Modelado e implementación de un proceso de elaboración de vino

JBoss jBPM

Autor: Jesús Moreno Conde

19/02/2013

Tutor: Juan Manuel Vozmediano Torres

Resumen

El proyecto del que parte esta memoria tratará de integrar las tecnologías de la información y de la comunicación para la gestión de un proceso productivo real. Para llevarlo a cabo se hará uso de un sistema de administración conocido como BPM (Business Process Managment) que, mediante la utilización de un servidor de aplicación, en este caso JBoss, será capaz de comprender, ejecutar y gestionar este proceso de negocio. La definición del proceso se modelará mediante el estándar BPMN marcando las etapas fundamentales aparecidas durante la producción para una planta de elaboración de vino. El principal objetivo de este trabajo será marcar las pautas necesarias para la creación del modelo de negocio, así como demostrar la viabilidad final del sistema desarrollado. Como factores adicionales también se realizarán valoraciones personales a las distintas herramientas utilizadas en el desarrollo, remarcando las principales virtudes y deficiencias de cada una de ellas.

Índice

1. Introducción

El presente documento describe el proyecto fin de carrera titulado "Modelado e implementación de un proceso de elaboración de vino". A lo largo de esta memoria se expondrán las distintas estrategias utilizadas para el diseño del proyecto, así como otros aspectos de interés relacionados con la implementación final de la solución desarrollada.

1.1. Estructura del documento

Para la estructuración del documento se ha seguido un esquema donde, en primer lugar, se realizará un breve barrido conceptual de las distintas tecnologías utilizadas en el proyecto. Tras esta breve introducción técnica se procederá con la definición del entorno de aplicación donde se fijará la secuencia de acciones necesarias para la elaboración de vino. Una vez producido el acercamiento al proceso de negocio real se realizará la posterior definición del esquema conceptual del modelo, resumiendo también las principales hipótesis simplificadoras tenidas en consideración. Ya conocido el modelo de definición para el proceso se proseguirá con la implementación específica para cada uno de sus módulos, haciendo hincapié en sus etapas más relevantes. Una vez terminada la implementación, se procederá con la definición de la interfaz de usuario así como otros aspectos relacionados con la simulación final del proceso desarrollado. Para finalizar, se realizará una breve valoración del trabajo realizado destacando además los aspectos más interesantes aparecidos durante su elaboración. Finalmente se incluirá un último apartado con posibles propuestas para la definición de futuras líneas de trabajo para posteriores desarrollos del proyecto.

1.2. Motivaciones y objetivos

Tras la evaluación de las distintas técnicas de gestión y planificación de negocio existentes en el mercado actual, la gestión de procesos de negocio se presenta como una potente herramienta a tener en cuenta para cualquier entorno empresarial. Como cabe esperar, la integración de estas tecnologías de la información aplicadas a la gestión de procesos de producción industrial puede plantear importantes avances operativos tan determinantes como la eficiencia o el rendimiento. Estos y otros aspectos resultan de vital importancia para el desempeño final de la producción y, por ello, plantean importantes beneficios al proceso de negocio sobre el que se adapten. De forma particular, este proyecto se enmarca dentro del

sector vinícola, un entorno en constante crecimiento donde la aplicación de herramientas como jBPM puede mejorar en gran medida las prácticas utilizadas en la actualidad.

Dentro de este marco de trabajo se desarrollará este proyecto, cuya principal motivación buscará integrar esta tecnología dentro de un proceso de elaboración vinícola general. Como objetivo complementario, y a fin de esclarecer las cualidades y deficiencias de la tecnología, también se orienta el desarrollo del proyecto hacia un análisis real y detallado de la herramienta. A partir de este análisis se podrá conocer con mayor profundidad su funcionamiento interno, permitiendo además evaluar con una perspectiva más amplia sus posibles futuras implementaciones.

2. Antecedentes

2.1. Tecnología de flujos de trabajo

2.1.1. Definición de flujos de trabajo

Un flujo de trabajo es una herramienta que permite la implementación técnica de un proceso de negocio a través de la definición de los aspectos funcionales y operacionales propios de la actividad. Para llevarla a cabo será necesaria la asociación coordinada entre documentos, información y tareas; todas ellas relacionadas con los distintos participantes y sujetas a una serie de reglas definidas por el entorno sobre el que se modela. La secuencia correlativa de los elementos, la sincronización entre tareas o la información relacional entre procesos son algunos de los aspectos operacionales más representativos a tener en cuenta en la definición de estos flujos de trabajo.

Para logar la correcta definición de un flujo de trabajo dentro de un área específica de negocio se requiere no sólo la presencia de un experto en modelado, sino también un conocimiento exhaustivo de la actividad a modelar. Mediante el adecuado uso de los flujos de trabajo se podrán definir procesos fidedignos, capaces de trasladar el comportamiento real del sistema a proceso modelado y conseguir, de este modo, el posible control automatizado del mismo.

Actualmente, las nuevas tendencias en la regulación de organizaciones y empresas muestran que los flujos de trabajo son comúnmente utilizados como herramientas de gestión. De esta forma se consigue mejorar sustancialmente la agilidad y el rendimiento de tareas tanto administrativas como comerciales. Por otro lado, la constante evolución en las tecnologías asociadas a estos flujos de trabajo está permitiendo incrementar notablemente la efectividad de la automatización y del control de estos procesos.

La tecnología de los flujos de trabajo está cimentada en la creencia de que algunas tareas se realizan con mayor efectividad de forma automática. Por ello se delega el control de este tipo de tareas a computadoras. De este modo se obtiene mayor fiabilidad y rapidez en el procesamiento de la información, facilitando además el manejo de los elementos asociados al flujo y proporcionando así mejoras sustanciales en la ejecución.

