

Bonita Open Solution

Conceptos básicos de la notación.
Ejemplos de uso de web services

Temario

- Conceptos de BPM
 - Elementos de BPMN
 - Ejemplos de uso de web services
-

Conceptos

- BPM (Business Process Management por sus siglas en inglés), Gestión de procesos de negocio, es una disciplina que se encarga de proveer mecanismos para el análisis, diseño, implementación y monitoreo de los procesos de negocio que se encuentran en cualquier organización.
 - BPMS (Business Process Management System), Sistema de Gestión de procesos de negocio. Son sistemas que implementan, generalmente en forma de suite, un conjunto de componentes que permiten gestionar las distintas fases del ciclo de vida de los procesos. Ejemplos de BPMS son IBM Websphere, Bonita, Bizagi, entre otros.
-

Conceptos

- Un proceso de negocio es un conjunto de actividades que se encuentran ordenadas en forma lógica en pos de cumplir un objetivo dentro de la organización.
 - De allí la importancia de que los procesos de la organización se encuentren alineados con los objetivos generales de la misma.
-

Conceptos

- BPM como metodología posee una notación para expresar los diagramas que surgen luego de la actividad de análisis.
 - Dicha notación (BPMN - Business Process Management Notation) posee una serie de componentes con una semántica específica, permitiendo diagramar procesos de negocio
-

Definiciones útiles

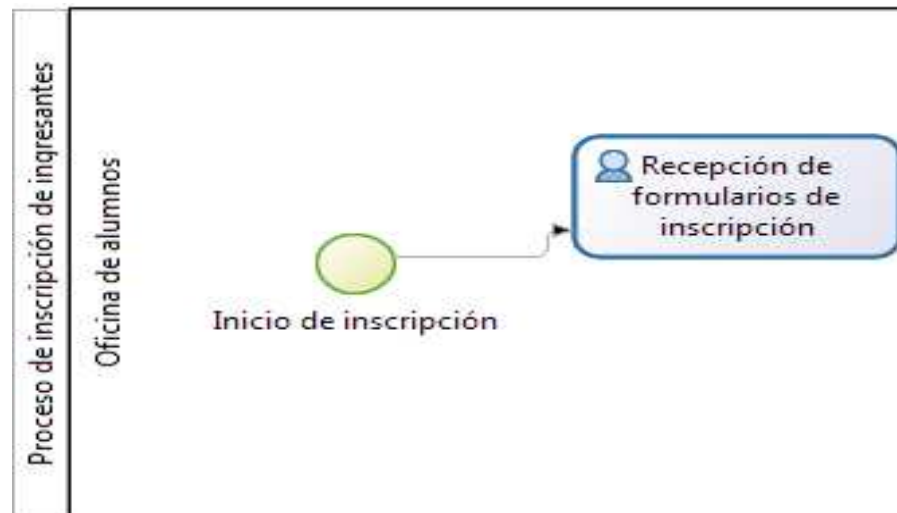
- Diagrama de proceso: es el template del proceso usado como base para luego crear instancias del mismo.
 - Instancia de proceso: es una versión ejecutable del proceso. Son "copias vivas" de una definición de proceso.
 - Archivo de proceso: son las instancias o casos de proceso que ya han finalizado.
 - Actividad: es un paso manual o automatizado dentro del proceso
 - Participante: ente destinado a la ejecución de una actividad.
 - Propietario del proceso: las actividades dentro de un proceso son responsabilidad de personas o sectores dentro de la organización. Las mismas se pueden identificar dentro del diagrama de proceso por medio de pools y lanes
-

Componentes de BPM

- Eventos: son acciones que ocurren a lo largo del flujo del proceso.
 - Pueden ser de distintos tipos:
 - Inicio
 - Intermedios
 - De fin
-

Eventos de inicio

- Son eventos que permiten la iniciación de una instancia de proceso.
- Pueden ser de distinto tipo. En un diagrama de proceso solo puede haber un evento de inicio normal y varios de otros tipos.
- Tipos:
 - Tradicional: indica por default el comienzo de la instancia



