Diseño SOA Silenus

Silenus Consultoría

SOA Silenus

SOA/09009

Junio de 2009

Índice

1 Introducción

La finalidad de este documento es especificar la arquitectura de la solución que se implementará para la Prueba de Concepto. A partir de esta especificación, el equipo de desarrollo implementará la solución escogida.

Para implementar la solución mediante una arquitectura SOA con empleo de un ESB y la integración de un motor BPM se requiere:

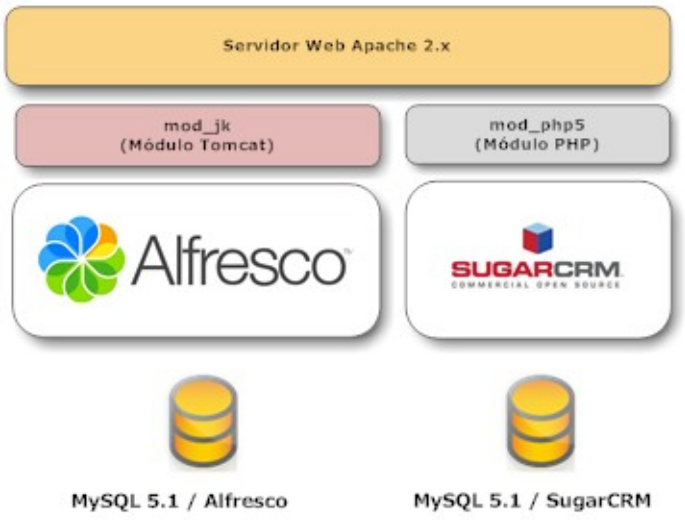
- ✔ Definir la arquitectura de la solución
- ✔ Definir los servicios y puntos de comunicación en cada una de las aplicaciones a integrar. En nuestro caso Alfresco y SugarCRM.
- ✔ Definir los canales de comunicación en el bus ESB.
- ✔ Definir los procesos BPM / BPEL necesarios para implementar los Flujos de Trabajo.

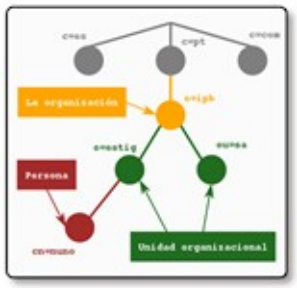
2 Arquitectura

El despliegue de las aplicaciones de código abierto del proyecto exige la instalación de:

- Un servidor web Apache o similar, capaz de ejecutar aplicaciones PHP o de comunicarse mediante FastCGI con un motor de PHP.
- Un motor de PHP.
- Mysql como Motor de Base de Datos relacional (impuesto por SugarCRM).
- SugarCRM desplegado sobre el servidor web y la base de datos.
- Un servidor de aplicaciones J2EE como Apache Tomcat.
- Alfresco desplegado sobre Apache Tomcat y Mysql.
- Un ESB (ServiceMix o Mule) desplegado de forma independiente del servidor de aplicaciones, junto con los servicios necesarios como el servidor JMS ActiveMq.
- Un motor BPM / BPEL. Si se trata de un BPM se puede desplegar integrado con el ESB o bien de forma independiente. Si se trata de BPEL puede desplegarse como una aplicación web J2EE dentro de Apache Tomcat o si soporta JBI y el ESB también, como un componente JBI. Los motores BPM / BPEL que pueden emplearse por su idoneidad para el proyecto son Jboss BPM (con o sin módulo BPEL) o Apache ODE.
- Servidor LDAP como OpenLDAP, empleado como backend de usuarios para SugarCRM y para Alfresco.