2.1.2. Representación e Interpretación de Flujos de Trabajo

Al margen de la actividad que se pretenda modelar, la definición y representación de procesos de negocio mediante flujos de trabajo está sujeto a una serie de reglas. En la siguiente enumeración se establece una breve descripción sobre los elementos más comunes presentes en este tipo de modelado:

- ✓ **Tareas**: las tareas se relacionan con el conjunto de actividades o acciones desempeñadas en un proceso. Generalmente son realizadas por una única persona y están englobadas dentro de una competencia específica o rol.
- ✓ **Usuarios**: se corresponden con el conjunto de participantes incluidos en el proceso.
- ✓ **Roles**: define cada una de las distintas competencias existentes en el sistema con independencia de los usuarios a los que vayan a ser asignados. De este modo, un participante podrá tener más de un rol en el flujo.
- ✓ **Rutas**: representa la secuencia de pasos establecida en el sistema para garantizar la consecución de los objetivos del mismo. Para ello, todos los usuarios del proceso deben poder formar parte de esta secuencia. Se definen además varios tipos de ruta, apareciendo las rutas fijas y las rutas condicionales. En las rutas fijas, el camino establecido es unívoco de forma que siempre permanece establecida la misma secuencia de tareas. Por otro lado, las rutas condicionales dependerán de la evaluación de diversas condiciones o variables del sistema para decidir el siguiente salto en su secuencia de tareas.
- ✓ **Reglas de transición**: son un conjunto de reglas lógicas que determinan la siguiente acción a realizar. La definición de estas reglas puede llegar a ser compleja, contemplando múltiples opciones, variaciones o excepciones.
- ✓ **Datos**: se corresponde con toda la información incluida dentro del flujo ya sean documentos, imágenes, archivos, etc. Se Distinguen dos categorías de datos:
  - **Datos de control**: conjunto de datos útiles para la lógica del sistema.
  - **Datos relevantes**: datos utilizados para las tareas de cálculo de rutas correspondientes al siguiente salto en el flujo.
- ✓ **Eventos**: establece una interrupción al comportamiento secuencial del flujo. Suele contener información asociada al evento, además posee un origen y puede tener

uno o más destinatarios. Los eventos pueden ser lanzados voluntariamente por los usuarios o de forma automática por el sistema.

- ✓ **Plazos**: conjunto de tiempos asociados a ciertos elementos del sistema. Se permite la asociación de eventos a la finalización de estos plazos o deadlines.
- ✓ **Políticas y Procesos**: definen la nomenclatura y la forma de actuar de los distintos flujos sobre los que se pretenda realizar el modelado. Además se establecen las interacciones y las sentencias para e el manejo de los distintas tareas contenidas en el flujo de trabajo.

Desgraciadamente, el principal problema que aparece a la hora de modelar procesos de negocio reside en el uso de las metodologías apropiadas para el análisis y la interpretación de los requisitos reales del sistema. Bajo esta premisa aparecen diferentes niveles de comprensión del proceso a partir de los cuales se deberán formular las condiciones aparentes que modelen el comportamiento final del sistema modelado. Por este motivo y como puede observarse en la siguiente figura, el desarrollo obtenido para el modelo software se originará como resultado de la yuxtaposición de una serie interpretaciones llevadas a cabo por distintos actores conocedores del proceso.

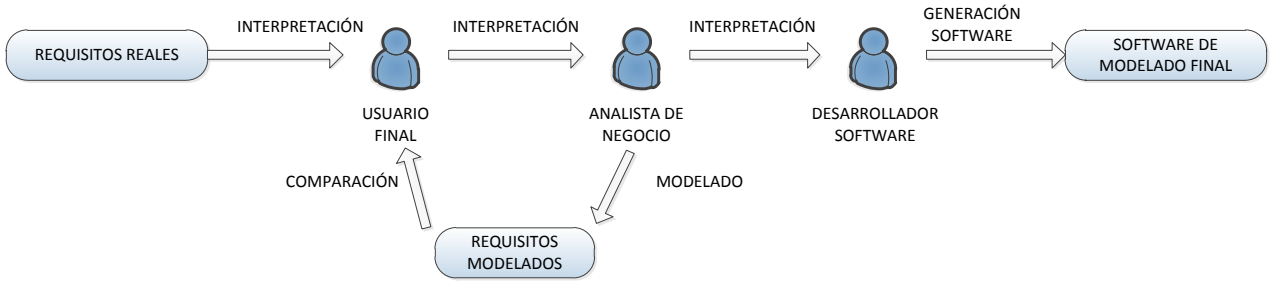


Figura 1: Esquema conceptual de desarrollo para el modelado de un proceso de negocio

En el diagrama anterior se observa la aparición del analista de negocio como figura intermedia entre el usuario final y el desarrollador software. Este actor interpreta un papel fundamental en el modelado encargándose de la traducción de las condiciones y los requisitos del sistema entre ambos extremos del diagrama. De este modo, el analista de negocio deberá caracterizar el modelo de negocio para que la diferencia entre los requisitos reales y los requisitos modelados sea mínima. Para lograr este grado de compromiso, el analista deberá no sólo conocer en profundidad el proceso a modelar, sino que también deberá realizar un estudio exhaustivo de las distintas actividades a realizar por los actores o participantes del proceso. Por último, el analista deberá además ser capaz de transmitir este conocimiento al

desarrollador software y trasladarlo así al lenguaje de modelado de flujos de trabajo requerido por el proyecto.

Como cabe esperar esta tarea será de vital importancia para la futura implementación del proceso. Por este motivo, cualquier inconsistencia a la hora de trasladar las características del sistema así como su posterior modelado repercutirá significativamente en la solución final obtenida.

Aunque el concepto de automatización asociado a los flujos de trabajo se encuentra estrechamente ligado con la optimización de procesos de negocio, la aplicabilidad real de esta tecnología resulta bastante limitada. Por este motivo, el desarrollo del proyecto se orientará hacia otra tecnología asociada, conocida como Business Process Managment y presentada en la siguiente sección de la memoria.