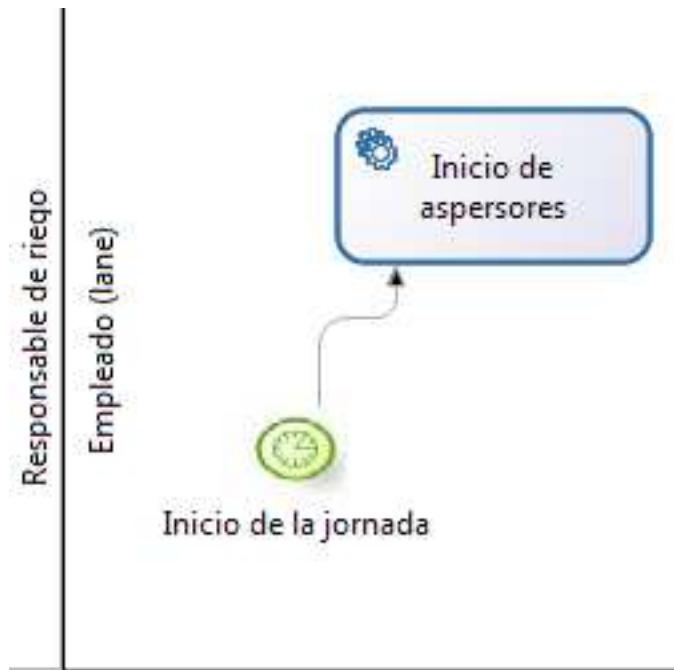
Eventos de inicio

- Por mensaje: la instancia se inicia cuando se recibe un mensaje determinado





Eventos de inicio

- Por temporizador: la instancia se inicia ante la ocurrencia de un determinado plazo temporal



Eventos de inicio

- Inicio por recepción de señal: el inicio de la instancia ocurre cuando el proceso recibe una señal de una instancia determinada. 
- Inicio múltiple: la instancia puede iniciarse por confluencia de varias de las condiciones anteriores. No todos los BPMS lo implementan 



Tareas

- Son unidades de ejecución indivisible dentro del proceso. En caso que sean de usuario deben tener un participante o actor asociado. También existen de tipos automáticos.
- Tipos:
 - Tarea de usuario: los responsables son actores del sistema. Se genera una interacción con dicho sistema



Tareas

- Tarea manual: es ejecutada completamente por mano de obra humana

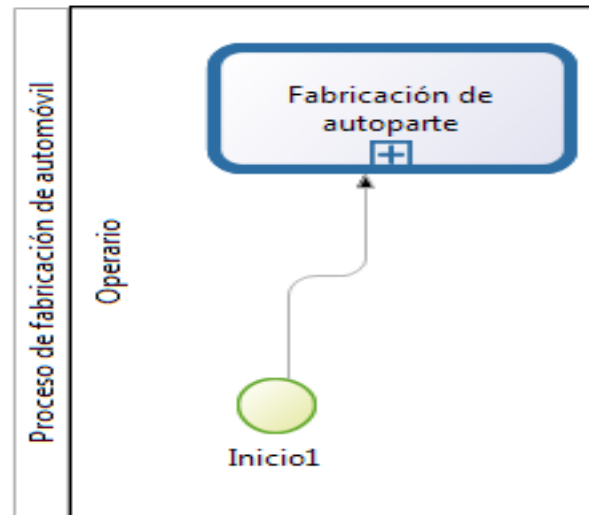


- Tarea automática: se ejecuta por medio de un script o una unidad de software independiente del proceso



Subprocesos

- Son unidades de ejecución divisibles dentro del proceso. Se implementan mediante la sucesión de actividades. Internamente se explotan como un proceso tradicional



Eventos intermedios

- Son hitos de importancia dentro de la ejecución del proceso que puede ser de interés remarcar. Pueden ser de distintos tipos
- Son de tipos análogos a los eventos de inicio. Pueden ser por mensaje, señal o temporizador. En el caso de ser por mensaje, al alcanzarse enviarán un mensaje al destinatario. En el caso de ser por temporizador, establece una pausa en el proceso hasta que se alcance un determinado plazo temporal. En caso de ser por señal, espera hasta el arribo de la misma o emite una señal hacia otro proceso. Existen BPMS que implementan también el tipo por condición (el proceso se detiene hasta que se cumpla una determinada condición.)



Mensaje1



Señal1



Temporizador1

Compuertas

- Permiten evaluar condiciones que indican variaciones en el flujo dentro de la ejecución del proceso.
- Existen de distintos tipos:
 - Compuerta exclusiva: permite seguir solo una de las ramificaciones indicadas. No puede haber más de un camino verdadero.



- Compuerta inclusiva: permite seguir más de una ramificación, en caso de que más de una sea verdadera



