
Bonita Open Solution

Ejemplos de uso de web services y de
acceso a la base de datos

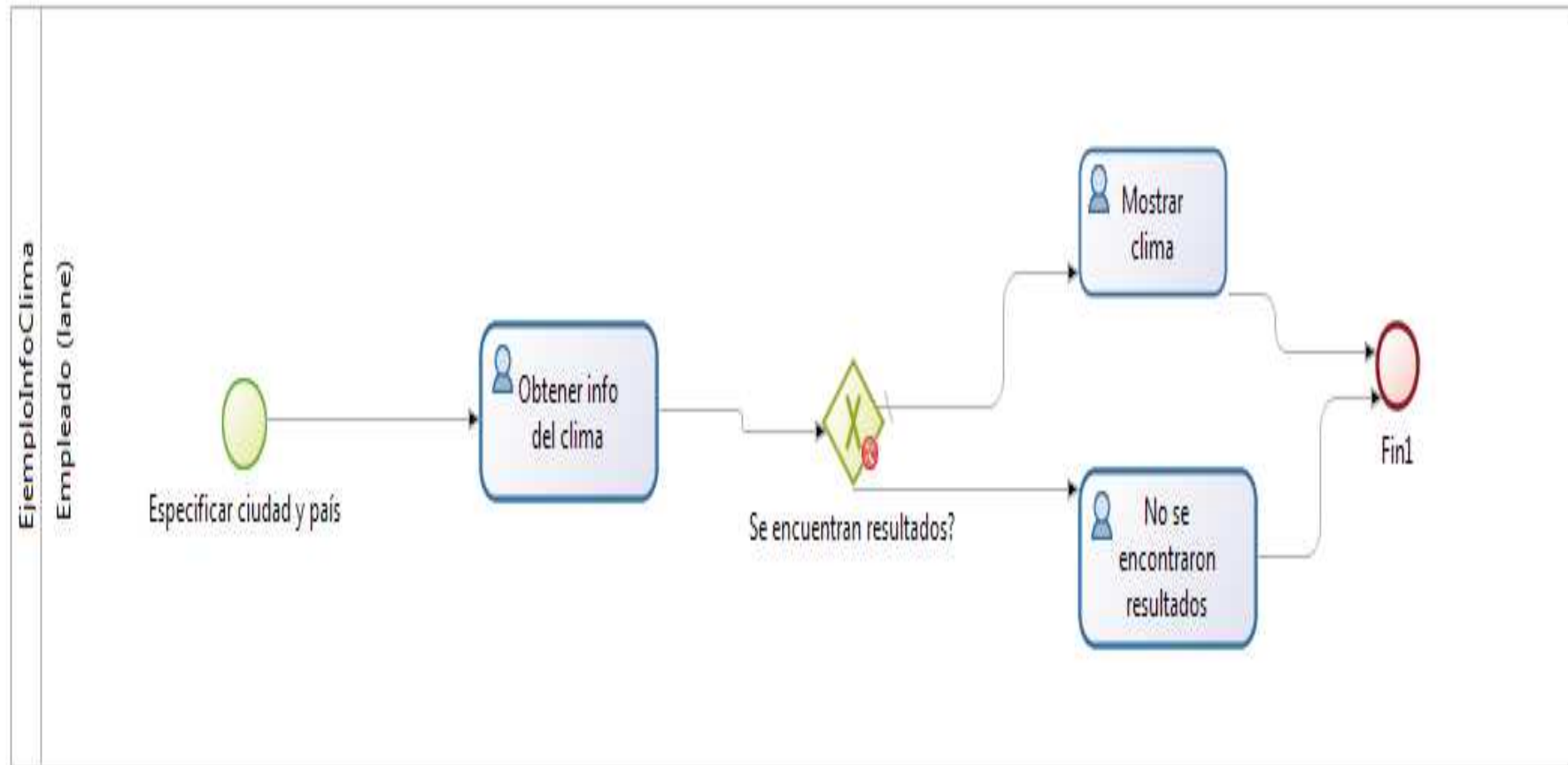
Temas

- Se planteará un ejemplo donde se puedan visualizar las distintas configuraciones necesarias para acceder a un web service, así como a una conexión con una base de datos
-

Primer ejemplo

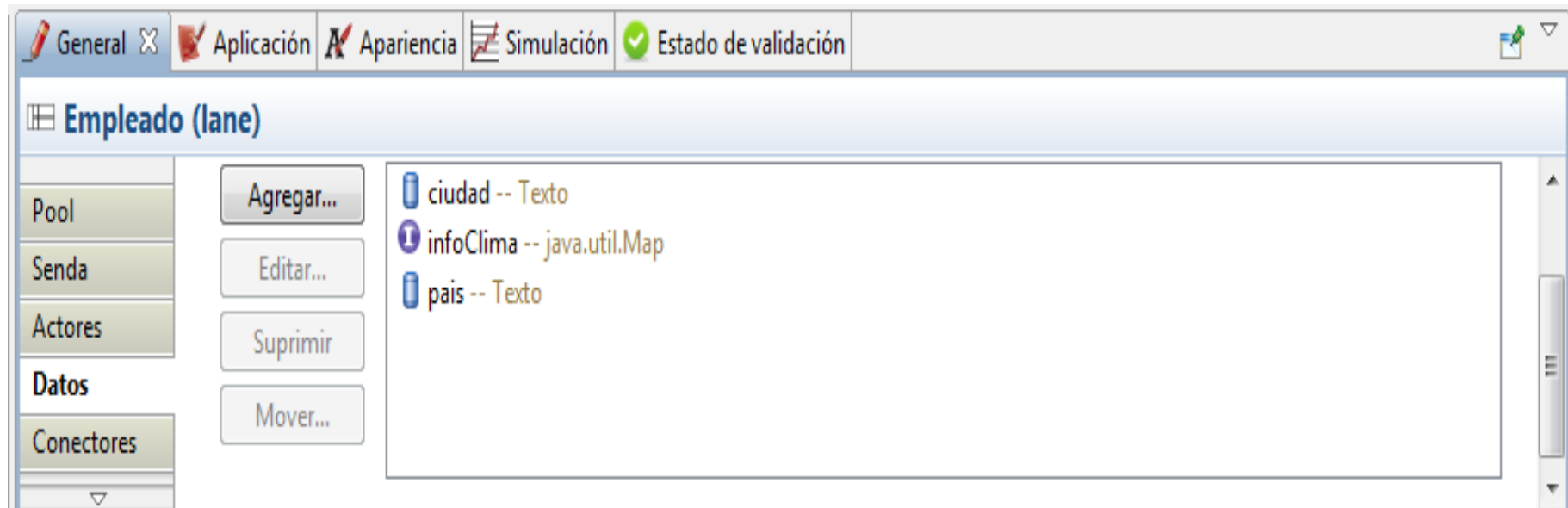
- Se posee una definición de web service que permite obtener información sobre las condiciones del clima. Se deberá ingresar información de localización, y en base a la misma se mostrará el contenido disponible. Si no se posee información se indicará tal situación.
-