2.1.3. Business Process Managment (BPM)

Se denomina BPM al conjunto de herramientas y servicios que facilitan la administración de un proceso de negocio. Como se comentó en el cuadro anterior, el principal objetivo perseguido por este conjunto se centra en la mejora del desempeño organizativo de los procesos, buscando optimizar la eficiencia y eficacia de los mismos. De este modo, las competencias fundamentales de BPM abarcarán el diseño, el modelado, la organización, la documentación y la optimización de toda la actividad, de una forma estable y controlada. A través de estas capacidades aparece un ciclo continuo y cerrado capaz de desarrollar, en cada iteración, la propia implementación del proceso. Por todo ello, el sistema desarrollado por BPM ambicionará la continua mejora del proceso, permitiendo a los desarrolladores tener un acercamiento mayor a la actividad real sobre la que se desarrolla. Este ciclo aparece gráficamente representado en la figura que se muestra a continuación.

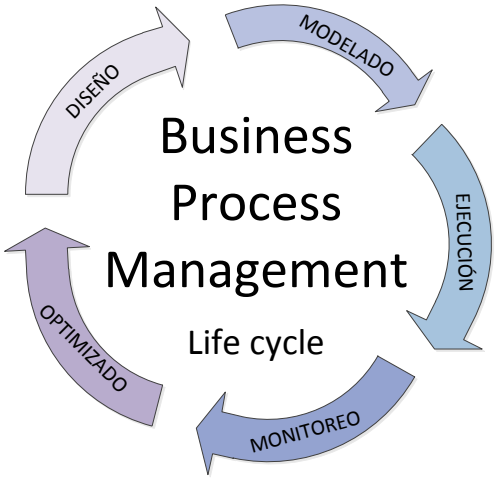


Figura 2: Ciclo de vida para un gestor de procesos de negocio

El ***DISEÑO*** es una etapa que incluye la identificación de los procesos existentes y el diseño de futuras líneas de desarrollo para nuevos procesos. Dentro de esta etapa se incorporará la representación de los flujos además de otros factores como alertas y notificaciones, procedimientos de operación estándar, acuerdos de nivel de servicio y mecanismos de entrega de tareas, entre otros. Cabe destacar que un diseño correcto y eficiente reducirá sustancialmente el número de problemas durante el tiempo de vida del proceso.

Durante la etapa de ***MODELADO*** se realizará la correspondiente traducción del diseño realizado en la etapa anterior al lenguaje requerido para su ejecución. Para completar esta etapa se incluirán una serie de variables adicionales. De este modo se buscará extender la operación del proceso con total independencia de las circunstancias en las que se encuentre.

Para la ***EJECUCION*** se hará uso de aplicaciones que realicen el seguimiento secuencial del proceso definido en las etapas anteriores. Por medio de esta ejecución se obtendrá una aproximación controlada sobre la automatización general del proceso en cuestión. La variedad de las aplicaciones utilizadas con este fin adoptará diferentes niveles de complejidad en relación con los requisitos de ejecución estipulados durante esta etapa.

Durante la ***MONITORIZACION*** se capturarán los distintos procesos que se encuentren en ejecución. A través de ellos se almacenará la información concerniente a su estado real, así como datos relativos a su operación específica. El grado de monitorización dependerá directamente tanto de la información del negocio que se quiera evaluar y analizar, como de las características propias del proceso que se esté considerando

La última etapa se corresponde con la ***OPTIMIZACIÓN***. En esta etapa se recuperará la información sobre el rendimiento del proceso proveniente de la etapa anterior. A partir de ella se analizará en busca de posibles deficiencias, cuellos de botella u otros aspectos que puedan limitar el funcionamiento optimizado del proceso. Por otro lado, ciertos aspectos como la reducción de costes, o mejoras en algunas de las etapas del proceso de negocio pueden también hacerse patentes, permitiendo así mejorar el rendimiento general del mismo.

2.1.4. Business Process Managment Notation (BPMN)

BPMN es el acrónimo de Business Process Management Notation. Este acrónimo se corresponde con un estándar para la notación gráfica de los procesos de negocio utilizados en la plataforma JBPM. Esta notación permite unificar la representación de los procesos de negocio facilitando, de forma intuitiva, la organización gráfica de sus elementos. Los diagramas y gráficos usados en BPMN se integran perfectamente a las tecnologías de la información aplicadas comúnmente en las empresas dentro del formato establecido para los flujos de trabajo. El principal objetivo perseguido por esta notación consiste en estandarizar la representación de los procesos y facilitar así su utilización entre los distintos participantes involucrados. De este modo se busca crear un lenguaje de comunicación fiable, capaz de servir de enlace entre el diseño y la implementación final de estos procesos.

Aunque el estándar perseguido por BPMN se encuentra bastante extendido, en la actualidad existe una amplia variedad de lenguajes y herramientas destinadas a esta representación. Al margen de ello, para el objetivo perseguido por este proyecto este lenguaje de representación no sólo aporta interesantes características de rendimiento y aplicabilidad sino que también añade otros factores a tener en cuenta. Entre estos factores, resulta interesante destacar los siguientes:

- BPMN representa un estándar internacional de modelado de procesos consolidado y aceptado por la comunidad.
- BPMN resulta completamente independiente de la metodología utilizada en el modelado de procesos.
- BPMN permite crear un puente que limite la brecha existente entre el proceso de negocio y su implementación.

- BPMN consigue modelar los procesos de forma unificada y estandarizada, consiguiendo un cómodo entendimiento entre los distintos participantes implicados.

2.1.5. Herramientas BPMN

En el mercado existen múltiples herramientas dedicadas al modelado de procesos en BPMN. Debido al éxito en la implementación de procesos con esta notación han ido surgiendo a lo largo de los años numerosas herramientas que dan soporte a esta nomenclatura. La mayoría de ellas ofrecen un entorno de diseño bastante depurado y de fácil uso, presentando además interfaces de usuario estéticas y cómodas.

