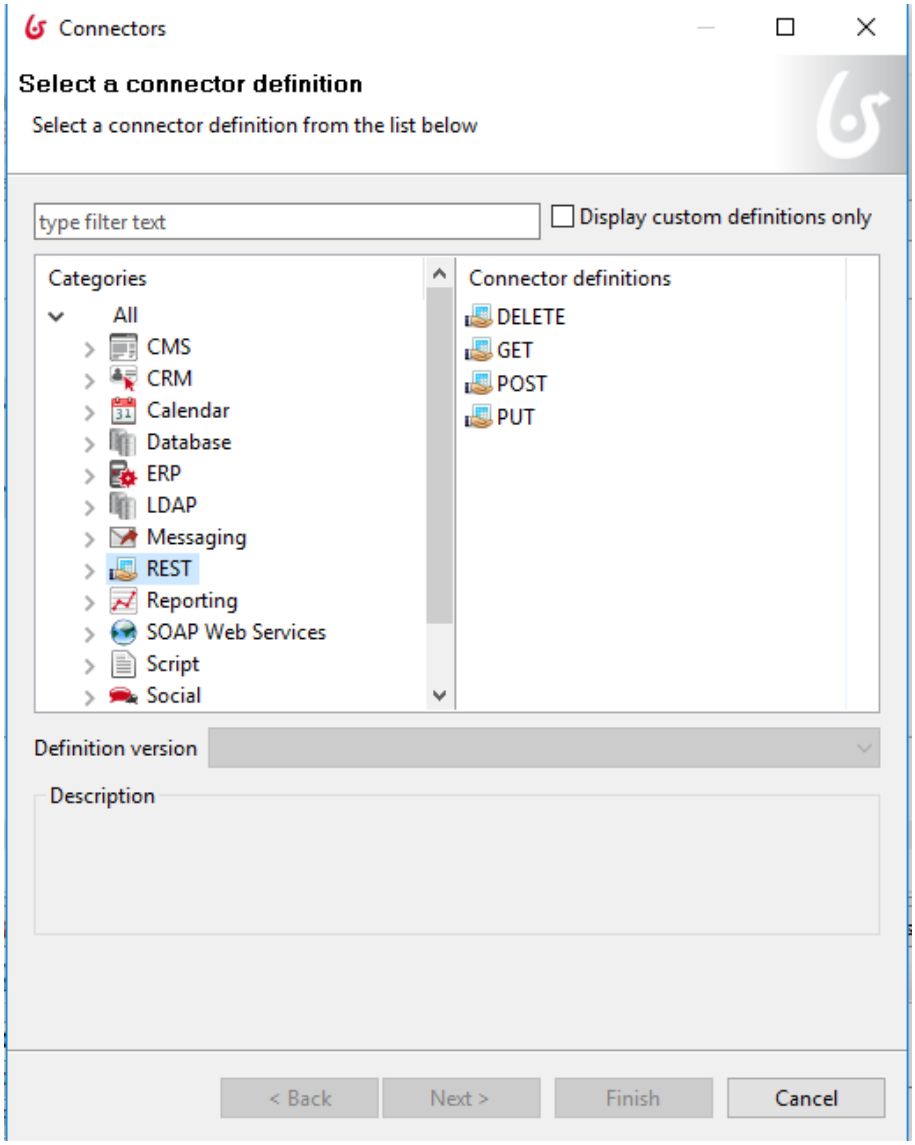


Figura 27. *Selección del tipo de petición.*

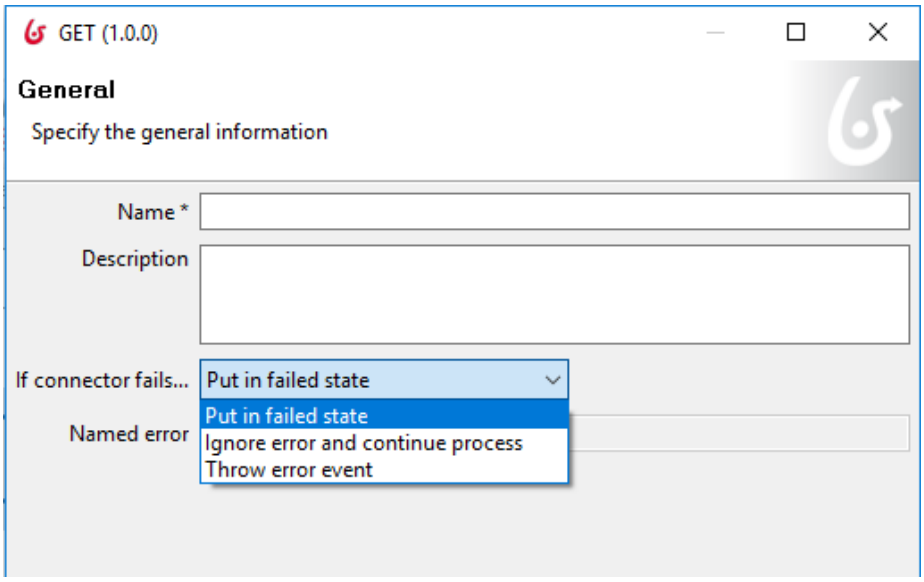


