Tabla de contenido

[1. Necesidad 2](#_Toc403637931)

[2. Requisitos 2](#_Toc403637932)

[a. Solución sencilla de integrar en el diseño de procesos 2](#_Toc403637933)

[b. Sincronización de información de negocio 2](#_Toc403637934)

[c. Validar la terminación de las operaciones con sistemas externos necesarias para concluir una tarea. 3](#_Toc403637935)

[d. Asegurar la transaccionalidad en la ejecución de las operaciones asociadas una tarea. 3](#_Toc403637936)

[3. Propuesta de solución: 4](#_Toc403637937)

[a. La solución debe ser sencilla de integrar en el diseño de los procesos 4](#_Toc403637938)

[b. Sincronización de información de negocio 9](#_Toc403637939)

[c. Validar la terminación de las operaciones con sistemas externos necesarias para concluir una tarea. 11](#_Toc403637940)

[d. Asegurar la transaccionalidad en la ejecución de las operaciones asociadas una tarea. 12](#_Toc403637941)

# Necesidad

Consolidar los datos de negocio asociados a los expedientes en un modelo de datos externo al motor BPM. Queremos de este modo poder explotar toda la información de los expedientes generando informes, automatizando tareas, etc.

Además debemos poder asegurar la integridad de los datos de negocio vinculados a los expedientes en fuentes de datos externas al motor BPM.

# Requisitos

La solución debe **asegurar**

* Solución sencilla de integrar en el diseño de procesos.
* la sincronización entre la información de negocio guardada en el motor de BPM y el modelo negocio externo.
* Validar la terminación de las operaciones con sistemas externos necesarias para concluir una tarea.
* Asegurar la transaccionalidad en la ejecución de las operaciones asociadas una tarea.

## Solución sencilla de integrar en el diseño de procesos

La solución se debe implementar dentro del diseño de procesos BPM:

* Fácil de integrar dentro de un proceso BPM.
* Permitir la conexión con sistemas externos.
* Altamente desacoplado de los sistemas externos.

## Sincronización de información de negocio

Se debe generar un mecanismo que asegure la sincronización de los datos del expediente dentro del motor BPM con respecto al modelo de negocio externo.

Durante la ejecución de una tarea se pueden consultar, modificar y crear datos en sistemas externos. Por ejemplo crear o modificar la información de un tercero.

Hay que **asegurar** por tanto que los **datos** que se creen durante la ejecución de la tarea, sean **coherentes en el sistema independientemente de la finalización o no de la tarea**.

Si la tarea no finalizó y se retoma debería comprobarse si existen estos datos y **retomar el estado anterior de la tarea.**

***El diseño debe asegurar que no se abuse de las subtareas necesarias dentro de una tarea, de modo que se aseguren que las operaciones con sistemas externos sean atómicas y estén aisladas.***

## Validar la terminación de las operaciones con sistemas externos necesarias para concluir una tarea.

Durante la ejecución de una tarea se pueden generar documentos, enviar correos, publicar información en sistemas externos, etc.

Hay que asegurar por tanto que **operaciones que se realizan durante la ejecución de la tarea, sean coherentes en el sistema independientemente de la finalización o no de la tarea**.

Si la tarea no finalizó y se retoma debería comprobarse si se han realizado estas operaciones previamente y **retomar el estado anterior de la tarea**, no volviendo a generarse y ensuciando el sistema.

El diseño del proceso BPM, debe asegurar que no se abuse de incorporar muchas operaciones como necesarias dentro de una tarea, de modo que se intente que las operaciones con sistemas externos sean atómicas y estén aisladas en subtareas.

## Asegurar la transaccionalidad en la ejecución de las operaciones asociadas una tarea.

Hay que controlar que las operaciones que se realicen durante el proceso de finalización de una tarea sean transaccionales. De modo que si la tarea termina, se asegure la consolidación de estas operaciones y si no se termina correctamente la tarea se realice un rollback de todas las operaciones vinculadas con el cierre de la tarea.

# Propuesta de solución:

## La solución debe ser sencilla de integrar en el diseño de los procesos

Normalmente un motor BPM tiene que ser la ayuda para realizar procesos en una organización. Si se convierte en el centro del sistema tendremos un problema de tener una dependencia demasiado grande a un único elemento arquitectónico. Un flujo BPM nos tiene que ayudar a organizar las tareas de una parte de un proceso o del proceso completo pero la organización tiene que tener en control del estado y de los datos en entidades externas.

En las grandes organizaciones se mezclan habitualmente procesos online y batch. Gran parte de esos procesos batch pueden estar ya construidos, incluso en tecnología clásicas como Natural, Cobol, RPG o procedimientos almacenados por lo que es importante valorar como se van a integrar esos dos mundos.

Usando un flujo y motor BPM, tenemos varios modos de hacerlo, cada uno mejor que el anterior:

* Los datos pueden guardarse en variables transitorias almacenadas en el proceso BPM mientras está activo en el motor, cosa que crea mucha dependencia con él.
* El flujo BPM puede guardar datos directamente en una base de datos, con Scripts o mapeados visuales, (dependiendo de las capacidades de la herramienta) para que estén muchos datos en el proceso y los deseados en un modelo de datos concretos en una base de datos tradicional. Cosa que va a acoplar mucho el flujo y la base de datos. Más aun considerando que muchas organizaciones están deseando cambiar los modelos de datos obsoletos.
* El flujo BPM puede informar a servicios web (directamente u orquestados a través de un ESB) que es el que se encargue de almacenar los datos. Hay más capas por medio pero la responsabilidad del BPM queda más acotada al flujo visual y menos acoplado a la persistencia.

Si el proceso BPM informa al sistema global de la organización, a través de una capa de servicios, sobre los cambios importantes en el proceso (recordad un patrón de diseño llamado Memento) estaremos haciendo un uso muy equilibrado de las tecnologías (siempre esto valorado en un contexto).

Soluciones tecnológicas que aporta Bonita

* **Conectores** 🡪 Solución del BPM, para asegurar la conexión con sistemas externos desde bonita. En la versión comunity tenemos que descartar esta solución, a día de hoy puede dejarnos un expediente en un estado irrecuperable. No están accesibles desde cualquier punto de un formulario, funcionalidad que se ofrece en las otras versiones de Bonita.
* **Transient data y Business Data Model (BDM) 🡪** En las versiones de pago únicamente. Se trata de objetos que se crean temporalmente durante la vida de una tarea y el uso de hibernate o scripts groovy atacando a una base de datos con el negocio asociada al proceso.