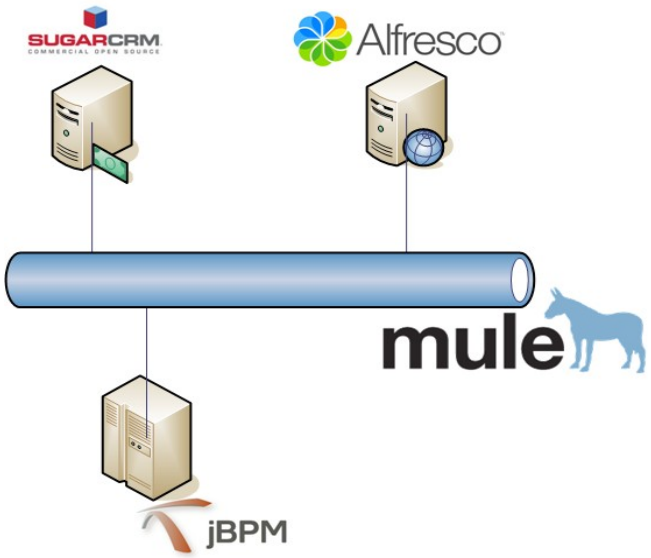
Como ejemplo de esquema de los aplicativos en un entorno anterior a la implementación de la arquitectura SOA sirva la siguiente figura:





Directorio LDAP

Mientras que la topología del bus quedará:



2.1 Servidor LDAP OpenLDAP

El servicio LDAP se emplea como repositorio de información relativa a usuario, grupos y roles. Es un servicio que permite la gestión y almacenamiento de información en una estructura arbórea y de acuerdo con plantillas o esquemas configurables.
Se ha escogido OpenLDAP como implementación de este servicio debido a que es una aplicación de código abierto y uso muy extendido en los sistemas operativos GNU/Linux.
Para implementar la estructura de usuarios y grupos se emplean los siguientes esquemas:

- RFC 2252/2256: LDAP versión 3
- RFC 1274: COSINE e Internet X500
- RFC 2798: InetOrgPerson
- RFC 2307: NIS

Los usuarios se modelarán con las clases:

- top
- inetOrgPerson
- posixAccount

Los grupos se modelarán con las clases:

- top
- posixGroup

Las ventajas de emplear LDAP como repositorio de usuarios son:

- Acceso centralizado a la información para la autenticación y autorización de las aplicaciones.
- Soporte de multitud de lenguajes de programación.
- Muchas aplicaciones se integran con LDAP como repositorio de información sobre usuarios y grupos.

2.2 Servidor Web Apache 2.2

Se necesita un servidor web para actuar como capa frontal frente a las peticiones de los clientes, cualquier servidor web capaz de comunicarse con un intérprete de PHP y de actuar como proxy inverso sirve para las necesidades del proyecto. Por ello, tanto Apache como Nginx u otros similares se pueden emplear como servidores web.

La función del servidor web en el proyecto consiste en:

- Servir el contenido estático (HTML, imágenes, Javascript, CSS ...)
- Proteger el acceso a determinados recursos que deben ser privados e internos del servidor.
- Proporcionar una capa de acceso única y homogénea al resto de aplicaciones de segundo plano: Motor PHP y Servidor de Aplicaciones.

El servidor web se integra con los servidores de aplicaciones mediante los módulos de proxy inverso o mediante el protocolo AJP si el servidor de aplicaciones es Apache Tomcat o Jetty. Los detalles de integración son dependientes del servidor web escogido y los servidores de aplicaciones designados para hospedar el proyecto.

Módulos requeridos:

- ✔ SSL: Capa de Sockets Seguros para acceso desde Internet a la aplicación.
- ✔ Encoding: para fijar la codificación de las páginas.
- ✔ Deflate: para comprimir las respuestas y ahorrar ancho de banda.
- ✔ Proxy: para actuar como intermediario para otros sitios web.
- ✔ Rewrite: para reescritura de direcciones URL.
- ✔ Headers: para mejorar la compatibilidad con ciertos navegadores.
- ✔ JK: para integrarse como Apache Tomcat Server.

- PHP: para ejecutar aplicaciones escritas en el lenguaje PHP.

2.3 Intérprete de PHP

PHP es un lenguaje de programación interpretado, muy popular en el entorno de los proyectos web y de código abierto. Permite la ejecución de páginas con contenido dinámico y se integra con los servidores web mediante módulos propios de cada servidor (mod_php en el caso de Apache) o el estándar FastCGI.
SugarCRM requiere PHP puesto que está implementado en éste lenguaje. Los módulos de PHP requeridos para un correcto funcionamiento son:

- Adodb: para conexión a bases de datos.
- Gd: para generar imágenes y gráficos.
- Imap: para comunicarse con servidores de correo IMAP.
- Ldap: para comunicarse con servidores LDAP.
- Mysql: para ejecutar consultas SQL sobre Mysql.