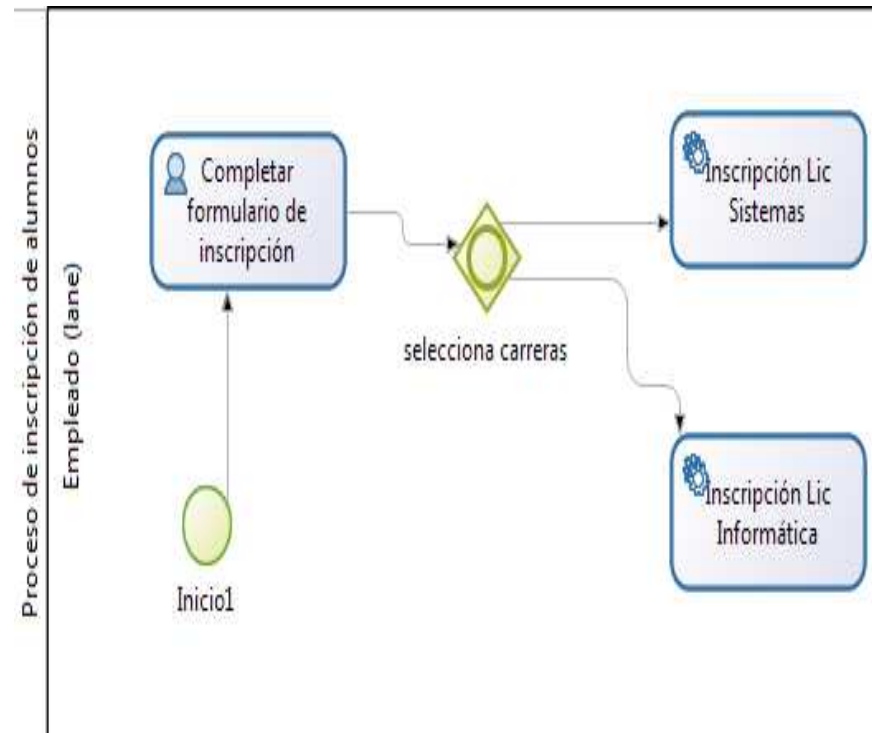
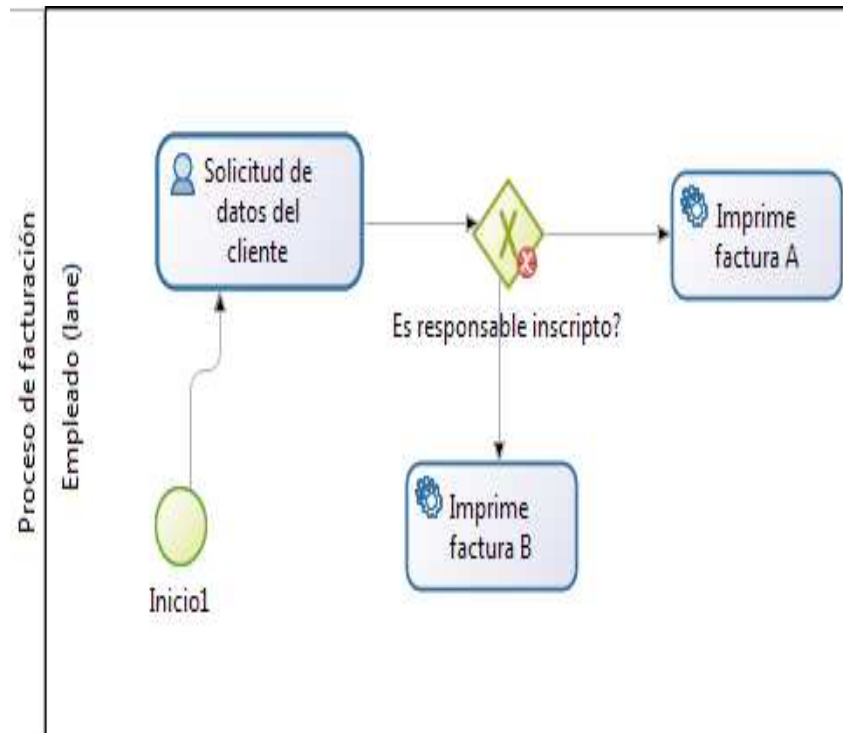
Compuertas

- Compuerta paralela: permite desencadenar la ejecución de varios flujos del proceso en forma simultanea.