Soluciones tecnológicas a medida.

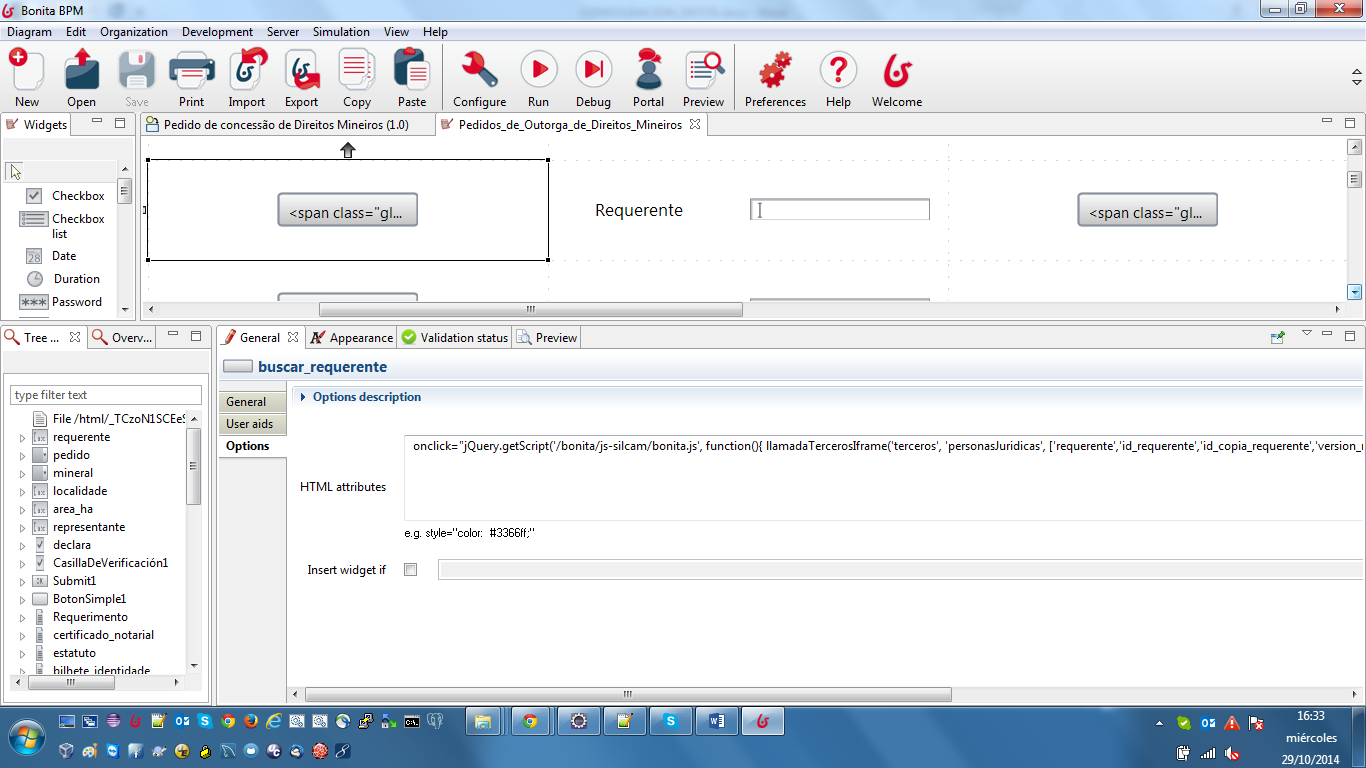
* **Diálogos** 🡪 Actualmente ya conectamos con servicios externos a través de diálogos que integramos dentro de una ejecución de bonita. Estos diálogos se pueden llamar en cualquier momento durante la visualización de un formulario. Ejemplo de llamada a terceros.
* **Servicios Rest** 🡪 Actualmente ya usamos servicios rest para obtener información de servicios externos. Se pueden usar también para ejecutar operaciones o modificar información en servicios externos. Actualmente los podemos usar en la carga de cualquier campo de un formulario **(Groovy),** en la modificación de cualquier campo **(jquery+rest+ajax).** También se podrían ejecutar al realizar cualquier acción sobre componentes de un formulario, click en un botón etc. Se pueden lanzar también una vez finalizada una tarea en el formulario de confirmación de la realización de la tarea.
* **Core que exponga servicios rest a bonita BPM-SILCAM** 🡪 Este core será una capa de servicios especializada sobre procesos de BPM de bonita para SILCAM. Este core usara las operaciones expuestas por el core de SILCAM pero será responsable de asegurar en paralelo el sincronismo y transaccionalidad de las operaciones y la información de los expedientes con el negocio.

El uso de diálogos y servicios REST a medida, aseguran que es fácil de integrar en bonita, permiten la conexión con sistemas externos altamente desacoplados.