2.4 Servidor de Base de Datos Mysql 5.1

El motor de base de datos escogido o mejor dicho impuesto por SugarCRM es Mysql 5.1, al ser la única base de datos de código abierto con la que es compatible.
Mysql cumple con un conjunto de los estándares SQL y proporciona diferentes motores de almacenamiento para las tablas siendo los más populares MyISAM (no es ACID, pero permite búsquedas fulltext) o InnoDB (ACID y con soporte de claves foráneas).

2.5 SugarCRM 5.X

SugarCRM es el CRM de código abierto más extendido y con el mayor número de funcionalidades que existe. Es un aplicativo comparable a productos comerciales como Salesforce.com, pero sin coste alguno de licencia en la versión para la comunidad. SugarCRM ofrece la posibilidad de migrar de la versión CE o de Comunidad a las versiones Profesional o Corporativa.

Es interesante la comparativa entre versiones:

Funcionalidades	Comunidad	Profesional	Corporativa
Funcionalidades CRM			
Ventas. Marketing, Soporte y Colaboración	SI	SI	SI
Cuadros de Mando	SI	SI	SI
Plugins para MS Office	-	SI	SI
Predicción de Ventas	-	SI	SI
Informes	-	SI	SI
Dispositivos Móviles (PDA...)	-	SI	SI
Portal para Clientes	-	-	SI
Cliente Offline	-	-	SI
Cluster y Cloud Computing	-	-	SI
Personalización			
Constructor de Módulos	SI	SI	SI
Campos personalizados y Objetos	SI	SI	SI
Flujo de Trabajo	-	SI	SI
Integración			
API para Servicios Web	SI	SI	SI
API para Conectores Cloud	SI	SI	SI
Cargador de Módulos	SI	SI	SI
Fusión de Datos y Control de Calidad	-	SI	SI
Administración			
Control de Acceso por Perfil de Equipo	-	SI	SI
Seguridad a nivel de campo	-	SI	SI
Administrador de Módulos	-	SI	SI

La matriz anterior puede parecer que ofrece un producto bastante recortado, pero la realidad es totalmente distinta, siendo los informes la únicas característica que se echan en falta.

El desarrollo e implantación del CRM se apoya sobre los siguientes recursos de SugarCRM:

- **Constructor de Módulos**: es una herramienta dentro del propio SugarCRM que permite diseñar módulos definidos por sus campos de forma rápida y que, además permite desplegar dichos módulos sobre SugarCRM o bien empaquetarlos para su despliegue en otras instalaciones de SugarCRM.
- **Cargador de Módulos**: permite la instalación de módulos desarrollados por terceras partes dentro de una instancia de SugarCRM. Un catálogo de módulos gratuitos de terceras partes se puede localizar en **SugarForge**.
- **Campos personalizados**: los módulos de SugarCRM prefefinidos pueden ampliarse añadiendo nuevos campos.
- **Logic Hooks**: son funciones que se disparan cuando suceden eventos de aplicación tales como la carga de un registro, la modificación de un registro, el borrado de un registro... proporcionando capacidades de automatización y extensión de la aplicación.

2.6 Servidor de Aplicaciones Apache Tomcat 6.0.X

Es necesario un servidor de aplicaciones J2EE para albergar Alfresco. Tomcat 6.0 es un servidor de aplicaciones J2EE que soporta los estándares Servlet 2.5 y JSP 2.1 de la especificación J2EE para servidores de aplicaciones.
Por comodidad, se empleará la distribución de Alfresco que contiene un Apache Tomcat junto con el resto de aplicaciones de Alfresco y se personalizará para realizar el despliegue.