Modelo de proceso



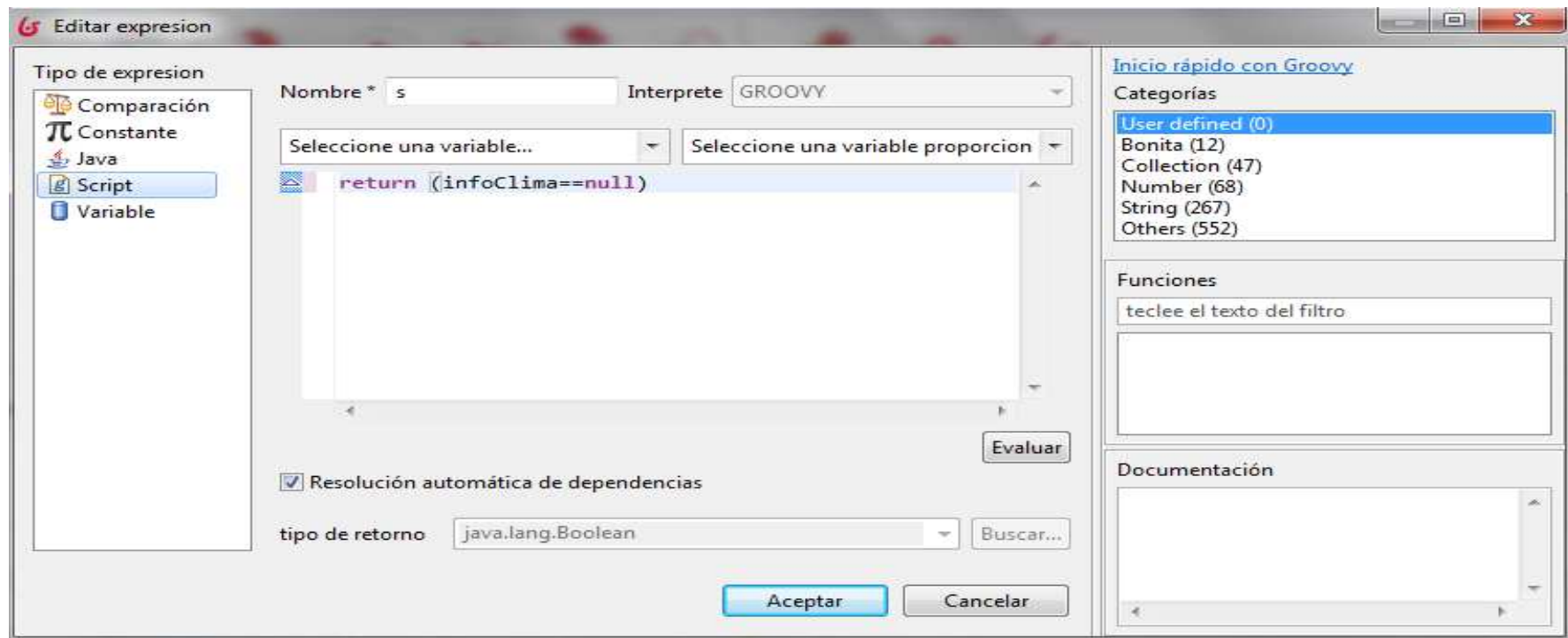
Definición de variables

- Ingresar en la solapa de datos del pool y definir estas tres variables (dos de texto y un Map)



Definición de condiciones

- Seleccionar el flujo que va hacia “Mostrar clima” como flujo por defecto.
- Colocar en el flujo restante la siguiente condición



Formularios

- Crear el siguiente formulario para el pool

The screenshot shows a form builder application with a palette on the left and a form canvas on the right. The palette contains the following elements:

- Casilla de verificac...
- Lista de Checkbox
- Fecha
- Duración
- Contrase...
- Lista
- Grupo Radio
- seleccio...
- Campo con sugerenci...
- Texto
- Área de...