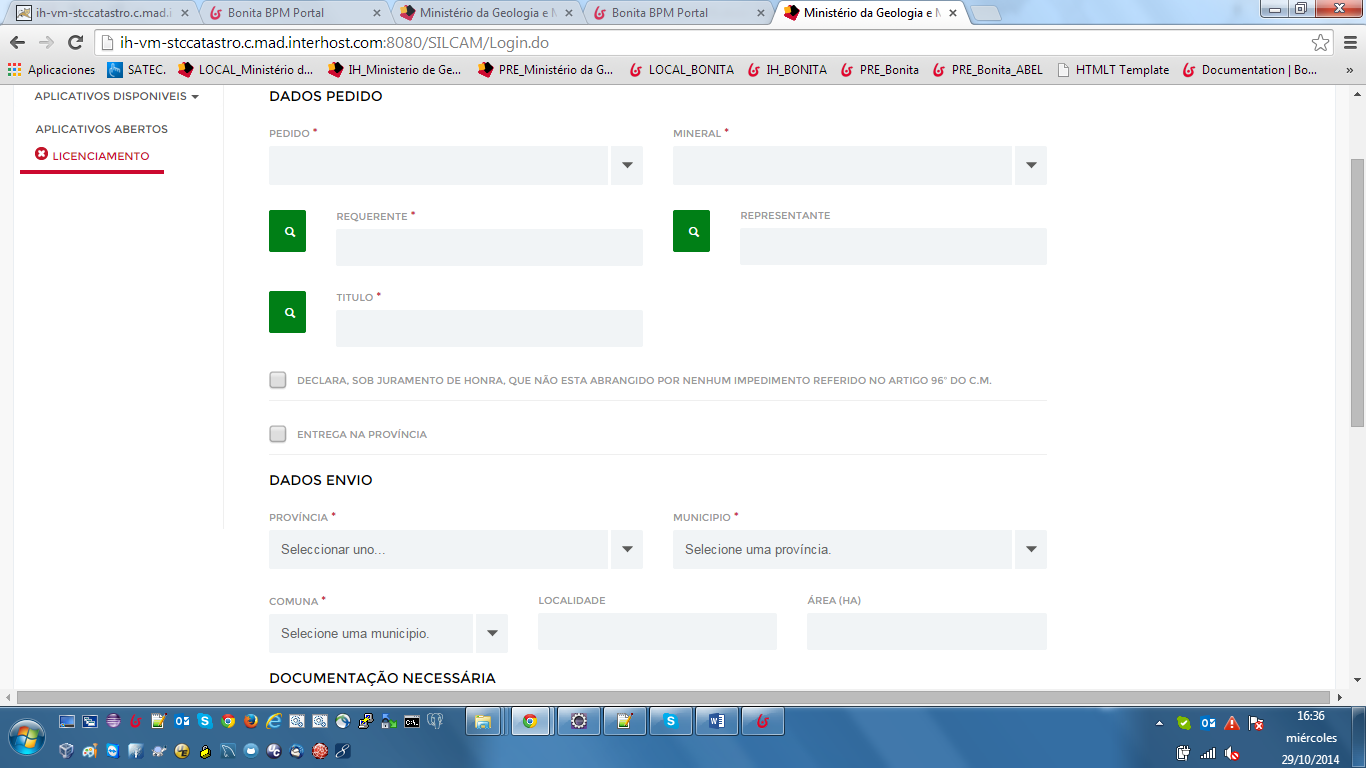
El disponer de un core especializado para bonita-SILCAM que exponga estos servicios nos aporta seguridad y control sobre el BPM.

Ejemplo de dialogo:

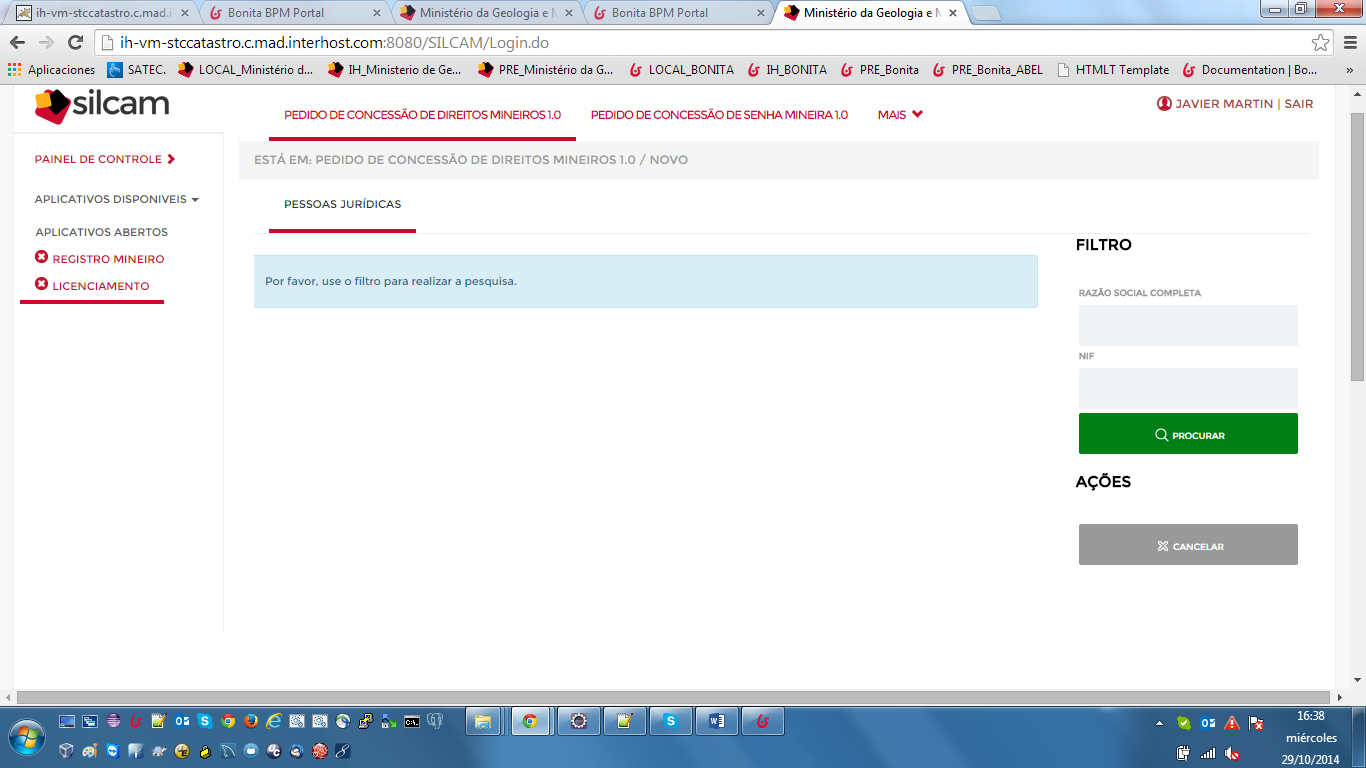
Diseño:



Uso:

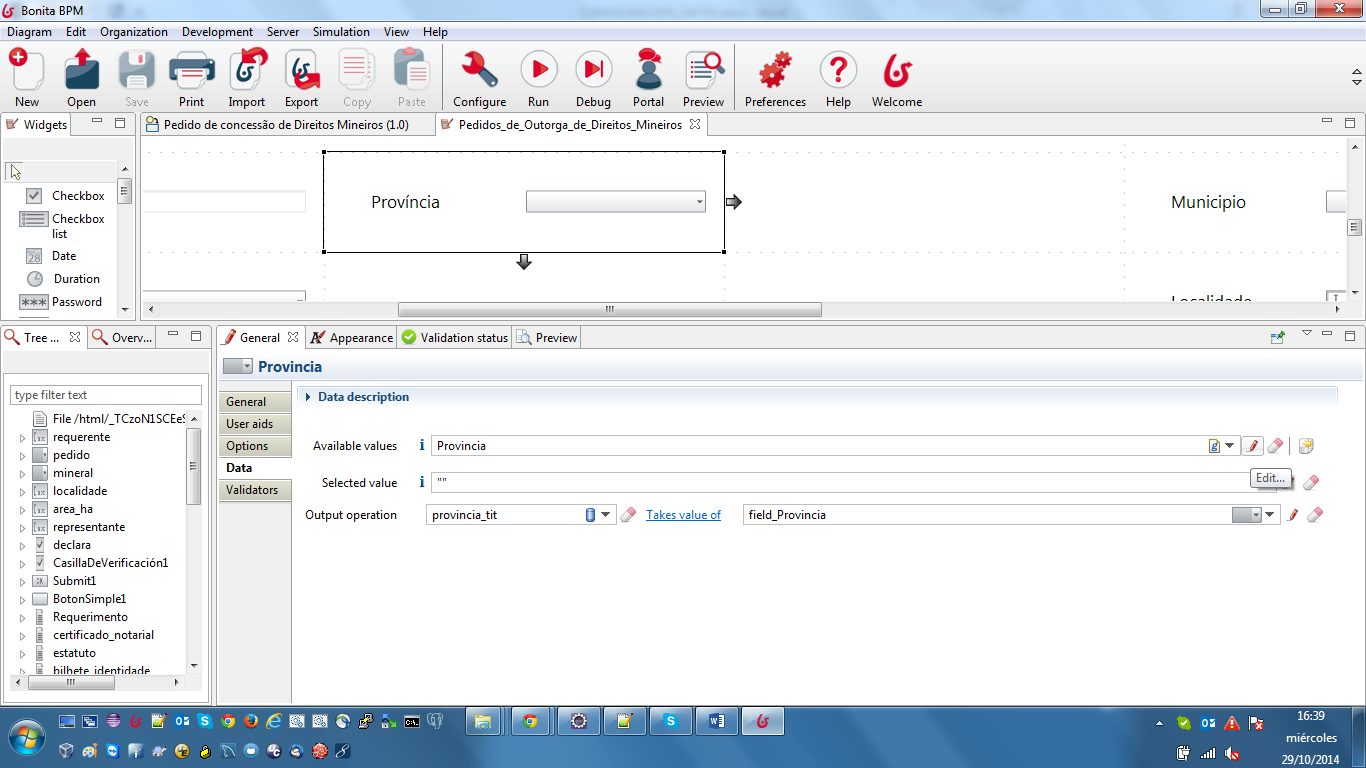


Al hacer click en la lupa se abre un dialogo contra terceros:

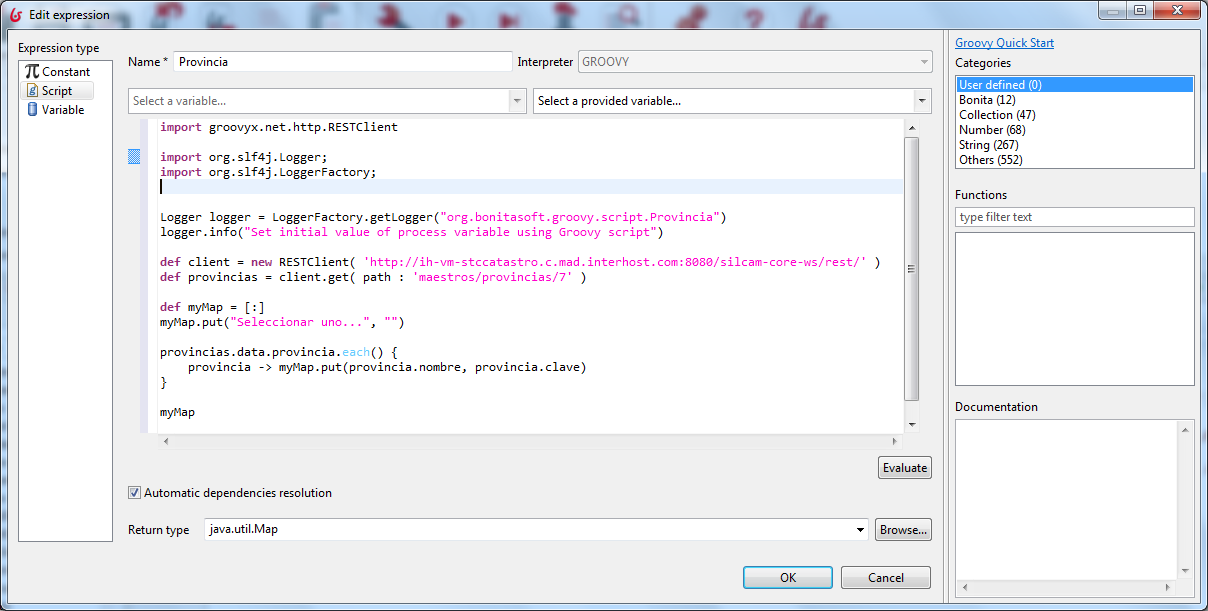


Ejemplo de servicio Rest:

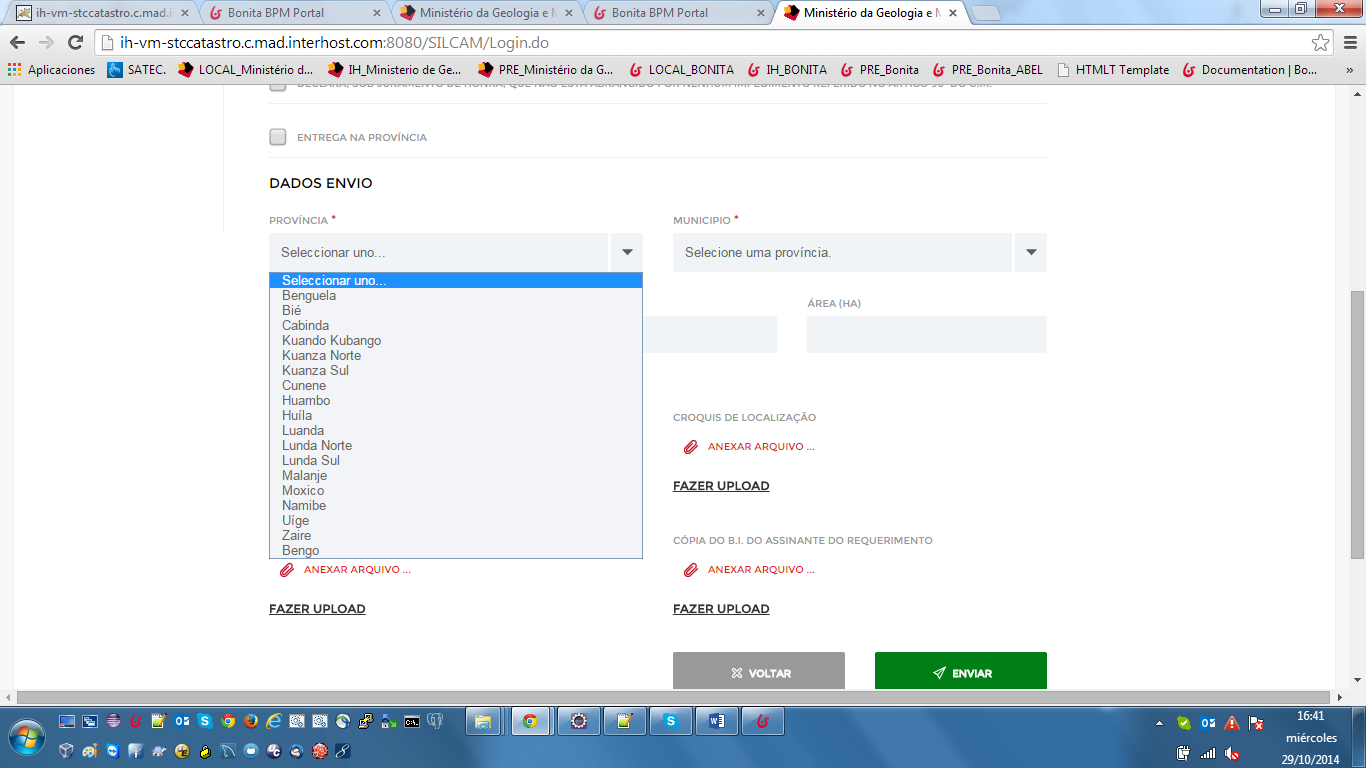
Diseño:



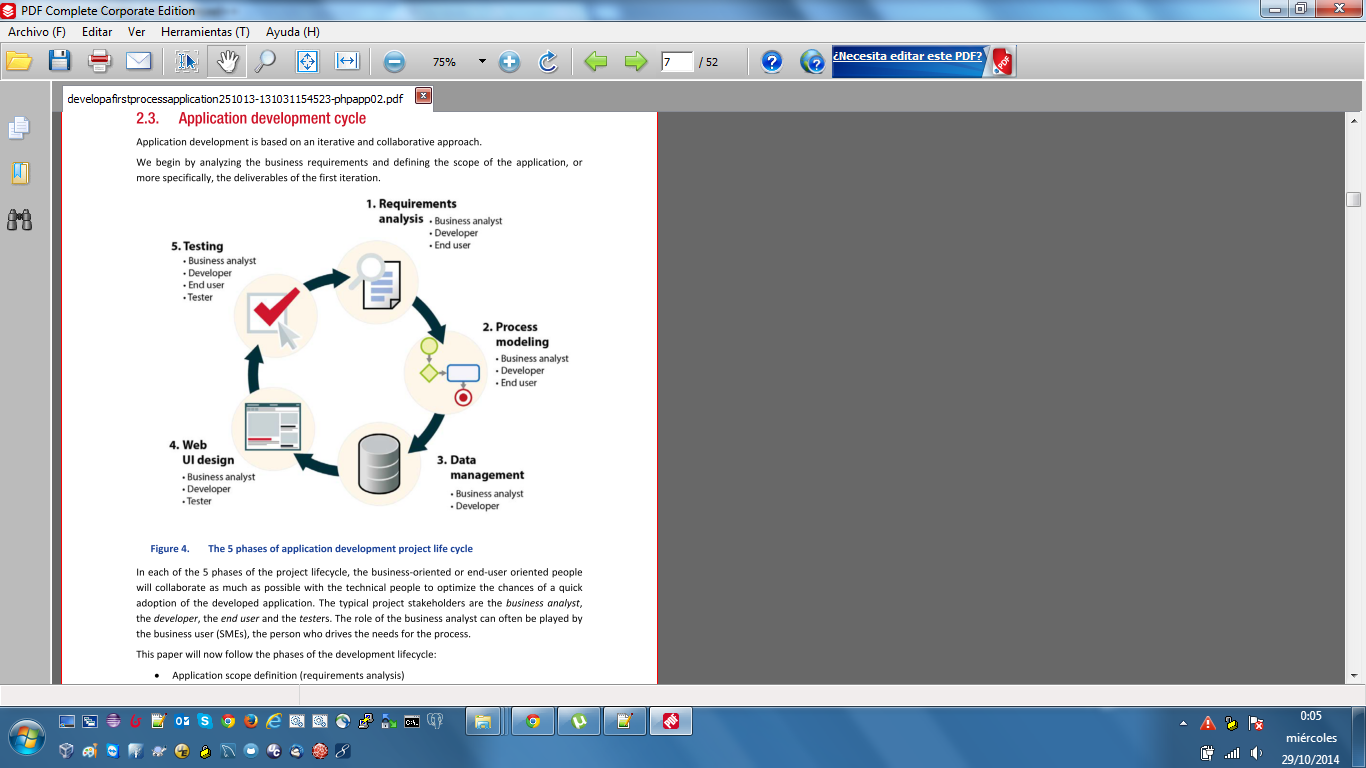
Editar Valores disponibles



Uso



El uso de estas tecnologías implica perfiles muy específicos que deberán cooperar en el desarrollo del proceso BPM.