Figura 28. *Configuración del nombre y de seteos por errores de conexión.*

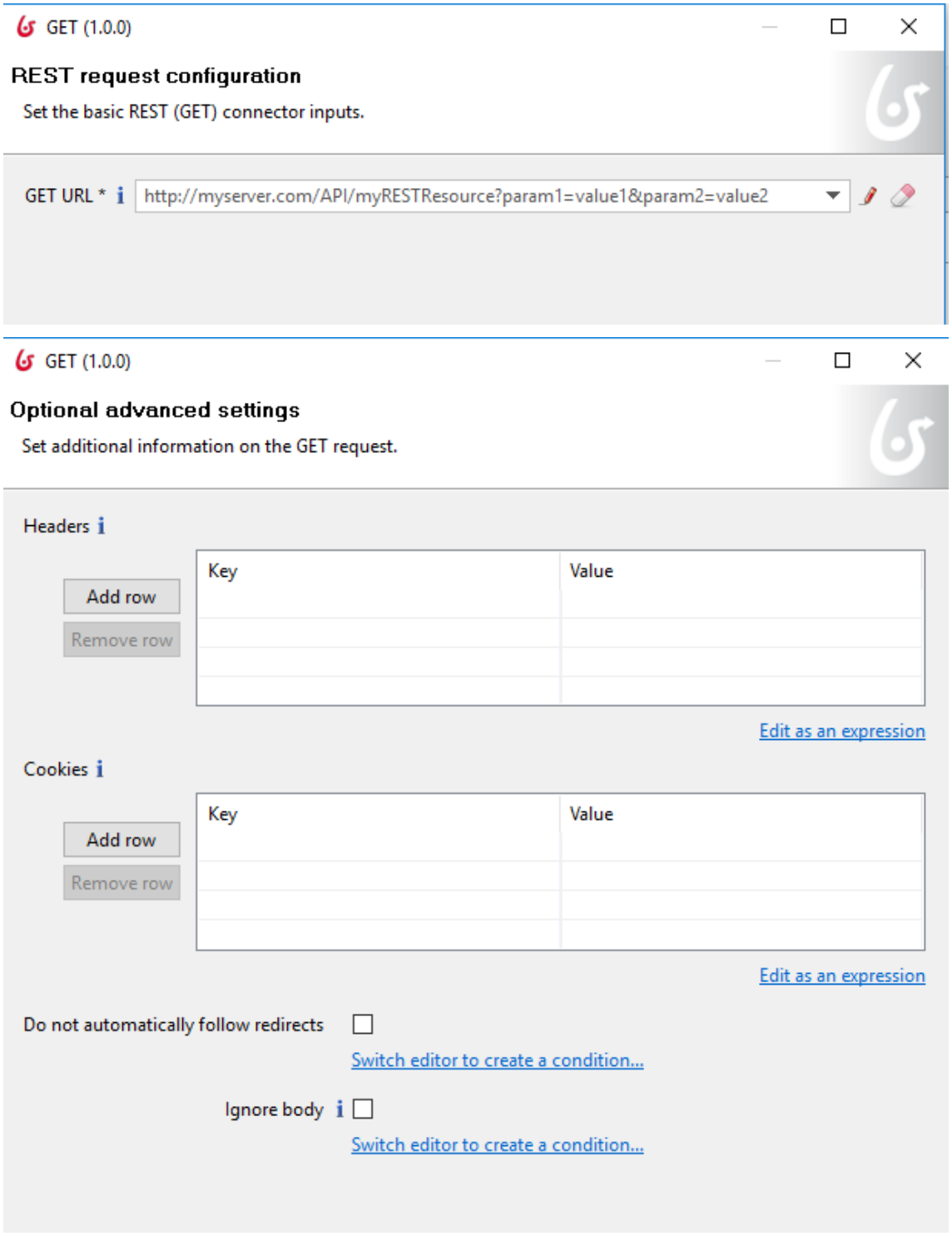


Figura 29. *Configuración del headers y cookies*