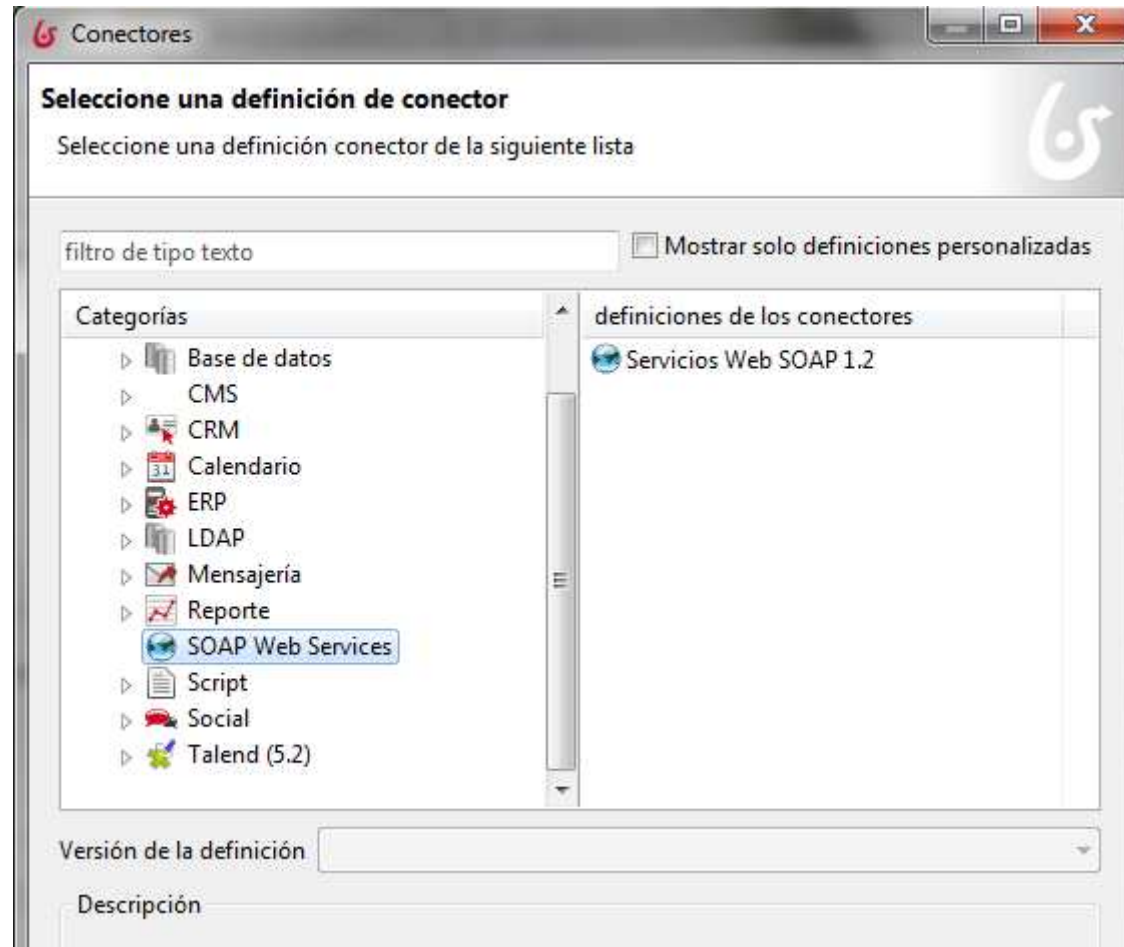
The form canvas displays the following elements:

- A text input field labeled "ciudad".
- A text input field labeled "pais".
- A button labeled "Obtener info cli...".

Agregar la descripción del web service

- En la primera actividad del proceso agregar un conector para realizar la invocación al web service

Conector



Definición del conector


Servicios Web SOAP 1.2 (1.0.0)

General
Especificar la información general

Nombre * GlobalWeather

Descripción

Seleccionar evento *

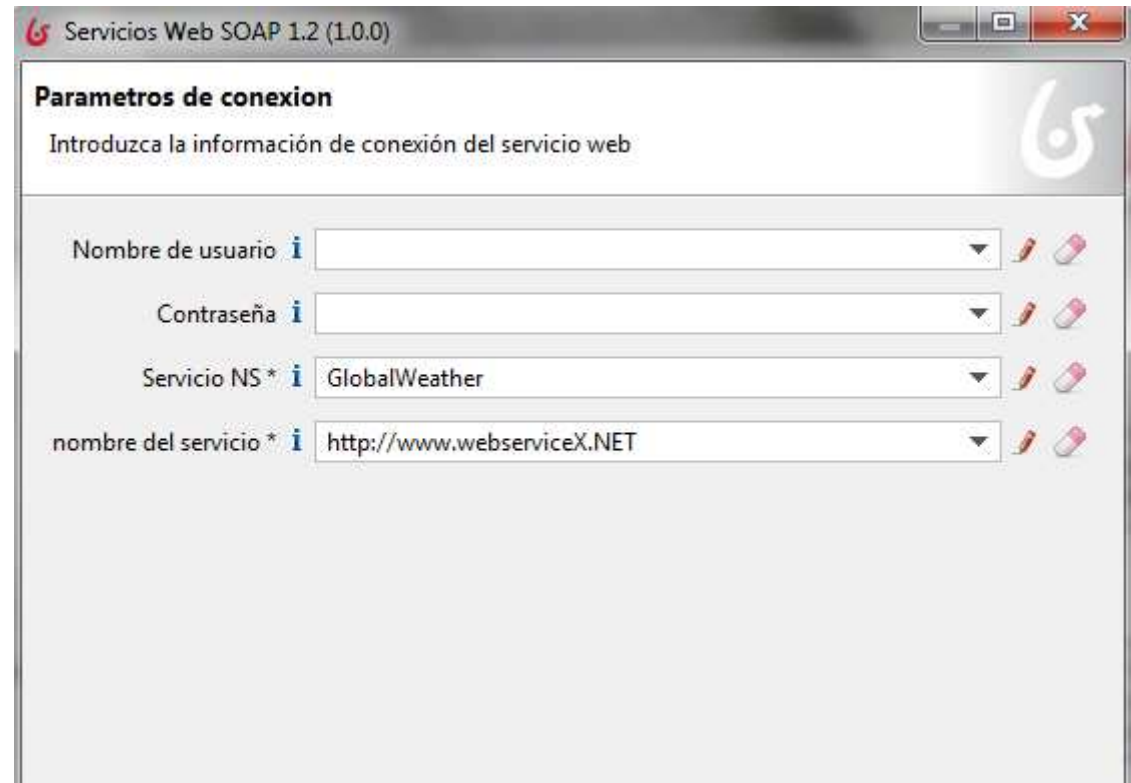


Si falla el conector ... Lanzar error

Error Nombrado

The diagram shows a rectangular connector box. On the left side, there is a small circle with an arrow pointing into it, labeled 'enter'. On the right side, there is a small circle with an arrow pointing out of it, labeled 'finish'.