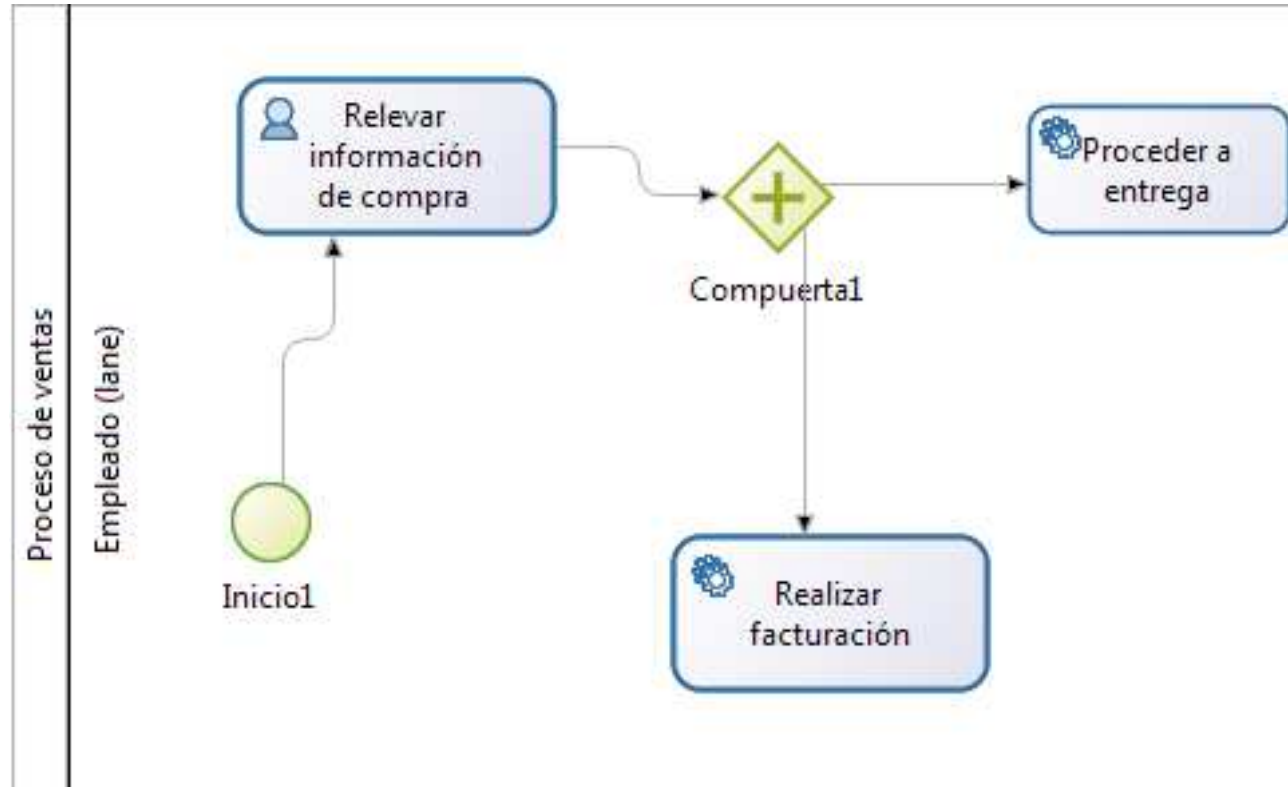


Compuerta3

Ejemplos de compuertas



Ejemplos de compuertas



Eventos de fin

- Permiten indicar la finalización del flujo del proceso. Existen de los tipos análogos a los eventos intermedios y de iniciación.
- Tipos:
 - De fin básico: simplemente finaliza la ejecución de la instancia
 - Mensaje de fin: al finalizar el proceso emite un mensaje
 - Señal de fin: al finalizar la instancia se emite una señal que puede ser recepcionada por otro proceso
 - Evento de terminación: en caso de alcanzarse permite finalizar todas las instancias activas del proceso.