Para la realización de este proyecto se optó inicialmente por la utilización del software Bizagi Process Modeler. Esta herramienta de modelado desarrollada por BizAgi permite un diseño rápido y completo de procesos de negocio basándose en el estándar de modelado BPMN. Adicionalmente, este software aportaba una interfaz de usuario clara y atractiva. A pesar de ello, la elección de este programa en concreto fue debida principalmente a otros motivos: a la **facilidad** de uso, ya que resulta bastante intuitivo, y a la **rapidez** en su aprendizaje, ya que existían mucha documentación asociada.

En la siguiente captura de pantalla se recoge la interfaz de trabajo de este software de modelado. Como puede observarse ofrece una interfaz de modelado cómoda y sencilla. Los distintos elementos utilizables permanecen recogidos en la columna izquierda del editor mientras que el entorno de edición representa la mayor parte de la ventana. Esta captura se corresponde con el diagrama de modelado completo para el proceso de elaboración de vino. Este diagrama, así como los subprocesos incluidos en él, se presentarán con mayor profundidad en el anexo correspondiente al modelado del proyecto mediante esta herramienta.

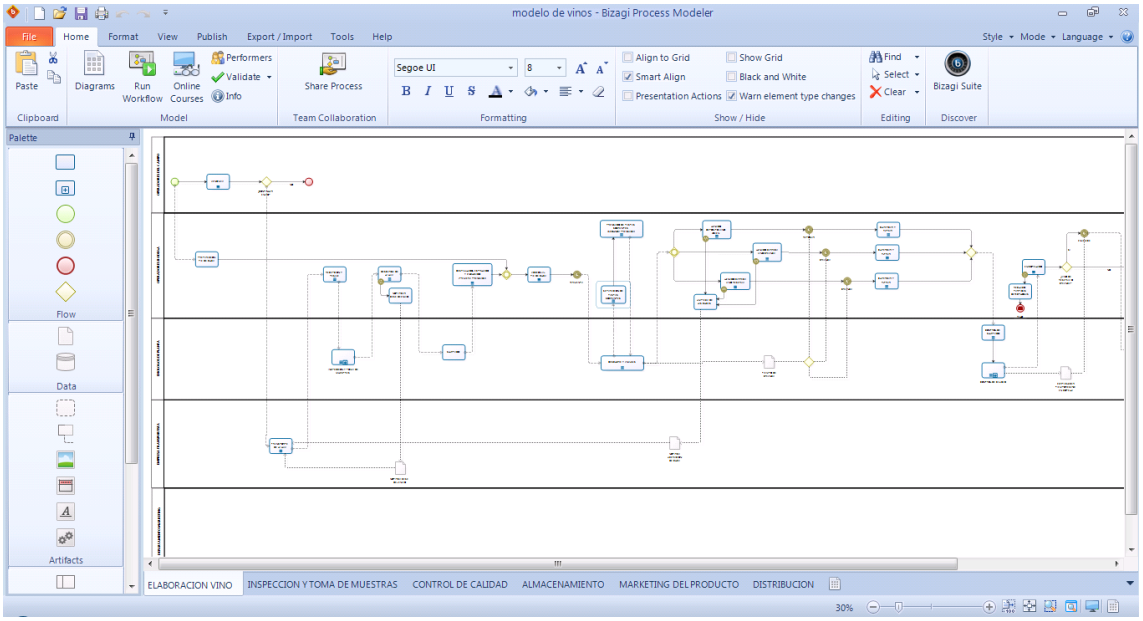


Figura 3: Modelado del proceso general de elaboración de vino mediante el software de Bizagi Modeler

Por otro lado, el software presentado anteriormente ofrece además una compatibilidad directa con otro programa de la misma compañía llamado Bizagi BPM Suite. Este programa, al igual que otras "suites" de BPM, permite ejecutar, monitorizar y gestionar los distintos procesos definidos en el programa de modelado. Una vez integrados ambos programas, el conjunto obtenido permitiría generar una solución plausible y adecuada a los objetivos principales que se abordan en este proyecto. Sin embargo, durante la integración del mismo (entre el proceso definido en el Bizagi Modeler y el software Bizagi BPM Suite) aparecieron numerosos problemas de compatibilidad entre sus versiones. Ese inconveniente unido a la gran cantidad de fallos recogidos por el BPM Suite en su versión gratuita provocó un cambio en la orientación del proyecto hacia otros entornos más estables y libres. Gracias a este cambio se decidió finalmente por la utilización del entorno de desarrollo Eclipse a través de un "plugin" específico para la definición de procesos mediante el estándar BPMN. En secciones posteriores de la memoria se verá de forma detallada tanto el proceso de instalación como la configuración de este entorno a partir del cual se ha desarrollado principalmente este proyecto.

Antes de continuar con el siguiente apartado de la memoria conviene hacer una breve introducción del entorno de desarrollo Eclipse y, más concretamente, del componente asociado a la definición de procesos en BPMN. Los correspondientes pasos necesarios para

llevar a cabo la instalación y la configuración del entorno de trabajo se encontrarán recogidos en el anexo titulado "Manual de configuración".

Una vez desplegado correctamente el entorno de desarrollo se podrá iniciar la definición gráfica de procesos siguiendo, para ello, los pasos que se describen a continuación. El primero de estos pasos consistirá en la creación de un nuevo proyecto incluido dentro del desplegable correspondiente a los proyectos jBoss jBPM. Al seleccionar esta casilla aparecerá otra ventana dedicada, en este caso, a nombrar el nuevo proyecto. Ambas ventanas están recogidas en las capturas de pantalla presentadas a continuación.