2.7 Alfresco 3.2r

El Gestor Documental facilita el almacenamiento estructurado de documentos, permitiendo implementar modelos documentales basados en meta-datos así como versionado de los mismos, búsqueda por contenido y otras interesantes características.
Dentro del proyecto se podría emplear para compartir documentación internamente dentro de una empresa o con los clientes, admite un modelo de autorización y autenticación muy flexible.
Alfresco es quizás el gestor documental más completo y popular y por eso se ha escogido. Proporciona entre otras características:

- Múltiples protocolos para acceder a los documentos: HTTP, WebDAV, CIFS, FTP
- Estructura basada en espacios o carpetas para organizar de forma lógica los documentos.
- Modelos documentales personalizables basados en metadatos.
- Sistema de reglas para gestionar documentos, que permiten ampliar la funcionalidad del Gestor Documental.
- Control de versiones sobre los documentos.
- Búsqueda basada en metadatos del modelo documental y en contenido de los documentos.
- Integración con MS Office y otras suites ofimáticas como OpenOffice.
- Servicios de transformación de documentos entre formatos. Por ejemplo de MS Word a Adobe PDF.
- Seguridad basada en ACLs para grupos y usuarios, junto con un sistema de permisos extensible.
- Motor BPM para implementar Flujos de Trabajo con los documentos.

A parte de estas características, proporciona varios mecanismos de extensión de gran ayuda para los desarrolladores como:

- Servicios web y Servicios REST
- Motor de scripts en JavaScript integrable con el sistema de Reglas.
- Desarrollo de modelos documentales basados en metadatos.
- Desarrollo de Flujos de Trabajo.
- API para modificación y ampliación de la funcionalidad interna de Alfresco.

2.8 Mule ESB 2.2

Mule es un ESB ligero que permite implementar arquitecturas SOA. Tiene una arquitectura basada en componentes, capaz de integrarse con simples clases Java o componentes más complejos, que permiten manipular mensajes. Sus características fundamentales son:

- Admite cualquier tipo de mensaje desde XML a imágenes o cualquier formato que se nos ocurra.
- Implementa los Patrones de Integración Empresariales para definir reglas de encaminamiento entre componentes y flujos de mensajería.
- Permite transformaciones sobre los mensajes recibidos.
- Admite diferentes transportes para los mensajes: HTTP, JMS, WS, REST, FTP, SMTP, IMAP, POP3...
- Puede configurarse a través de un IDE como Eclipse.

2.9 Jboss jBPM 3.3

Jboss jBPM es un motor BPM fácilmente integrable con aplicaciones hechas en Java. Las características fundamentales para su elección en este proyecto son:

- ✔ Flexibilidad y facilidad de integración.
- ✔ Soporta BPEL 1.1
- ✔ Tiene un entorno de desarrollo basado en Eclipse.
- ✔ Es un componente de Alfresco.
- ✔ Se integra con Mule ESB.

3 Implementación SugarCRM

SugarCRM se comporta dentro de los Casos de Uso de la Prueba de Concepto definida en el documento “Análisis SOA Silenus” de dos formas distintas:

- Como **fuente de eventos**. Por ejemplo cuando se crea una Cuenta o cuando se modifica una Oportunidad.
- Como **proveedor de servicios**. Por ejemplo cuando se cierra un proyecto en el ERP y debe cerrarse la correspondiente Oportunidad.

Que SugarCRM sea una fuente de eventos podría ser un problema si no proporcionase un mecanismo de extensión basado en eventos, denominado **Logic Hooks**. Si tal fuera el caso, la estrategia que se debería adoptar sería la de *polling* de cambios a través de la interfaz de servicios web, esos cambios detectados se transformarían en eventos para el ESB en la forma de mensajes JMS.

Los **servicios web** que proporciona SugarCRM son suficientes para implementar la Prueba de Concepto. Se proporcionan métodos para:

- Autenticación y creación de sesiones: login, logout, create_session, end_session.
- Localización, actualización y creación de registros: get_entry, get_entries, get_entry_list, search_by_module, set_entry, set_entries, set_entries_details.

De la definición de los Casos de Uso se requiere:

- Generar un evento cuando se cree o modifique una Cuenta.
- Generar un evento cuando se cree o modifique una Oportunidad.

- Proporcionar un servicio para modificar una Oportunidad y cambiarle su estado a cerrada.

Los eventos se generarán mediante **STOMP para PHP** que permite enviar mensajes JMS a un servidor ActiveMQ. El contenido de los mensajes será un XML con los datos necesarios para identificar el registro modificado o creado junto a sus datos característicos.