Fin1



Fin2



Fin3

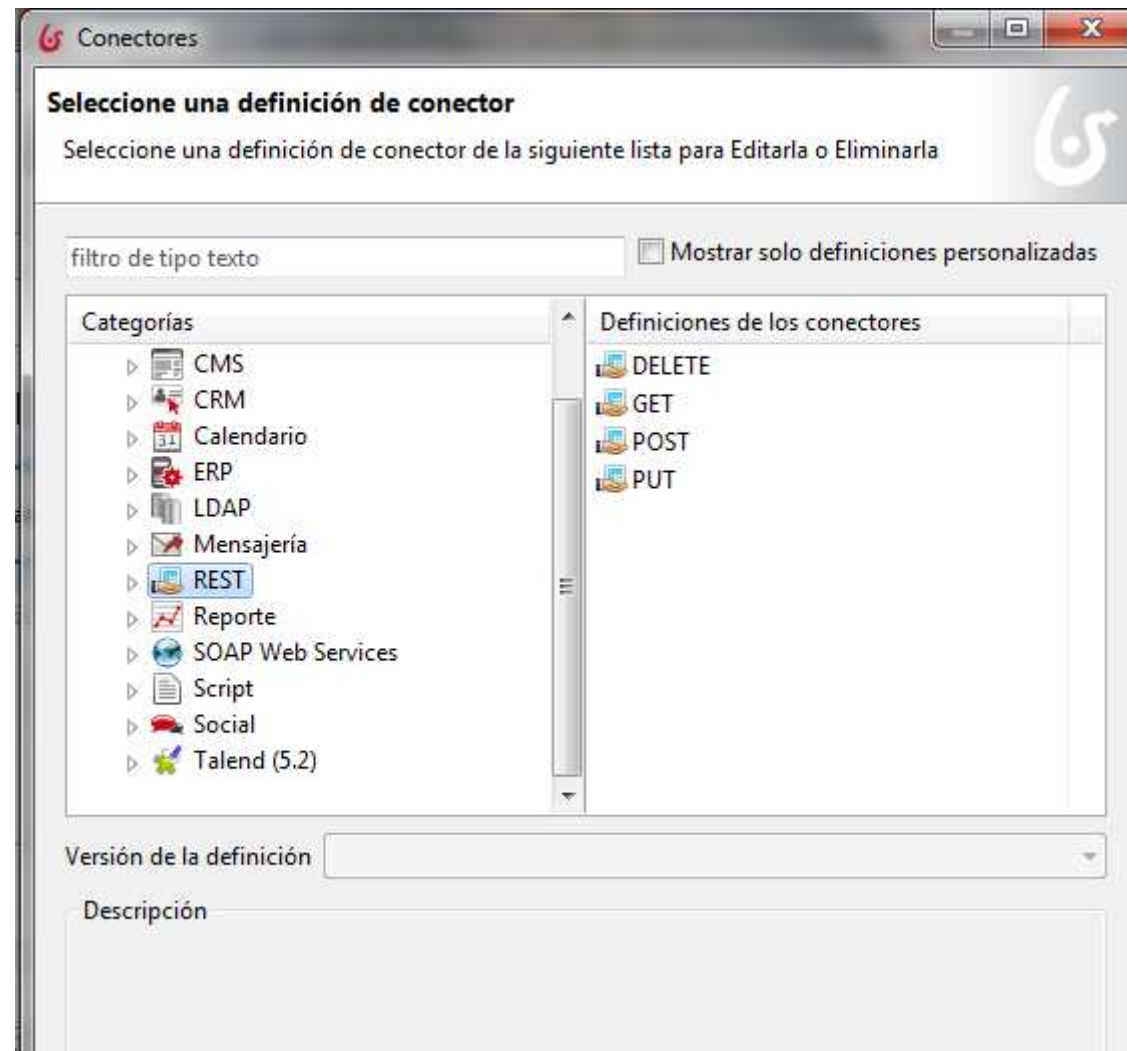


Fin4

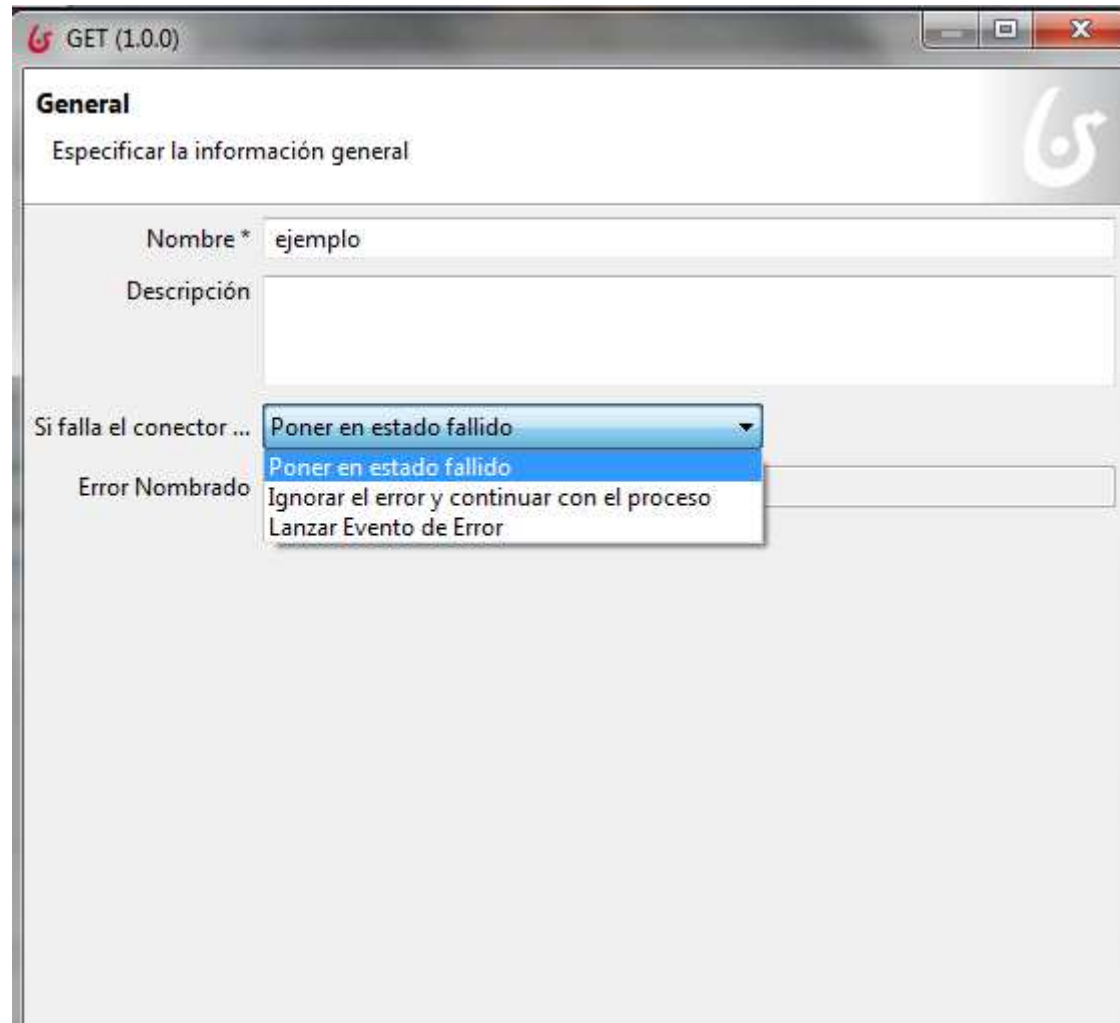
Conectores - Ejemplos

- Bonita extiende el comportamiento y semántica de los procesos a través de componentes no estándar llamados conectores
 - Los mismos incrementan la conectividad del BPMS con sistemas existentes
-

Invocando servicios REST



Servicios REST



GET (1.0.0)

General
Especificar la información general

Nombre * ejemplo

Descripción

Si falla el conector ...