Definición del conector



The image shows a software window titled "Servicios Web SOAP 1.2 (1.0.0)". Inside, there is a section titled "Parametros de conexion" with the instruction "Introduzca la información de conexión del servicio web". Below this, there are four input fields, each with an information icon (i) and edit/delete icons (pencil and eraser) to its right:


- Nombre de usuario: [Empty text box]
- Contraseña: [Empty text box]
- Servicio NS *: GlobalWeather
- nombre del servicio *: http://www.webserviceX.NET


Definición del conector


Servicios Web SOAP 1.2 (1.0.0)



parámetros de solicitud


Introduzca la información de solicitud de servicio Web

Accion SOAP *i*  

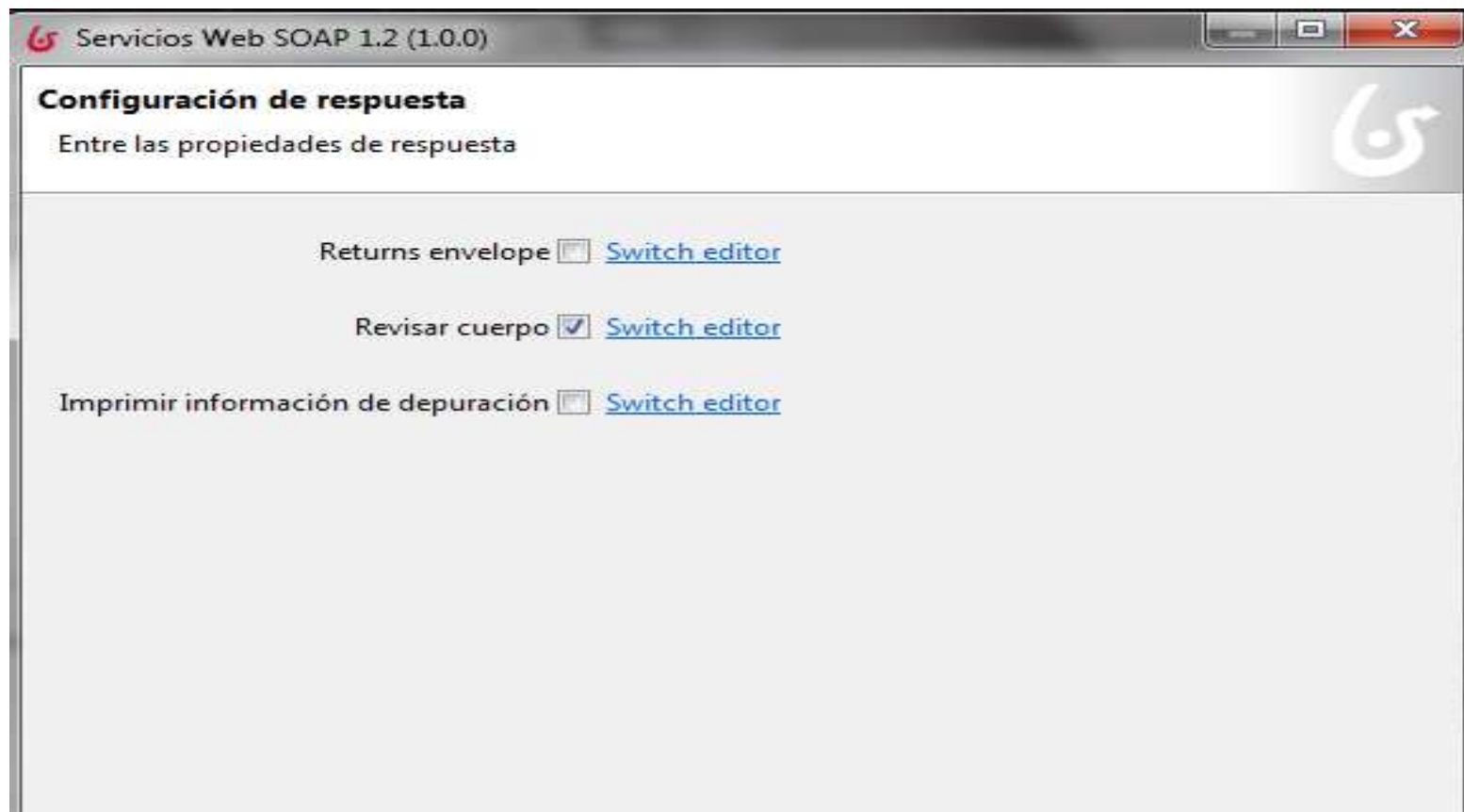
Nombre del puerto * *i*  

Dirección de punto final * *i*  

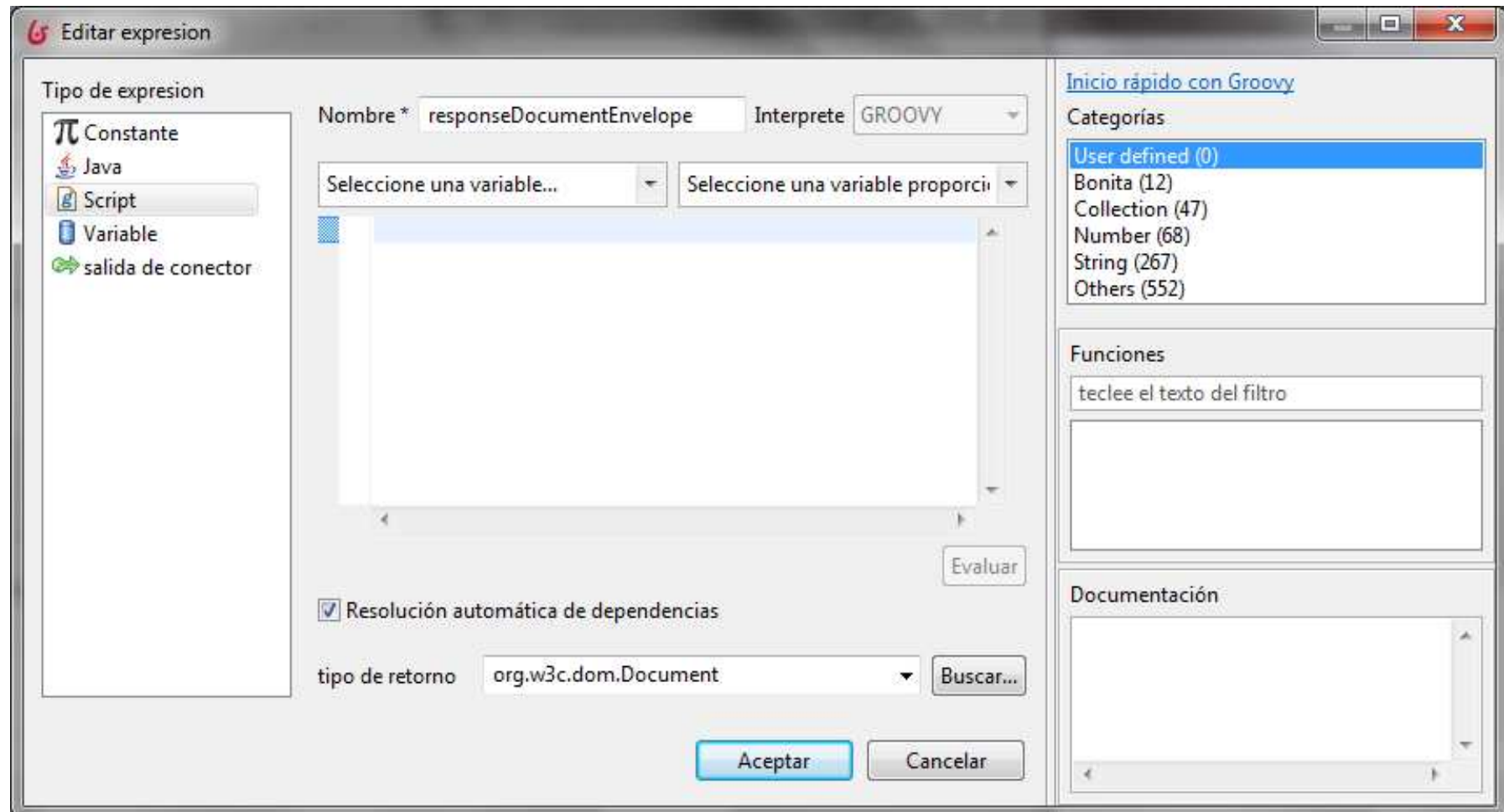
Enlace * *i*  

sobre * *i* 

Definición del conector



Mapeo de las variables de salida



Script para el mapeo

```
■ import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

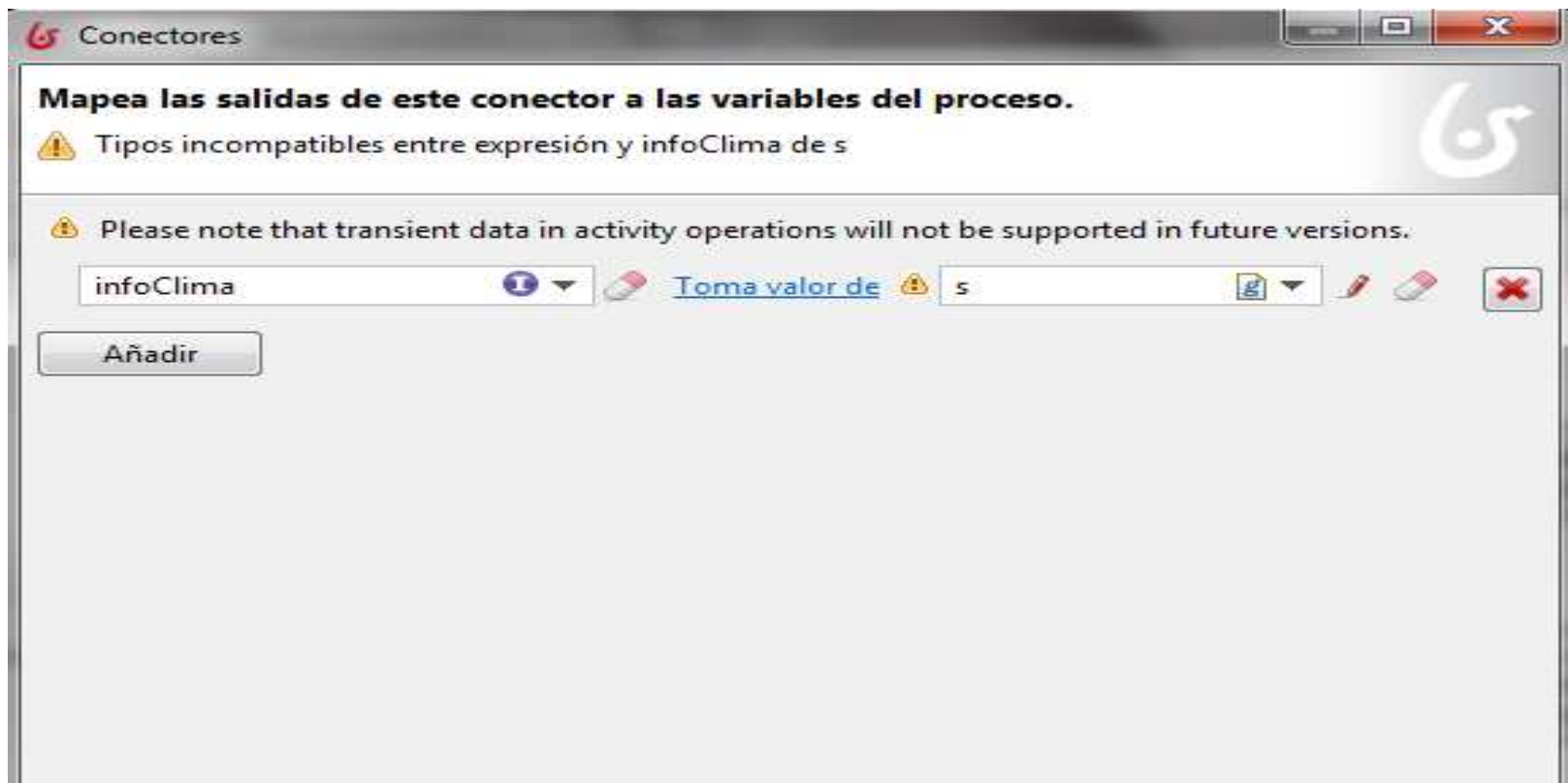
// Clean response xml document
responseDocumentBody.normalizeDocument();
// Get result node
NodeList resultList = responseDocumentBody.getElementsByTagName("GetWeatherResult");
Element resultElement = (Element) resultList.item(0);
String weatherDataAsXML = resultElement.getTextContent();

// Check for empty result
if ("Data Not Found".equalsIgnoreCase(weatherDataAsXML))
    return null;


// Parse embedded XML of result
DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
InputSource inputSource = new InputSource();
inputSource.setCharacterStream(new StringReader(weatherDataAsXML));
Document weatherDataDocument = documentBuilder.parse(inputSource);
Node weatherNode = weatherDataDocument.getDocumentElement();

// Save weather data
Map<String,String> data = new HashMap<String,String>();
NodeList childNodes = weatherNode.getChildNodes();
for (int i=0; i<childNodes.getLength(); i++)
{
    Node node = childNodes.item(i);
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
        String key = node.getNodeName();
        String value = node.getTextContent();
        data.put(key, value);
    }
}
return data;
```