## Sincronización de información de negocio

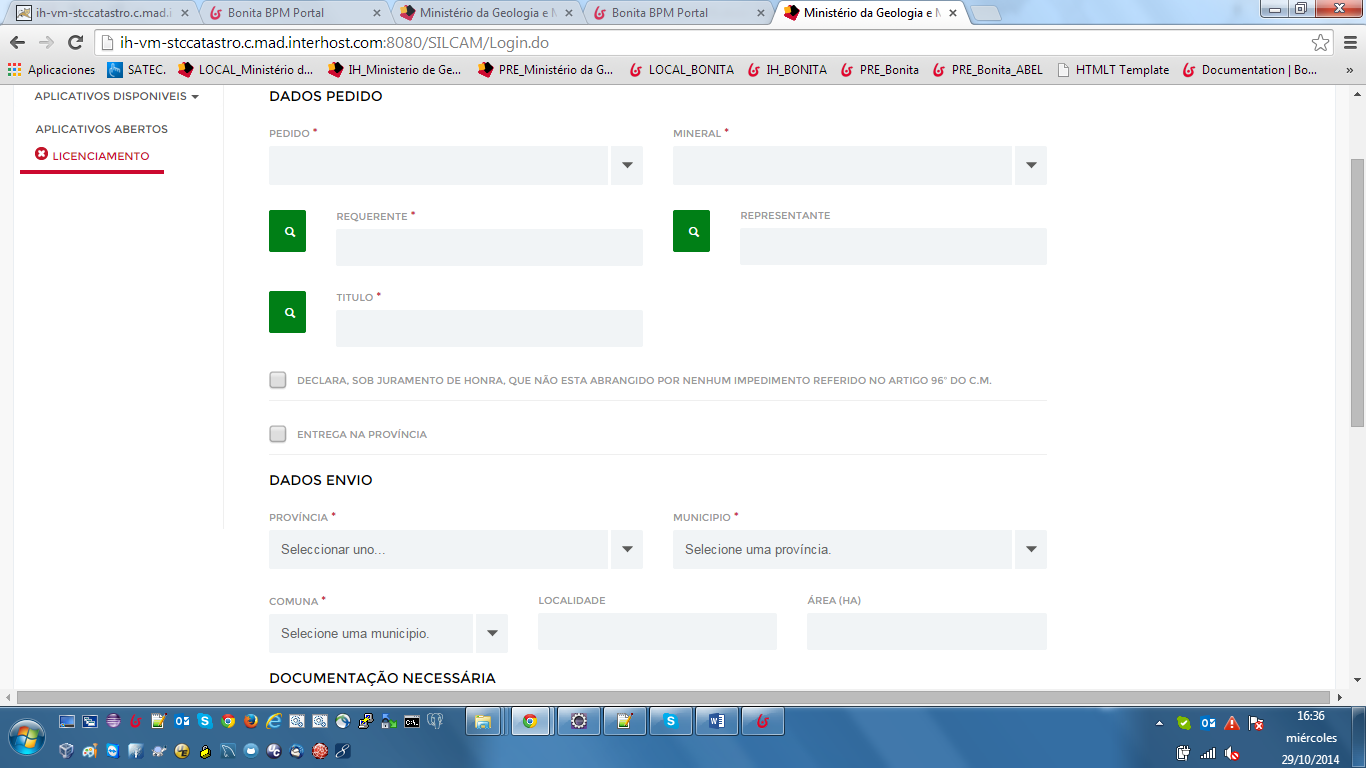
La solución tecnológica de bonita (**conectores,** **Transient data y Business Data Model (BDM)**) para este requerimiento no es viable en la versión comunity.

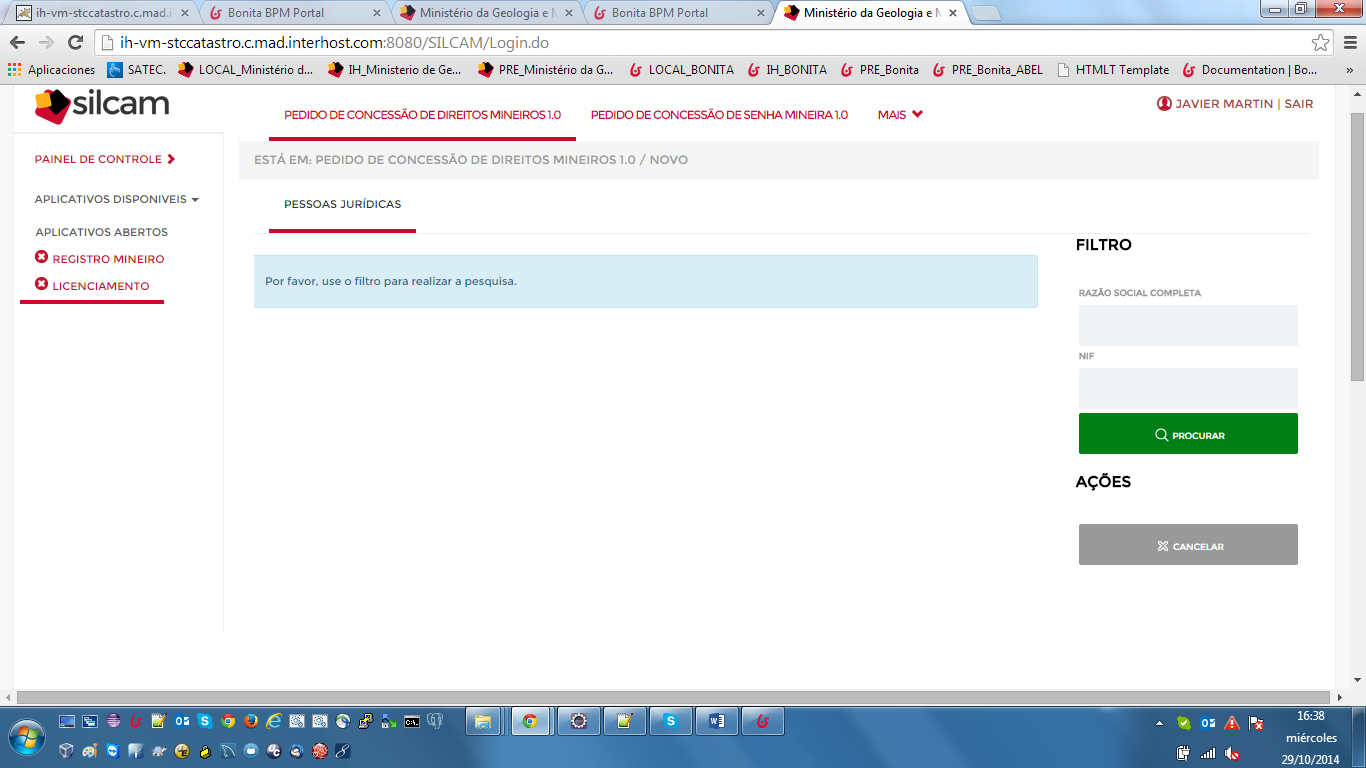
A través de los **servicios rest y de los diálogos personalizados**, se puede manipular la información en los servicios externos y **crear un mecanismo de sincronismo de datos entre el expediente y el negocio.**