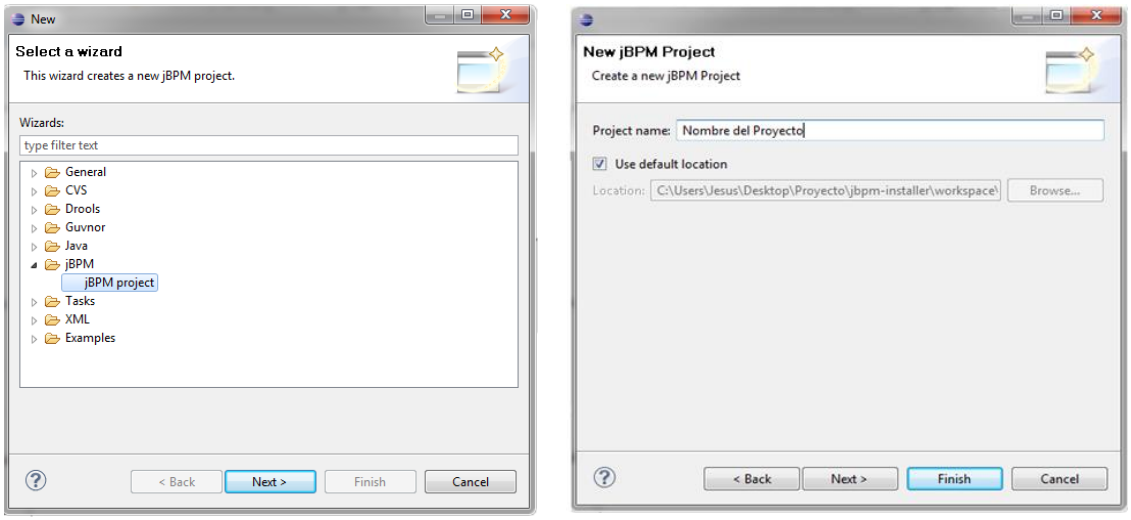


Figura 4: Creación de un nuevo proyecto jBoss jBPM mediante el componente instalado en Eclipse Indigo.

Tras crear correctamente el nuevo proyecto se abrirá la interfaz del entorno de desarrollo para la edición gráfica de diagramas en BPMN. Este editor, que se puede observar en la siguiente captura de pantalla, distingue tres ventanas principales además de una completa barra de herramientas localizada en la parte superior del conjunto.

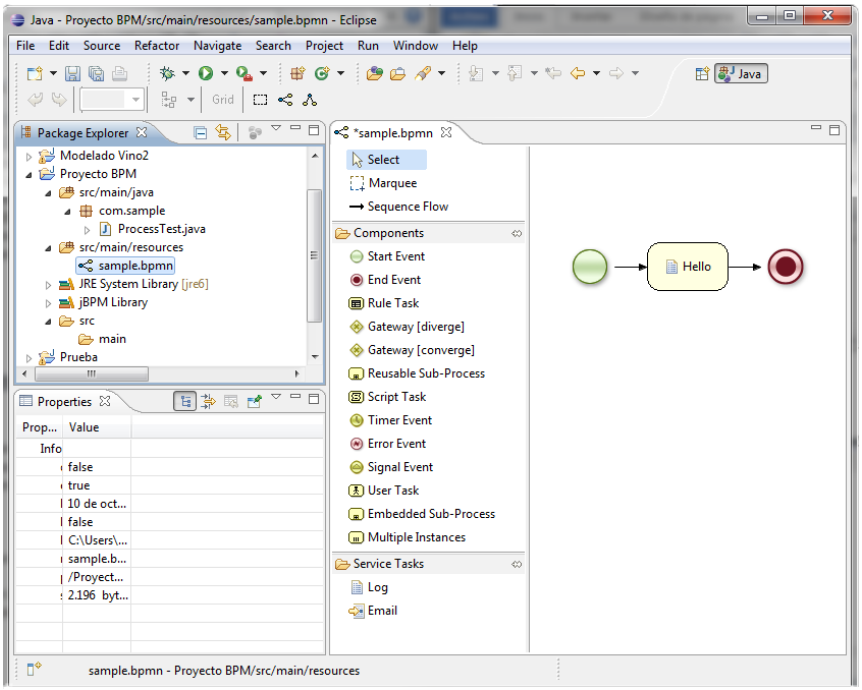


Figura 5: Interfaz de desarrollo gráfico para el modelado en BPMN.

La primera de estas ventanas, denominada “Package Explorer”, hace referencia al explorador del entorno de trabajo de Eclipse. Para los proyectos creados como jBoss jBPM, la estructura de directorios es tal y como se representa en la captura de pantalla. Los elementos principales de este árbol de directorios serán el correspondiente a “src/main/java” donde se almacenan las clases java definidas en el proyecto; el asociado a “src/main/resource” que incorpora todos los flujos BPMN fijados en el proyecto y, por último, todas las librerías incluidas, ya sean las definidas por el Java Runtime Environment (JRE); las recogidas por la jBPM o cualquier otra librería que se quiera referenciar.

La segunda ventana a tener en cuenta en el entorno de desarrollo se denomina “Properties”. Esta ventana, como su nombre indica, reflejará las propiedades intrínsecas de los elementos seleccionados en el entorno. A través de ella, será posible modificar y definir las características internas de los distintos nodos utilizados en el editor gráfico BPMN.

La tercera ventana a destacar hace referencia a la nombrada con el propio nombre del archivo a editar, en este caso “simple.bpmn”. Dentro de esta interfaz se hace posible la edición de los diagramas BPMN por medio de la elección de los nodos suministrados en la barra localizada a su izquierda. Entre todos los elementos incorporados por esta barra de edición, se pueden distinguir la gran mayoría de los facilitados en el anexo correspondiente al estándar de notación gráfica BPMN.

2.2. Servidor de aplicaciones JBoss

Las siglas JBoss se refieren al servidor de aplicaciones de código abierto implementado en su totalidad en Java. Este servidor, al ser de código abierto, fundamenta su desarrollo en una red mundial de colaboradores y testadores organizados bajo la gestión interna de los foros de www.jboss.org. Mientras que, por un lado, esta red de colaboradores representa un pilar fiable sobre el que plantear las posibles dudas de los desarrolladores, por el otro, la licencia abierta permite a los usuarios descargar, utilizar y distribuir su contenido sin ninguna restricción asociada.