Definición del conector



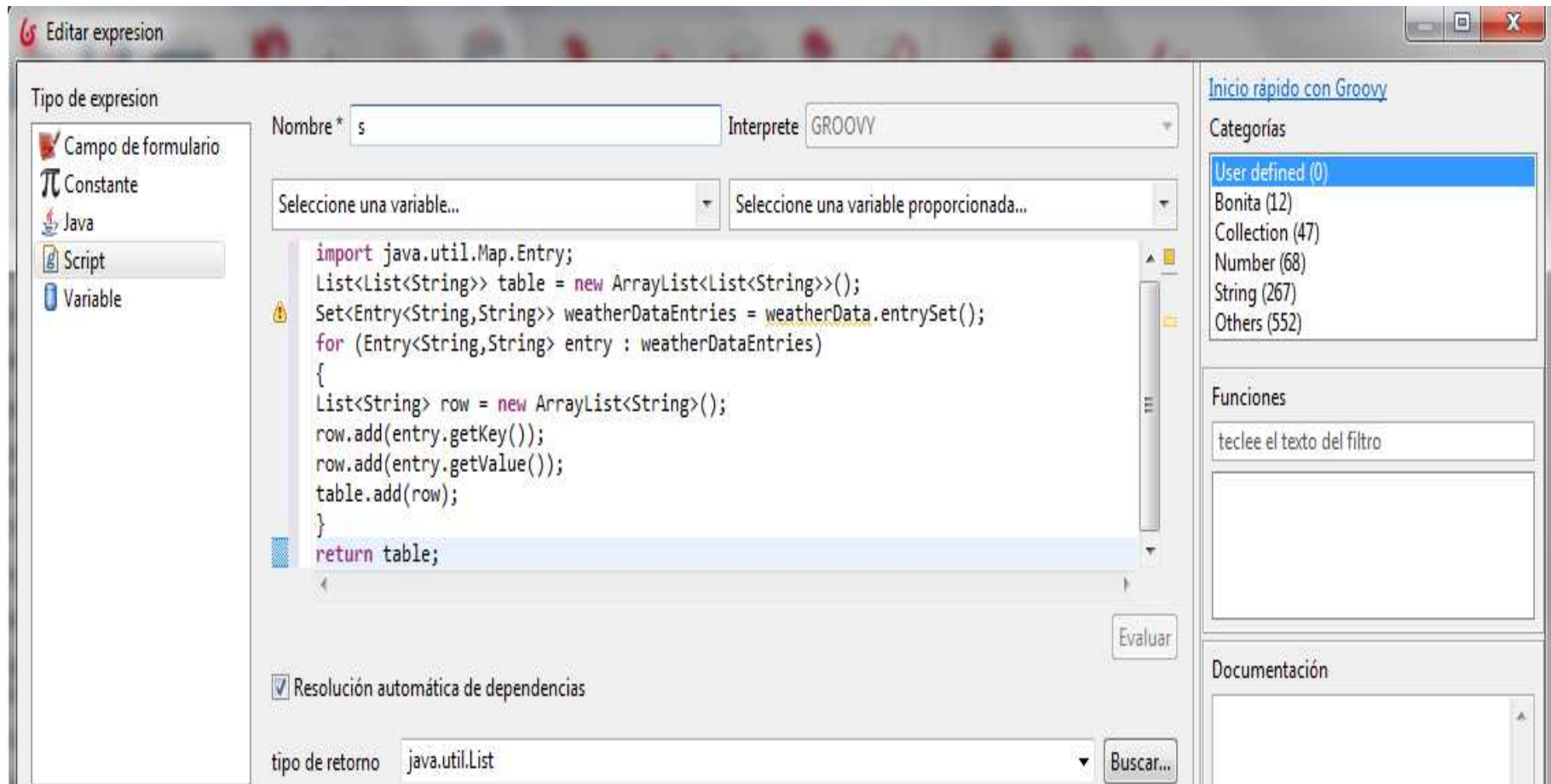
Agregar un formulario en la primera actividad



The screenshot shows a web application window titled "Agregar el formulario...". Inside the window, there is a section titled "Crear un nuevo Formulario" with the subtitle "Crear un Formulario con los datos seleccionados". Below this, there are two input fields: "Nombre" with the value "Obtener info del clima" and "Descripción" which is empty. Underneath these fields, there is a section titled "Agregar widgets basado en..." containing two buttons: "Seleccionar todo" and "Des-seleccionar Todo". Below the buttons, there are two checkboxes, each followed by a label and a description: "☐ ciudad -> Texto" and "☐ pais -> Texto". The "pais" checkbox is currently selected.