Error Nombrado

- Poner en estado fallido
- Poner en estado fallido
- Ignorar el error y continuar con el proceso
- Lanzar Evento de Error

Servicios REST



Servicios REST

GET (1.0.0)

Configuración avanzada opcional

Añade información adicional a la llamada GET.

Cabeceras ⓘ

[Agregar fila](#)
[Eliminar fila](#)

| Key | Value |
|-----|-------|
| | |
| | |
| | |

[Editar como expresión](#)

Cookies ⓘ

[Agregar fila](#)
[Eliminar fila](#)

| Key | Value |
|-----|-------|
| | |
| | |
| | |

[Editar como expresión](#)

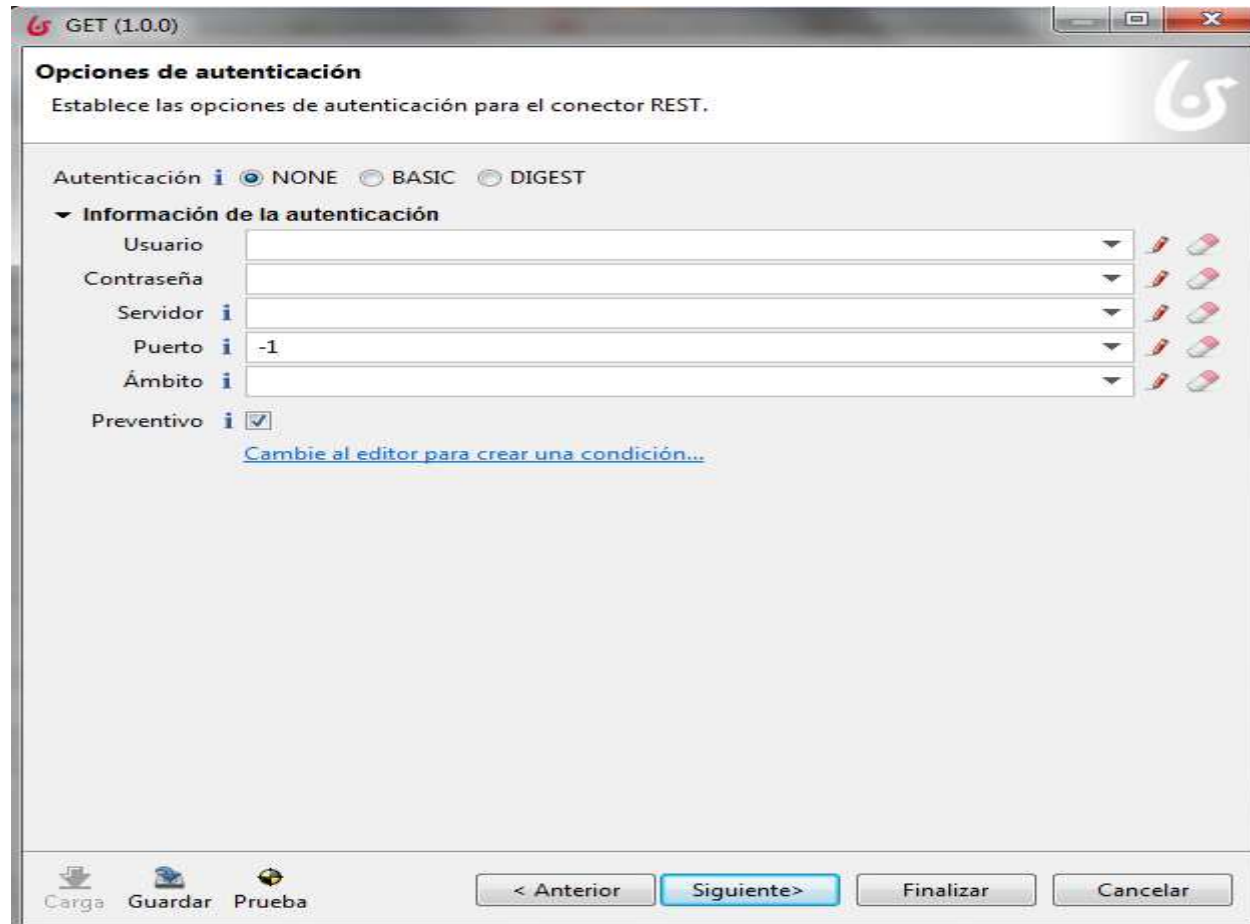
No seguir redirecciones automáticamente. ☐

[Cambie al editor para crear una condición...](#)

Ignorar el cuerpo ⓘ ☐


[Cambie al editor para crear una condición...](#)

Servicios REST

















GET (1.0.0)

Opciones de autenticación
Establece las opciones de autenticación para el conector REST.