Entre las características más atrayentes a tener en cuenta en la elección de este servidor, resulta interesante destacar las siguientes:

- Rápido, ligero y escalable
- Ofrece una arquitectura modular
- Resulta flexible y consistente
- Permite la administración de dominios
- Cumple con los estándares
- Aporta servicios middleware para cualquier objeto Java
- Incorpora plataformas de alto rendimiento para aplicaciones empresariales
- Permite despliegue en paralelo
- Administración elegante por medio de interfaz JMX
- Resulta bastante fiable en entornos empresariales

A raíz de ello, este servidor ha podido fácilmente desarrollarse y extenderse por todo el panorama internacional convirtiéndose, actualmente, en uno de los servidores más populares del mercado.

2.2.1. Arquitectura del servidor

En cuanto a su funcionamiento interno, JBoss implementa todo el paquete de servicios definidos en la plataforma J2EE (Enterprise Edition), la cual se dedica al desarrollo y la ejecución de software de aplicaciones. A través de este paquete de servicios se incorporan los estándares necesarios para usar JBoss como contenedor de componentes, ya sea Java Servlet Pages (JSP), Java Message Service (JMS) o Enterprise JavaBeans (EJB), entre otros. Por medio

de estos contenedores se pueden definir además diferentes capas en la aplicación (gestión de sesiones, acceso remoto a base de datos, interfaces usuario, etc.) de forma centralizada reduciendo, de este modo, la complejidad en el desarrollo de las aplicaciones y servicios.

Por otro lado, este servidor también incorpora otros servicios más allá de los propiamente definidos por la plataforma J2EE. A partir de ellos se asegura extender las capacidades de todo el conjunto hacia un entorno más completo y avanzado. Entre todos los servicios incluidos en JBoss, algunos de los más representativos quedan representados en el diagrama propuesto a continuación:

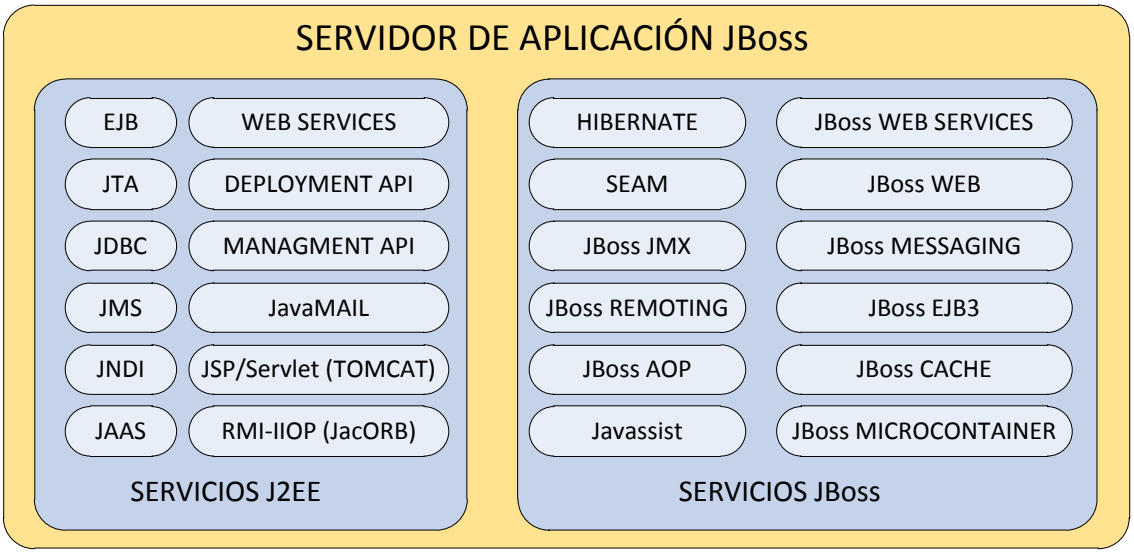


Figura 6: Elementos y servicios incorporados en el servidor de aplicaciones JBoss

Bajo la agrupación de todos estos elementos, el núcleo del servidor JBoss se encuentra implementado sobre Java Managment eXtension (JMX). Este núcleo, construido como una aplicación J2EE, permite el despliegue de servicios Mbeans para la gestión dinámica de dependencias y parámetros configurables en los distintos servicios del servidor.

A partir del modelo JMX se consigue, entre otras funciones, modificar una determinada configuración, gestionar el comportamiento de una aplicación, conocer las estadísticas o los cambios de estado de un servicio concreto, interoperar con otras tecnologías, etc.

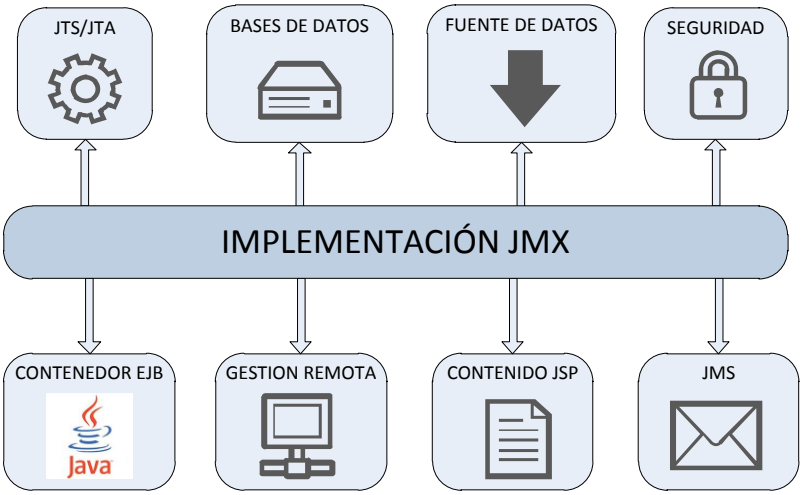


Figura 7: Integración entre JMX y los componentes estándar de JBoss

De entre todas sus acciones, algunas de las más interesantes vienen recogidas a continuación:

- ✓ Mostrar el árbol JNDI (Java Naming and Directory Interface)
- ✓ Forzar un volcado de memoria
- ✓ Mostrar el uso de la memoria
- ✓ Gestionar el escáner de despliegues
- ✓ Redesplegar una aplicación
- ✓ Detener JBoss
- ✓ Conocer el estado de los EJB (Enterprise JavaBeans) desplegados e instanciados
- ✓ Acceder a la base de datos Hypersonic (HSQL).