Formulario

- Agregar una tabla y en la solapa datos editar la expresión



Formulario

*MiDiagrama (1.0) *EjemploInfoClima *ObtenerInfoDelClima

Tabla1

Cerrar

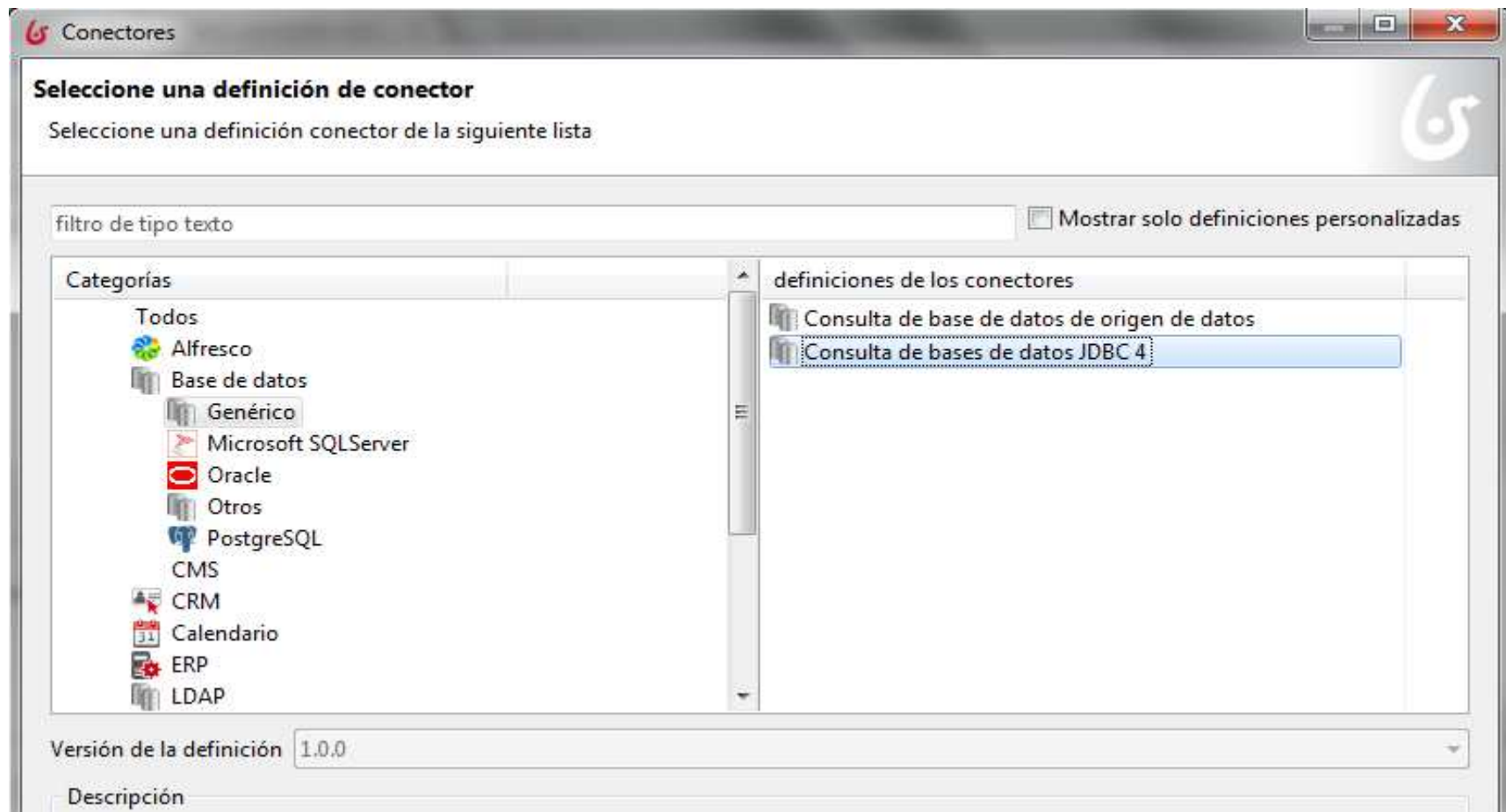
Agregar un formulario en la actividad de “No se encontraron resultados”

The image shows a software development environment. At the top, a large text box labeled 'Mensaje1' is visible. Below it is a button labeled 'Enviar1'. The interface includes a toolbar with options like 'Vista global', 'General', 'Apariencia', 'Estado de validación', and 'Preview'. A sidebar on the right shows a tree view with 'Mensaje1' selected, and a 'Datos Descripción' section with a message: 'Perdón, no se encontraron resultados'.