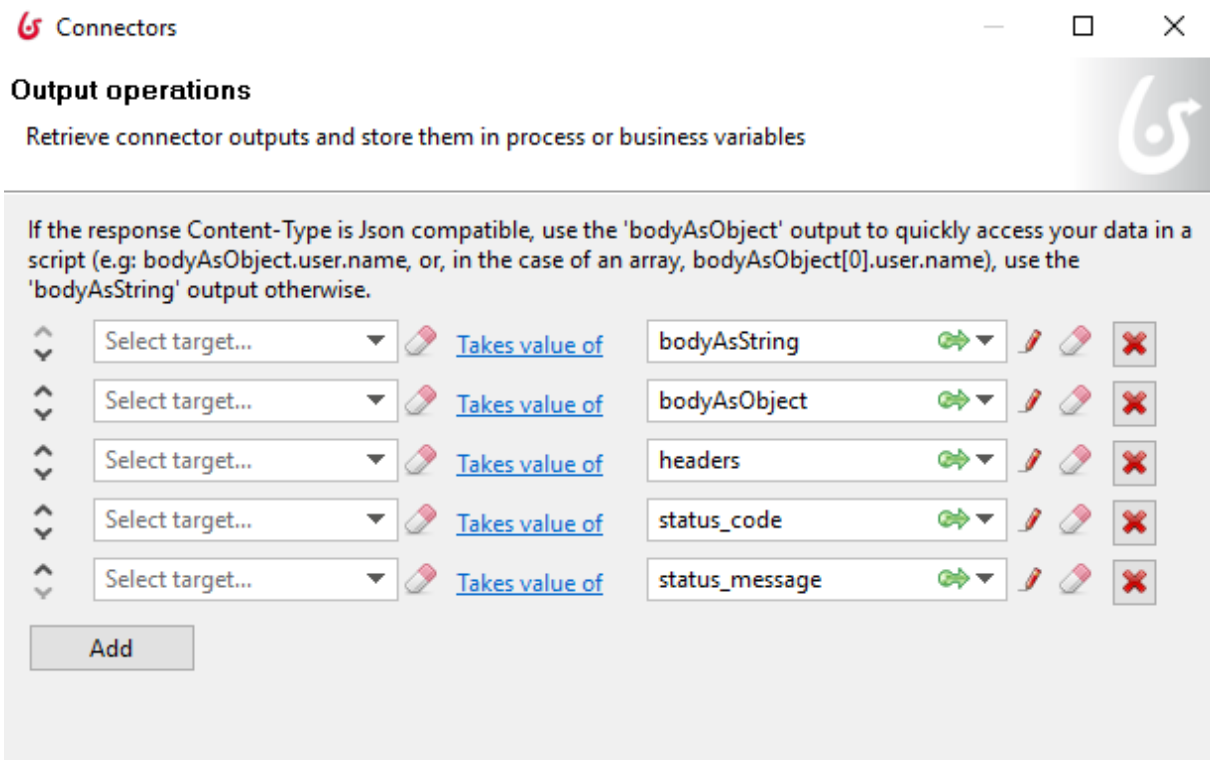


Figura 30. *Seteos de las salidas de la petición realizada, por ejemplo, a través de variables del proceso.*