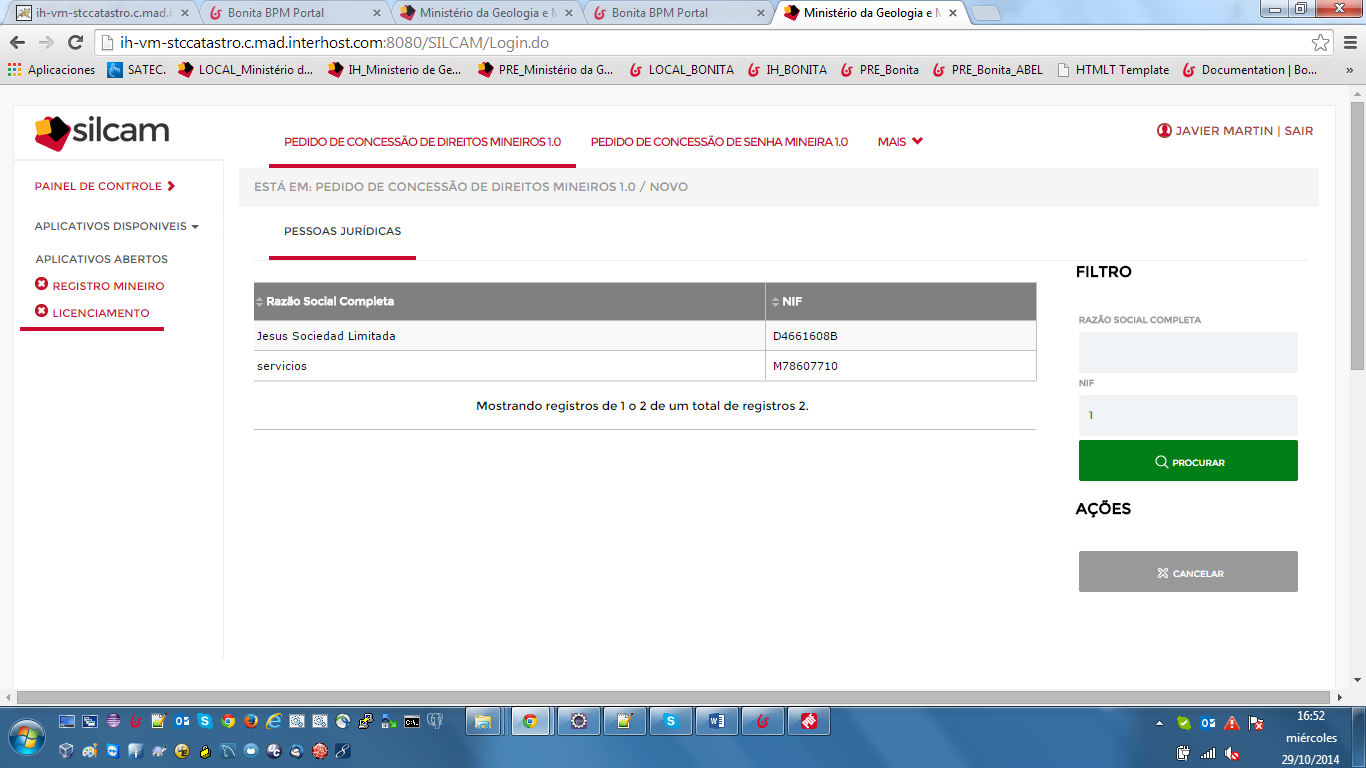
Llamar al servicio externo y después de devolver la información obtenida al formulario de bonita, **se puede persistir la información obtenida directamente en el propio motor de bonita.**

De este modo podemos mantener una coherencia y una sincronización de datos durante la ejecución de una tarea. Esta solución tiene beneficios añadidos como poder consultar información de tareas de expediente en curso.

Ejemplo queremos elegir un requerente en el servicio externo “Registro Minero”, se abre un dialogo con el “registro minero”. En este dialogo se selecciona o se crea el requerente si no existe, se genera una copia, etc







El dialogo devuelve los datos a la tarea, y desde la tarea se pueden guardar estos datos directamente en el motor de bonita, dando a un botón guardar, lanzando un evento, etc.

Si la tarea no se termina y se tiene que retomar, con esta tecnología se podría recuperar el estado del formulario de bonita, al retomar la tarea los datos se cargan directamente desde el motor de bonita, asegurando que se carga el formulario con los datos del estado en el que se quedó.

Problemas que implica esta solución:

* Si hemos generado información en el servicio externo y al devolver esa información a la tarea de bonita, falla por alguna razón. Quedarán datos creados en el servicio externo que no se contemplan en el proceso de bonita. Es responsabilidad del servicio externo gestionar este problema.
* Añadir código de llamadas a servicios (en este caso al propio motor de bonita) durante la ejecución de la tarea para así poder recuperar el estado anterior de la tarea.
* En el caso que el formulario sea una instancia de proceso, no se pueden guardar datos de forma transitoria en el expediente pues aún no se ha instanciado, esto hay que resolverlo con un diseño BPM coherente a esta solución.

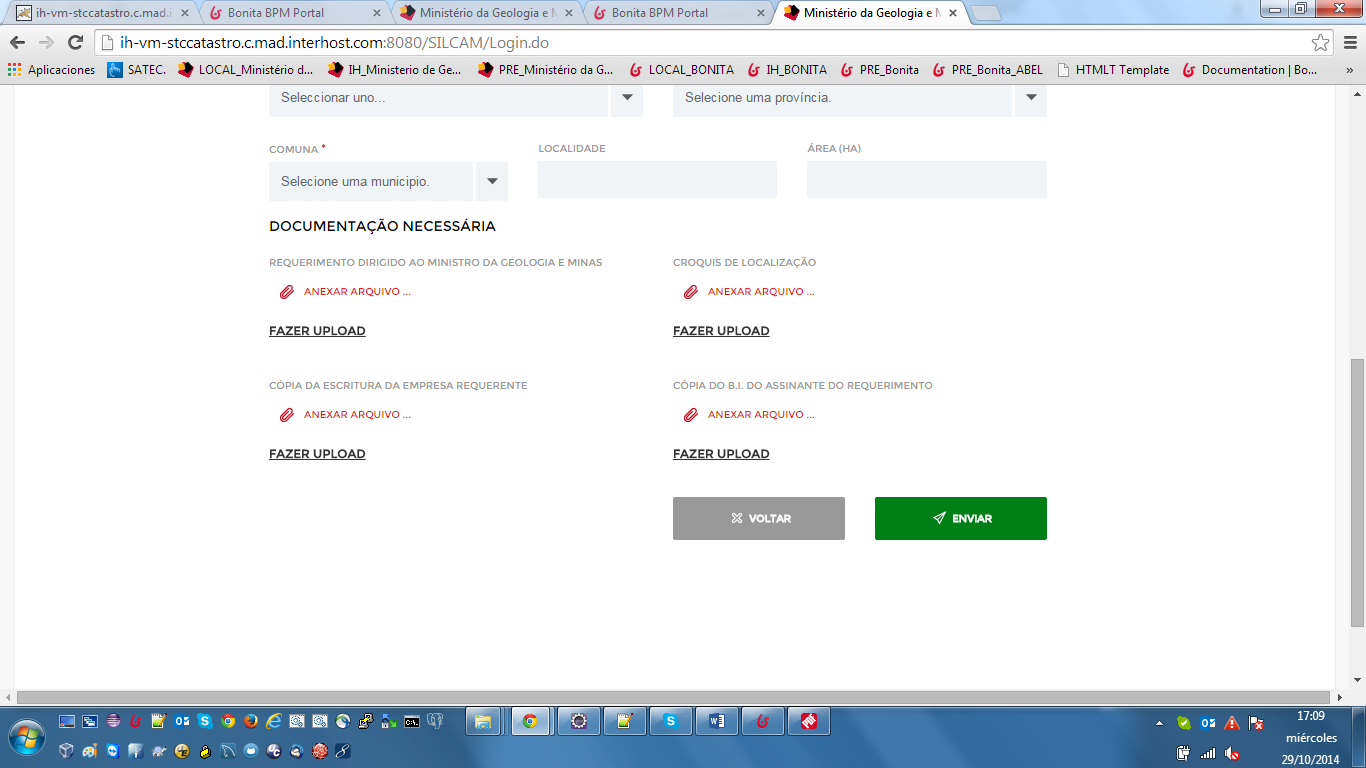
## Validar la terminación de las operaciones con sistemas externos necesarias para concluir una tarea.