Autenticación  ☒ NONE ☐ BASIC ☐ DIGEST

▼ **Información de la autenticación**

| | | |
|--|---------------------------------|---|
| Usuario | <input type="text"/> |   |
| Contraseña | <input type="text"/> |   |
| Servidor  | <input type="text"/> |   |
| Puerto  | <input type="text" value="-1"/> |   |
| Ámbito  | <input type="text"/> |   |

Preventivo  ☒

[Cambie al editor para crear una condición...](#)

 Carga  Guardar  Prueba

< Anterior **Siguiente>** Finalizar Cancelar

Servicios REST

Conectores

Mapea las salidas de este conector a las variables del proceso.

Obten las salidas del conector y almacenalas en las variables de proceso o de negocio

Si la respuesta Content-Type es compatible con Json, usa la salida 'bodyAsObject' para acceder rápidamente a los datos en un script (ejemplo: `bodyAsObject.userName` o, en el caso de un Array, `bodyAsObject[0].userName`). Si no es compatible, usa la salida 'bodyAsString'.

| | | | | |
|----------------------|---------------|----------------|--|--|
| Selecione el destino | Toma valor de | bodyAsString | | |
| Selecione el destino | Toma valor de | bodyAsObject | | |
| Selecione el destino | Toma valor de | headers | | |
| Selecione el destino | Toma valor de | status_code | | |
| Selecione el destino | Toma valor de | status_message | | |

Añadir

< Anterior Siguiente > Finalizar Cancelar

Creación de nuevos conectores

- Bonita permite definir nuevos conectores. Los conectores pueden ser implementados por una clase Java. A través de este mecanismo se amplía la funcionalidad del gestor de procesos
-

Ejemplo de conexión a base de datos usando una clase Java

- **package** org.bonitasoft.connectors.database.jdbc;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.StringTokenizer;

import org.bonitasoft.connectors.database.Database;
import org.bonitasoft.engine.connector.Connector;
import org.bonitasoft.engine.connector.ConnectorException;
import org.bonitasoft.engine.connector.ConnectorValidationException;

public class JdbcConnector **implements** Connector {

 public static final String USERNAME = "username";

 public static final String PASSWORD = "password";

 public static final String SCRIPT = "script";

 public static final String SEPARATOR = "separator";

Conector a clase java

- **private** String script;

private Database database;

private ResultSet data;

@Override

```
public Map<String, Object> execute() throws ConnectorException {  
    if (separator != null) {  
        return executeBatch();  
    } else {  
        return executeSingleQuery();  
    }  
}
```

@Override

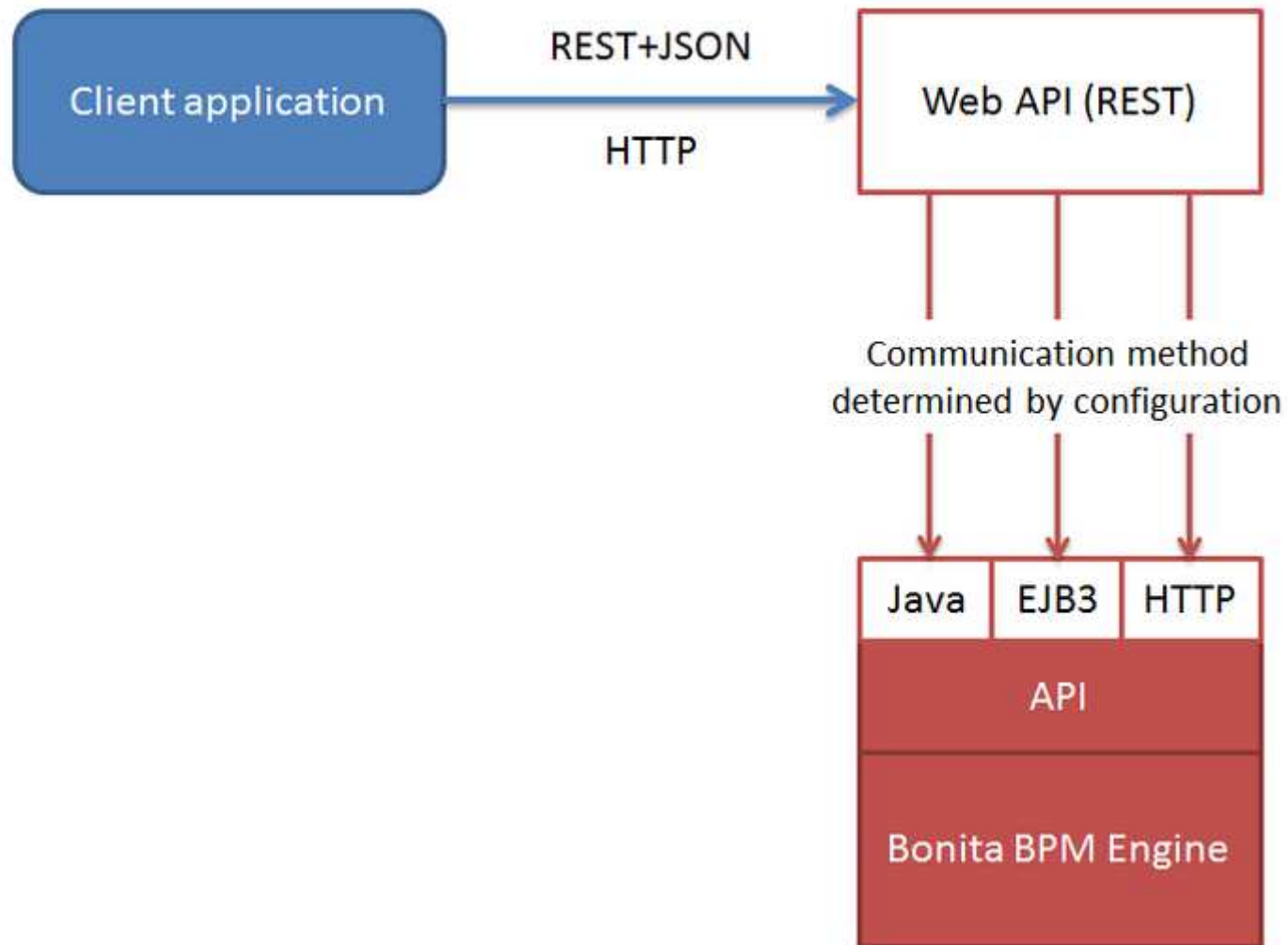
```
public void setInputParameters(final Map<String, Object> parameters) {  
    userName = (String) parameters.get(USERNAME);  
    final String paswordString = (String) parameters.get(PASSWORD);  
    if (paswordString != null && !paswordString.isEmpty()) {  
        password = paswordString;  
    } else {  
        password = null;  
    }  
    script = (String) parameters.get(SCRIPT);  
    separator = (String) parameters.get(SEPARATOR);  
    driver = (String) parameters.get(DRIVER);  
    url = (String) parameters.get(URL);  
}
```