4 Implementación Alfresco

El papel de Alfresco dentro de la arquitectura SOA del proyecto es la de un **proveedor de servicios**. No genera evento alguno, y nos basta con apoyarnos sobre su extenso **API de servicios web**.
De la definición de los Casos de Uso se requiere:

- Creación de carpetas o espacios dentro de un espacio concreto.
- Verificar si existe una carpeta o espacio.
- Subir documentos a un espacio concreto.
- Verificar si existe un documento.
- Mover documentos a una carpeta o espacio.

5 Implementación OpenBravo

El rol de OpenBravo es muy similar al de SugarCRM. Se emplea para:

- Generar Eventos.
- Proporcionar servicios.

OpenBravo no proporciona un mecanismo de generación de eventos al uso, pero como emplea desde su versión 2.5 al framework ORM Hibernate, podremos aprovechar los **eventos de Hibernate** para generar los eventos que necesitamos. De esta forma, si hacemos uso de los eventos **post-insert** y **post-update** filtrados para el módulos de Proyectos generaremos los correspondientes mensajes JMS que se enviarán al bus ESB.

OpenBravo no requiere ningún tipo de modificación para ser empleado como **proveedor de servicios**, puesto que ya cuenta con un **API de Servicios REST**.

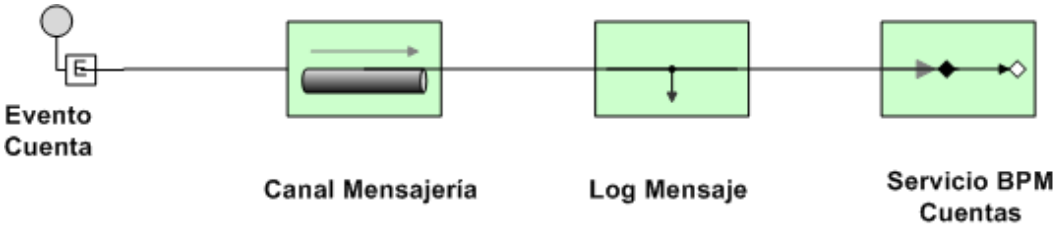
De la definición de los Casos de Uso se requiere:

- ✔ Generar un evento cuando se modifique un proyecto para conocer cuándo se ha cerrado, facturado o cobrado.
- ✔ Proporcionar un servicio para poder crear nuevos proyectos y nuevos clientes una vez se gane una oportunidad en el CRM.

6 Implementación Mule ESB

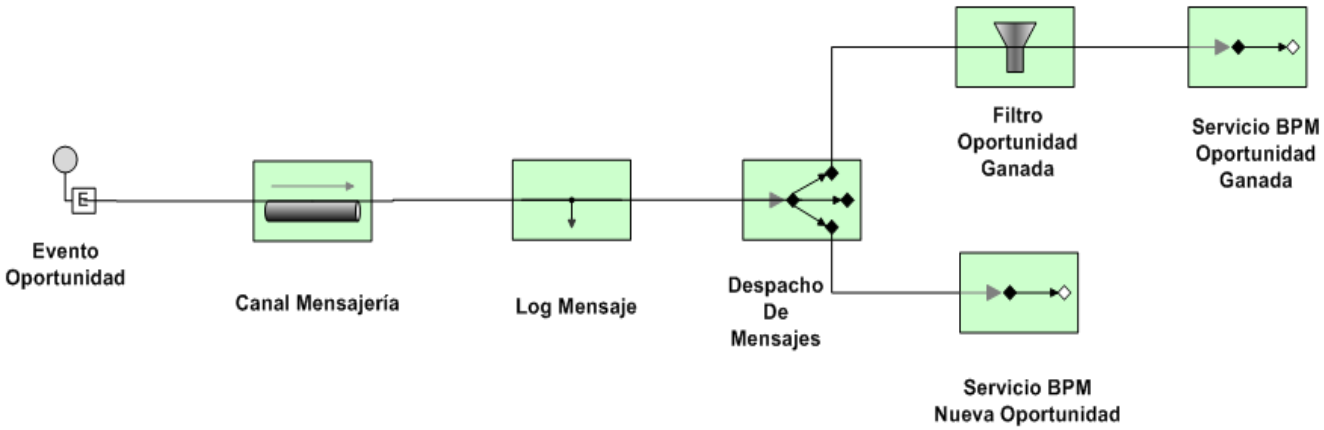
El ESB Mule debe ser configurado para permitir la recepción y enrutamiento de los mensajes hasta su punto final. Se definirán los caminos y los puntos de entrada y componentes en función de los casos de uso definidos.