Creación de nuevos conectores - Definición en una clase de JAVA

Los conectores pueden ser implementados por una clase Java. A través de este mecanismo se amplía la funcionalidad del gestor de procesos, a continuación, se puede ver un ejemplo de conexión a una Base de datos:

Los conectores pueden ser implementados por una clase Java. A través de este mecanismo se amplía la funcionalidad del gestor de procesos:

- Desde la línea de código 2 a la 13 inclusión de dependencias.
- Desde la línea de código 14 a la 21 seteo de parámetros de conexión a las variables
- Desde la línea de código 22 a la 28 manejo de excepciones para la conexión.
- Desde la línea de código 29 a la 42 seteo de parámetros para realizar la conexión.
- Desde la línea de código 44 a la 58 validación de los parámetros.
- Desde la línea de código 60 a la 67 genera la conexión.

A continuación, se puede ver el código desarrollado.

```
1. package org.bonitasoft.connectors.database.jdbc;
2. import java.sql.ResultSet;
3. import java.sql.SQLException;
4. import java.util.ArrayList;
5. import java.util.Collections;
6. import java.util.HashMap;
```

```
7. import java.util.List;
8. import java.util.Map;
9. import java.util.StringTokenizer;
10. import org.bonitasoft.connectors.database.Database;
11. import org.bonitasoft.engine.connector.Connector;
12. import org.bonitasoft.engine.connector.ConnectorException;
13. import org.bonitasoft.engine.connector.ConnectorValidationException;
14. public class JdbcConnector implements Connector {
15.     public static final String USERNAME = "username";
16.     public static final String PASSWORD = "password";
17.     public static final String SCRIPT = "script";
18.     public static final String SEPARATOR = "separator";

19.     private String script;
20.     private Database database;
21.     private ResultSet data;


    @Override
22.   public Map<String, Object> execute() throws ConnectorException {
23.         if (separator != null) {
24.             return executeBatch();
25.         } else {
26.             return executeSingleQuery();
27.         }
28.     }

29.     @Override
30.     public void setInputParameters(final Map<String, Object> parameters) {
31.          userName = (String) parameters.get(USERNAME);
32.         final String paswordString = (String) parameters.get(PASSWORD);
33.         if (paswordString != null && !paswordString.isEmpty()) {
34.             password = paswordString;
35.         } else {
36.             password = null;
37.         }
38.         script = (String) parameters.get(SCRIPT);
39.         separator = (String) parameters.get(SEPARATOR);
40.         driver = (String) parameters.get(DRIVER);
41.         url = (String) parameters.get(URL);
42.     }

43.     @Override
44.     public void validateInputParameters() throws ConnectorValidationExcep-
tion {
45.        final List<String> messages = new ArrayList<String>(0);
```

```
46.         if (url == null || url.isEmpty()) {
47.             messages.add("Url can't be empty");
48.         }
49.         if (driver == null || driver.isEmpty()) {
50.             messages.add("Driver is not set");
51.         }
 52.        if (script == null || script.isEmpty()) {
 53.            messages.add("Script is not set");
 54.        }

 55.        if (!messages.isEmpty()) {
 56.            throw new ConnectorValidationException(this, messages);
 57.        }
 58.    }
 60.  @Override
 61.     public void connect() throws ConnectorException {
 62.        try {
 63.            database = new Database(driver, url, userName, password);
 64.        } catch (final Exception e) {
 65.            throw new ConnectorException(e);
 66.        }
 67.    }
```