Conector a clase java

- **@Override**
public void validateInputParameters() **throws** ConnectorValidationException {
 final List<String> messages = **new** ArrayList<String>(0);
 if (url == **null** || url.isEmpty()) {
 messages.add("Url can't be empty");
 }
 if (driver == **null** || driver.isEmpty()) {
 messages.add("Driver is not set");
 }
 if (script == **null** || script.isEmpty()) {
 messages.add("Script is not set");
 }

 if (!messages.isEmpty()) {
 throw new ConnectorValidationException(**this**, messages);
 }
}
- @Override**
public void connect() **throws** ConnectorException {
 try {
 database = **new** Database(driver, url, userName, password);
 } **catch** (**final** Exception e) {
 throw new ConnectorException(e);
 }
}

Bonita API Rest



API REST

| | |
|-------------|---|
| Request URL | <code>http://host:port/bonita/loginservice</code> |
|-------------|---|

| | |
|----------------|------|
| Request Method | POST |
|----------------|------|

| | |
|--------------|--|
| Content-Type | <code>application/x-www-form-urlencoded</code> |
|--------------|--|

| | |
|-----------|---|
| Form Data | <code>username: a username</code> <code>password: a password</code> <code>redirect: true or false. false indicates that the service should not redirect to Bonita Portal (after a successful login) or to the login page (after a login failure).</code> <code>redirectURL: the URL of the page to be displayed after login</code> <code>tenant: the tenant to log in to (optional for Enterprise and Performance editions, not supported for Community, Teamwork and Efficiency editions)</code> |
|-----------|---|

API REST

Create a case

This way of creating a case using this method will only work for processes in which no contract is defined. To instantiate a process with a contract, check the [process instantiation resource documentation](#).

- **URL**
`/API/bpm/case`
- **Method**
`POST`
- **Request Payload**
The process definition id, in JSON

Create a case without variables

```
{  
  "processDefinitionId": "5777042023671752656"  
}
```

Create a case with variables

⚠ Warning: The attribute "variables" on the request payload is used to initialize the process variables (not BDM variables). If you want to initialize BDM variables at process instantiation, add a contract on the process and map BDM variables to the contract data. See **Start a process using an instantiation contract** for usage.

API REST

Get a case variable

- URL

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

- Method

`GET`

- Success Response

A case variable representation

- **Code:** 200

- **Payload:**

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

API REST

Update a case variable

Warning: only following types are supported for *javaTypeclassname*: java.lang.String, java.lang.Integer, java.lang.Double, java.lang.Long, java.lang.Boolean, java.util.Date

- URL

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

- Method

`PUT`

- Request Payload

```
{
  type: "javaTypeclassname",
  value: "newValue"
}
```

- Success Response

- Code: 200
-