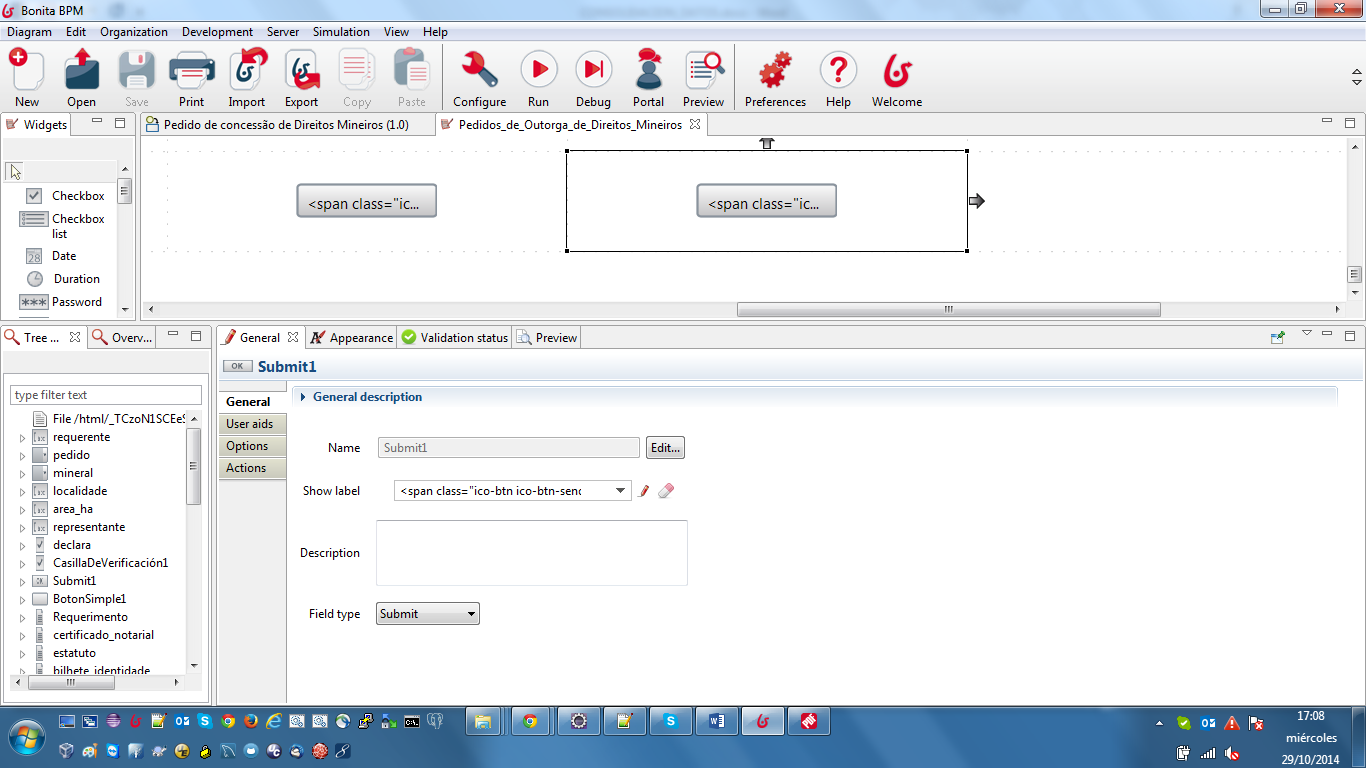
Si la tarea se intenta terminar se puede revisar a través de servicios externos que se han realizado las operaciones necesarias, actualizar el modelo de negocio externo y a continuación ordenar terminar la tarea.

Esta solución es fácil de implementar. Los formularios de bonita, se puede validar a nivel de campo o del formulario entero. En esta opción de validar el formulario entero se pueden lanzar servicios rest a través de groovy que comprueben que se han realizado las operaciones necesarias dentro de la tarea y actualicen el modelo de negocio externo.

Si no se han realizado todas las tareas necesarias lo indicará por pantalla y evitará la finalización de la tarea.

Si se pasa la validación viendo que se han realizado todas las tareas necesarias se terminará la tarea. De este modo no tenemos que preocuparnos por la transaccionalidad en la ejecución de la tarea.





De este modo si la tarea no finaliza por cualquier motivo, tenemos un estado transitorio asociado a la tarea, de modo que cuando la recuperemos se recuperará con el estado en que se quedó antes de la intención de finalizar la tarea.

## Asegurar la transaccionalidad en la ejecución de las operaciones asociadas una tarea.

Con la solución expuesta no contemplamos la necesidad de controlar la transaccionalidad, pues al asegurar que podemos guardar el estado de la tarea y responsabilizar a los servicios externos del control de la información generada durante la ejecución de la tarea, Bonita queda exenta así de controlar la transaccionalidad.