6.1 ESB Cuentas



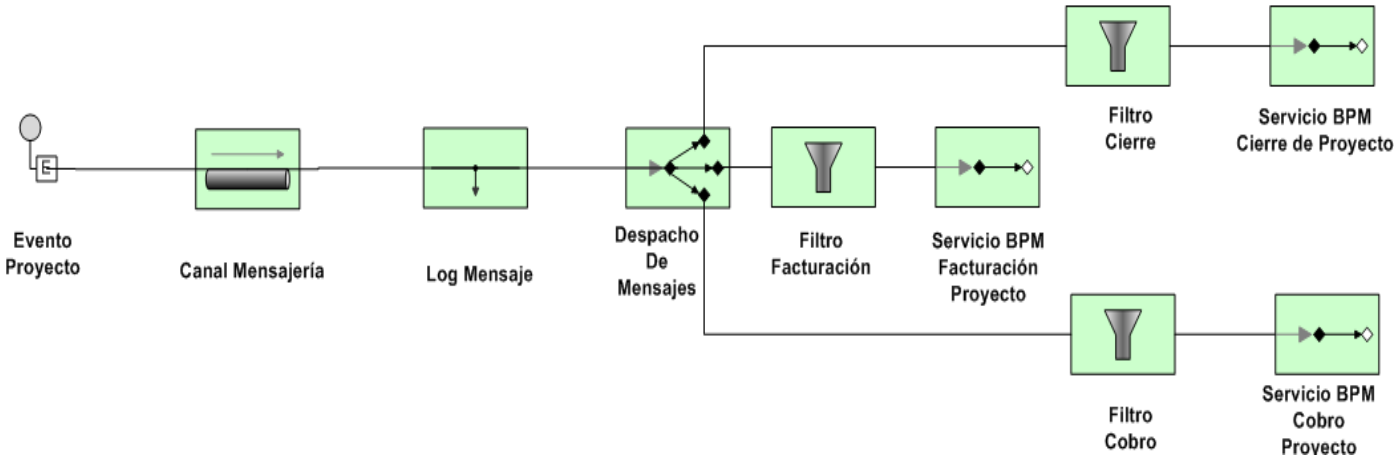
- El evento de modificación o creación de una cuenta, será enviado desde el Logic Hook de SugarCRM *after_save* del módulo Accounts.
- Se empleará STOMP para su envío asíncrono a un punto de escucha de ActiveMQ configurado dentro del propio Mule ESB.
- El contenido del mensaje será un XML que describa el bean de cuenta.
- El ESB despachará el mensaje hasta el Servicio BPM de Cuentas, activando el Flujo de Trabajo.
- Para trazar la recepción del mensaje se empleará un WireTap.

6.2 ESB Oportunidades



- El evento de modificación o creación de una oportunidad, será enviado desde el Logic Hook de SugarCRM *after_save* del módulo Opportunities.
- Se empleará STOMP para su envío asíncrono a un punto de escucha de ActiveMQ configurado dentro del propio Mule ESB.
- El contenido del mensaje será un XML que describa el bean de oportunidad.
- El ESB despachará el mensaje simultáneamente a varios caminos del ESB.
- El camino de Oportunidad Ganada cuenta con un filtro para dejar pasar sólo ese tipo de mensajes hasta el Servicio BPM de Oportunidad ganada.
- El camino de Nueva Oportunidad se aplicará siempre a los mensajes y activará el Flujo de Trabajo de Nueva Oportunidad.
- Para trazar la recepción del mensaje se empleará un WireTap.