Ejecución

- Luego al ejecutar, se ingresa la ciudad y el país, y se invoca el web service. Luego se visualizan los resultados obtenidos desde la invocación.
-

Configuración de conectores a la base de datos



Pasos para la configuración

- Seleccionar el pool o la tarea en el proceso
 - Seleccionar agregar nuevo conector
 - Seleccionar la definición de la base de datos, por ej JDBC 4 database query. Clicar siguiente.
 - Ingresar el nombre, descripción, evento y **mensaje de error** (si hubiera). Clicar siguiente.
 - Seleccionar el .jar de conexión a la base de datos.
 - Ingresar la información de acceso. Clicar siguiente.
 - Ingresar la consulta. La misma se envía a la base de datos. Puede ser una única consulta o puede contener separadores.
 - Especificar las operaciones de output, es decir cómo se visualizan los datos. Puede ser mediante HTML o expresión.
 - Seleccionar la variable sobre la que se guardan los datos.
 - Click en Finalizar.
-

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

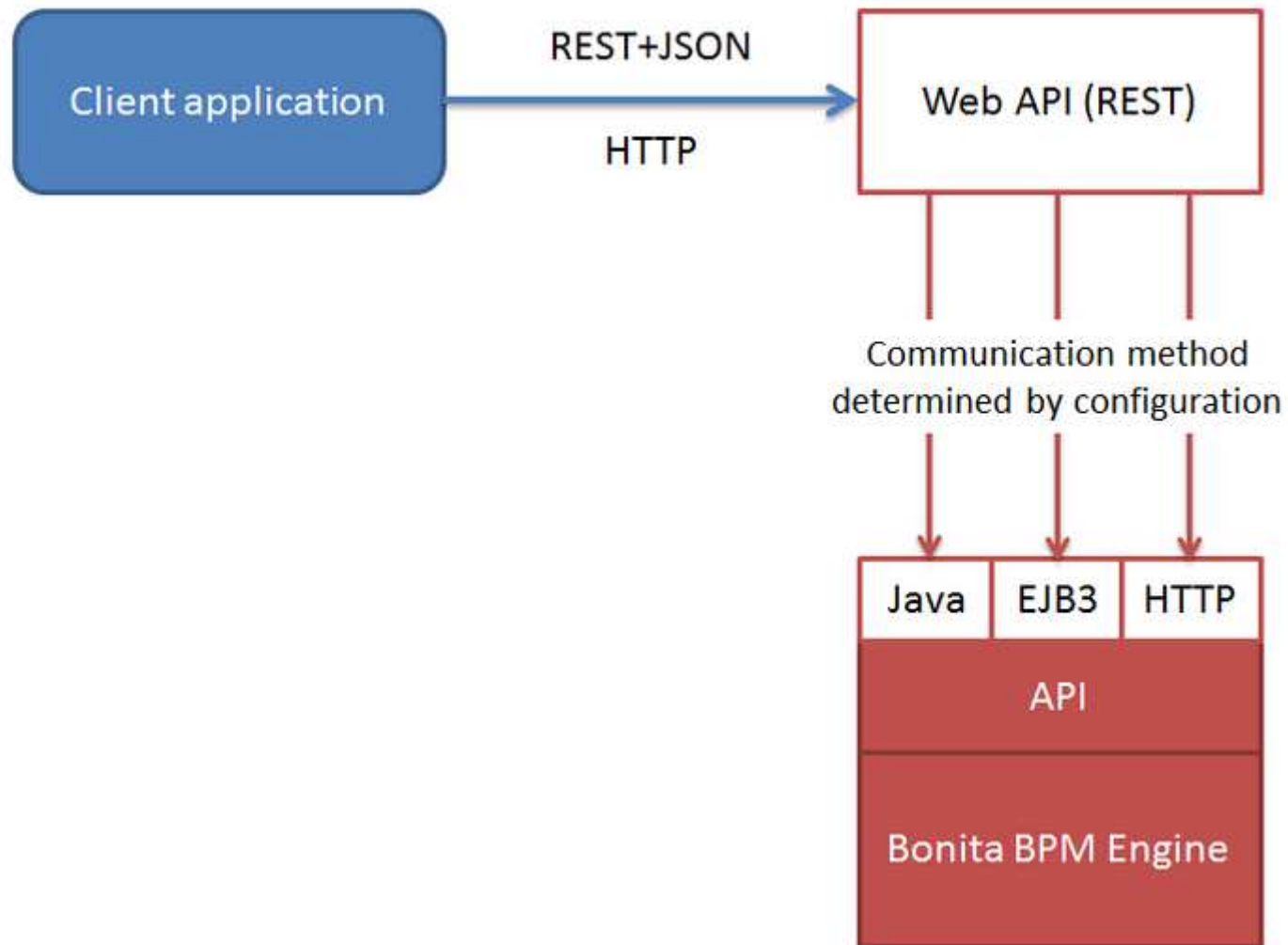
- **@Override**
public void validateInputParameters() **throws** ConnectorValidationException {
 final List<String> messages = **new** ArrayList<String>(0);
 if (url == **null** || url.isEmpty()) {
 messages.add("Url can't be empty");
 }
 if (driver == **null** || driver.isEmpty()) {
 messages.add("Driver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

Conector a clase java

- Este mecanismo se puede utilizar por ejemplo, para desarrollar una clase que se conecte a otro servidor utilizando una API Rest, en reemplazo del mecanismo estándar SOAP que provee Bonita por defecto.
-

Bonita API Rest



Ejemplos de uso de la API

- Actualización de variables
 - Request url
[http://../API/bpm/caseVariable/\[caseId\]/\[variableName\]](http://../API/bpm/caseVariable/[caseId]/[variableName])
 - Request method GET
 - Request payload empty
 - Response payload A case variable representation
-

Ejemplo de obtención de una variable

- Request url GET
|/API/bpm/caseVariable/1/myInvoiceAmount
 - Request payload empty
 - Response payload
 - { "description": "",
"name": "myInvoiceAmount",
"value": "14.2",
"case_id": "1",
"type": "java.lang.Float" }
-

Ejemplo de actualización de una variable

- Request urlGET
|/API/bpm/caseVariable/1/myInvoiceAmount

- Request payload
{ "type": "java.lang.Double",
"value": 22.98 }

Response payload

N/A