Para la navegación a través de la interfaz, JBoss incorpora una consola accesible desde varias URLs. De este modo y por medio de un navegador, se mostrarán todas las MBeans de acceso público manipulables por los usuarios en el servidor. El contenido mostrado por estas URLs (tanto por http://localhost:8080/jmx-console/ como por http://localhost:8080/web-console/) está, a modo de ejemplo gráfico, recogido en las siguientes capturas de pantallas.

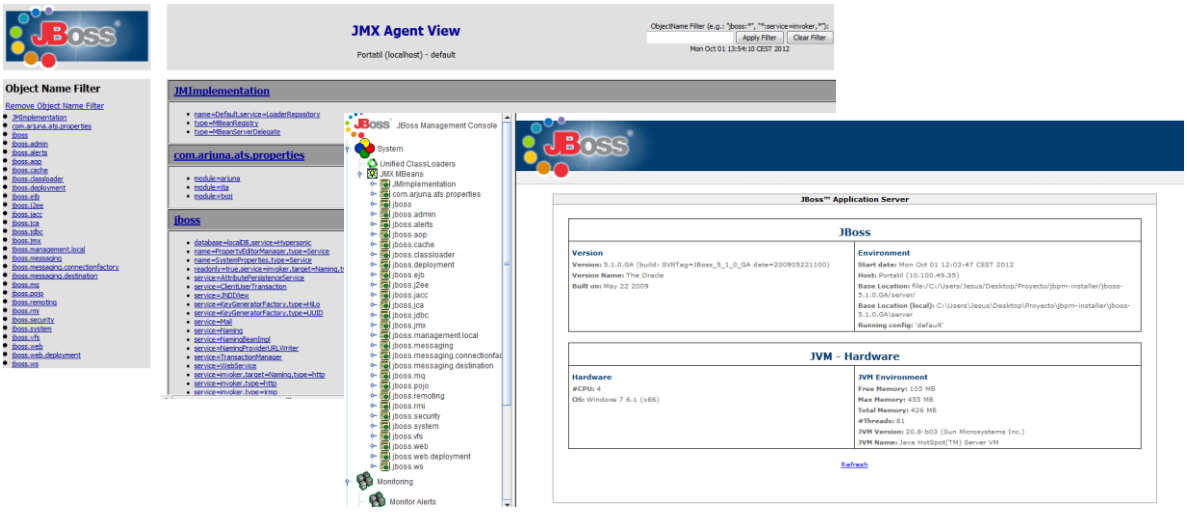


Figura 8: Capturas de pantalla de las consolas de acceso JMX para el servidor JBoss

2.2.2. Base de datos HypersonicSQL y conectores JDBC

Con objeto de aclarar las tecnologías más importantes utilizadas en este proyecto resulta interesante incidir brevemente sobre las características principales de la base de datos empleada. En concreto, esta base de datos se corresponde con HyperSQL (HSQL), un motor de bases de datos ligero, de código abierto, estable y completamente implementado en Java. Entre sus características principales se pueden destacar las siguientes:

- ✓ Soporta la sintaxis SQL estándar.
- ✓ Posee un cómodo sistema gestor de bases de datos.
- ✓ Todos sus procedimientos están almacenados en Java
- ✓ Permite la utilización de restricciones y disparadores con operaciones de INSERT, DELETE o UPDATE
- ✓ Soporta la integridad referencial (claves foráneas)
- ✓ Tiempo de arranque optimizado (mínimo)
- ✓ Gran velocidad en las operaciones sobre la base de datos: SELECT, INSERT, DELETE y UPDATE.
- ✓ Almacenamiento en disco hasta 8GB
- ✓ Al estar escrito en Java ofrece portabilidad
- ✓ Ofrece un controlador JDBC fácilmente utilizable de forma embebida en una aplicación

Tal y como se ha comentado en la enumeración anterior, este motor de base de datos incorpora un gestor cómodo y fiable. Este elemento, que puede invocarse desde la interfaz JMX definida anteriormente, ha sido de vital importancia a la hora de evaluar la interacción de los procesos definidos en el proyecto y su base de datos correspondiente. En la siguiente captura de pantalla pueden observarse dos ventanas superpuestas que corresponden tanto a la pantalla para la configuración de conexión con la base de datos como a la propia interfaz de operación del gestor.