6.3 ESB Proyectos



- El evento de modificación o creación de un Proyecto, será enviado desde el evento de hibernate adecuado de OpenBravo.
- Se empleará JMS para su envío asíncrono a un punto de escucha de ActiveMQ configurado dentro del propio Mule ESB.
- El contenido del mensaje será un XML que describa el bean de Proyecto.
- El ESB despachará el mensaje simultáneamente a varios caminos del ESB.
- El camino de Proyecto Cerrado cuenta con un filtro para dejar pasar sólo ese tipo de mensajes hasta el Servicio BPM de Cierre de Proyecto.
- El camino de Proyecto Facturado se aplicará siempre a los mensajes y activará el Flujo de Trabajo de Facturación de Proyecto.
- El camino de Proyecto Cobrado se aplicará siempre a los mensajes y activará el Flujo de Trabajo de Cobro de Proyecto.
- Para trazar la recepción del mensaje se empleará un WireTap.

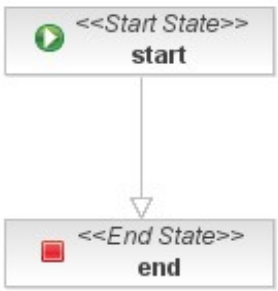
7 Implementación Jboss jBPM

El motor de procesos BPM se integra con Mule ESB a través de un conector o transporte desarrollado específicamente para Mule. Dicho transporte es capaz de enviar el mensaje proveniente del ESB al proceso e incluso especificar que continúe con un proceso ya existente.
Los ejemplos de procesos definidos en los Casos de Uso, carecen de tareas humanas, por lo que se pueden considerar como totalmente automáticos. Así mismo se trata de procesos que no tienen estado de parada, salvo el caso de que suceda un error.

7.1 Proceso de Cuentas

Este proceso debe:

- ✔ Crear la carpeta “Cuentas” dentro de la carpeta raíz si no existiera.
- ✔ Crear la carpeta “Cuenta X”, basada en la variable que proviene del mensaje del bus ESB dentro de la carpeta “Cuentas”.



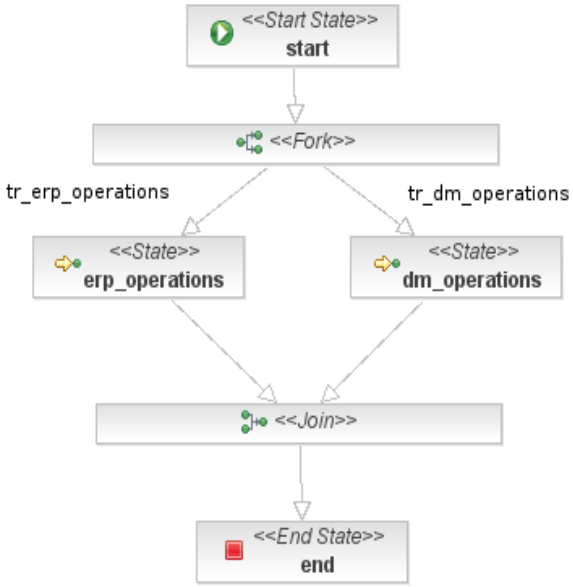
Esto se implementará en el evento de entrada al nodo final con la siguiente secuencia de pasos:

1. Script que genera la ruta a partir del mensaje ESB y la sitúa en una variable de contexto del proceso.
2. Acción que tomando la variable anterior invoca al servicio web de alfresco para crear la ruta.

7.2 Proceso Oportunidad Ganada

Este proceso debe:

- ✔ Obtener la cuenta correspondiente a la oportunidad del CRM.
- ✔ Crear la cuenta del cliente en el ERP si no existiera.
- ✔ Crear el proyecto en el ERP correspondiente a la oportunidad si no existiera.
- ✔ Crear la carpeta con el nombre de la oportunidad dentro de la carpeta de la cuenta del Gestor Documental.

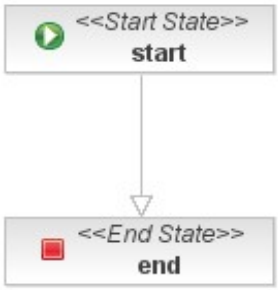


La implementación de este proceso paralelizará las operaciones que se realizan con el ERP y con el BPM. De tal manera que en el evento de entrada del estado **start** se obtendrá la cuenta del CRM, mediante una llamada al web service del CRM, a partir de la oportunidad que contiene el mensaje ESB y situará el resultado en una variable del proceso. A partir de la variable que contiene la cuenta, el estado **erp_operations** creará la cuenta de cliente en el ERP si no existiera, y el proyecto con el mismo nombre que la oportunidad y el estado **dm_operations** creará la carpeta de la oportunidad dentro del Gestor Documental.

7.3 Proceso Oportunidad Modificada

Este proceso debe:

- ✔ Crear la carpeta con el nombre de la oportunidad dentro de la carpeta de la cuenta del Gestor Documental.
- ✔ Crear las carpetas “Oferta Técnica” y “Oferta Comercial” dentro de la carpeta anterior.



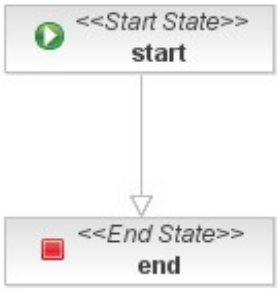
Esto se implementará en el evento de entrada al nodo final con la siguiente secuencia de pasos:

1. Llamada al al web service del CRM, a partir de la oportunidad que contiene el mensaje ESB, para obtener la cuenta y situará el resultado en una variable del proceso
2. Script que genera las rutas de las carpetas y las almacena como variables de proceso.
3. Acción que tomando la variable anterior invoca al servicio web de alfresco para crear la ruta de la oportunidad.
4. Acción que tomando la variable anterior invoca al servicio web de alfresco para crear la ruta de la carpeta “Oferta Técnica”.
5. Acción que tomando la variable anterior invoca al servicio web de alfresco para crear la ruta de la carpeta “Oferta Comercial”.

7.4 Proceso Proyecto Cerrado

El proceso debe:

- ✔ Cerrar la oportunidad en el CRM.
- ✔ Enviar un mail al comercial indicando el cierre del proyecto y de la oportunidad.



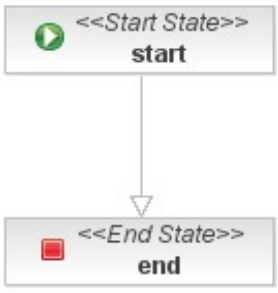
Esto se implementará en el evento de entrada al nodo final con la siguiente secuencia de pasos:

1. Llamada al servicio web del CRM para modificar la oportunidad y fijarle el estado de “Cerrada”.
2. Envío de correo electrónico al usuario asignado a la oportunidad para notificarle del cierre de proyecto.

7.5 Proceso Proyecto Facturado

El proceso debe:

- ✔ Obtener el documento de la factura.
- ✔ Enviar dicho documento a Alfresco, situándolo en la carpeta del proyecto.



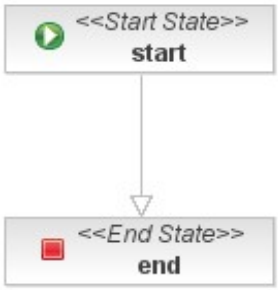
Esto se implementará en el evento de entrada al nodo final con la siguiente secuencia de pasos:

1. Llamada a la URL de Openbravo para la descarga del documento de la factura.
2. Invocación del servicio web de Alfresco para subir el contenido a la carpeta del proyecto, que se supone ya creada anteriormente.

7.6 Proceso Proyecto Cobrado

El proceso debe:

- ✔ Crear la carpeta “Proyectos Cerrados” dentro de la carpeta de la cuenta del cliente en el gestor documental.
- ✔ Mover la carpeta del proyecto a la carpeta anterior.



Esto se implementará en el evento de entrada al nodo final con la siguiente secuencia de pasos:

1. Script para calcular las rutas de las carpetas a partir el proyecto que se recibe como una variable de entorno en el mensaje del ESB.
2. Llamada al servicio web de Alfresco para crear la carpeta “Proyectos Cerrados”.
3. Llamada al servicio web de Alfresco para mover la carpeta del proyecto a la carpeta “Proyectos Cerrados”.