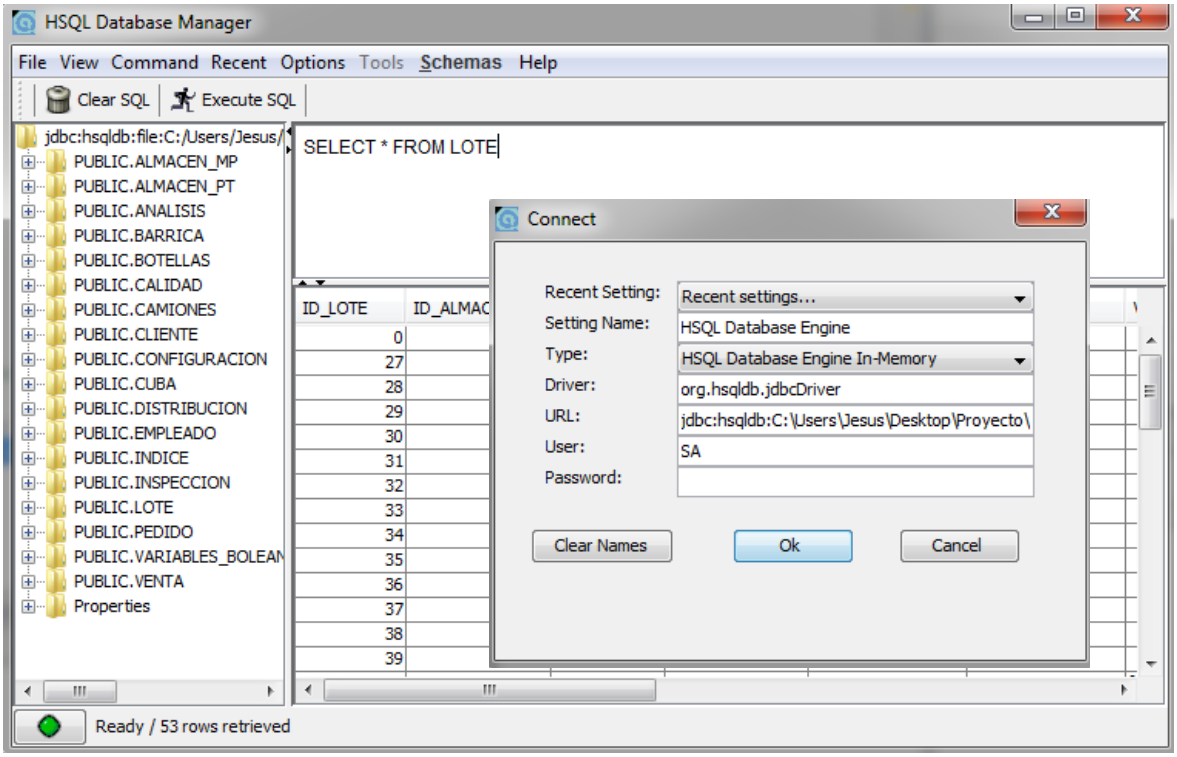


Figura 9: Capturas de pantalla para el sistema gestor de base de datos para HSQL y correspondiente ventana de conexión

En la primera de ellas, titulada "Connect", se establecen los parámetros de configuración para la conexión entre el sistema gestor y la base de datos. Los principales parámetros los representan el tipo de conexión, el driver utilizado, y la URL específica con la ubicación de la base de datos en memoria. Para mayor seguridad se pueden asociar nombre de usuario y contraseña a la base de datos y evitar así posibles accesos indebidos o no autorizados.

La segunda pantalla, titulada "HSQL Database Manager", se corresponde con la interfaz de usuario para el sistema gestor de bases de datos. Esta interfaz se encuentra distribuida en tres partes. La primera, localizada en la parte izquierda de la pantalla, hace referencia a un árbol esquemático donde están representadas todas las tablas creadas en la

base de datos. Las otras dos partes se corresponden con la línea de comandos, situada en la parte superior, y con el resultado de la operación, ubicada en la inferior.

Respecto a la conexión con la base de datos, se utilizarán conectores JDBC. JDBC es un API (Application Programming Interface) que define una librería estándar para el acceso a bases de datos. En el siguiente cuadro de código se puede observar un acceso a la base de datos a través de un conector JDBC.

```
global java.lang.String sql
global java.sql.Connection conn
global java.sql.Statement st
ResultSet  rst1 = null;
try {       // Cargamos el controlador JDBC
            Class.forName("org.hsqldb.jdbcDriver").newInstance();
    }
catch (Exception ex)
    {
            System.err.println("Se ha producido un error al cargar el controlador JDBC");
    }
Try
    {       // Se realiza la conexión JDBC
            conn = DriverManager.getConnection(url_db,"sa","");
            st  = conn.createStatement();
    }
catch (Exception ex)
    {
            System.out.println(ex);
    }
 try
    {       // Se ejecuta una sentencia SQL
            st.executeUpdate( "insert into almacen_mp (id_almacen,localizacion,esp_libre,estado,
            c_max) values ("+id+",'"+string2+"', "+num1_f+", true, "+num1_f+")");
    }
catch (Exception e)
    {
            System.out.println(e);
    }
 try
    {     // Se cierra la conexión JDBC
          st.executeUpdate("SHUTDOWN");
    }
catch (Exception e)
  {
             e.printStackTrace();
  }
```

Figura 10: Código correspondiente a un accesos a la base de datos HSQLDB mediante conectores JDBC

Aunque el código anterior resulta a la par simple y autoexplicativo merece la pena destacar que tanto las posibles dudas que pudieran surgir como otros aspectos de interés

(librerías, clases, variables, etc.) quedarán recogidos con mayor profundidad en capítulos posteriores de la memoria.

La elección de esta base de datos concreta no está sujeta a ningún criterio específico, sino que se fundamenta en el hecho de estar incluida dentro del paquete descargable para el servidor JBoss. A partir de diversas pruebas de uso, así como de su correspondiente testeo durante la ejecución del propio proceso general, se determinó que esta herramienta cumplía convenientemente con los principales requisitos perseguidos por este proyecto.

2.3. jBPM y JPDL

2.3.1. JBoss jBPM

Se conoce como JBoss jBPM a una plataforma de desarrollo de código abierto e implementada en Java que modela, gestiona y ejecuta procesos de negocio mediante el estándar BPMN 2.0. Este sistema, basado en una programación orientada a gráficos (GOA), representa un elemento clave para la implantación de los flujos de trabajo en entornos de negocio empresarial. Mediante la utilización de los servicios ofrecidos por esta plataforma se conseguirá implementar los distintos procesos de negocio coordinando, para ello, personas, servicios y aplicaciones.

A través de JBoss jBPM se permitirá cumplir con las necesidades actuales de las organizaciones empresariales, asegurando además su adaptabilidad frente a los requisitos tanto de los clientes como de los mercados. De forma adicional, la escalabilidad garantizada por su motor de procesos admitirá su utilización en toda clase de entornos, con total independencia de la complejidad inherente al mismo. El resultado final será una plataforma flexible, capaz de crear, coordinar y monitorizar procesos de negocio de forma ágil, flexible y fiable.

Los elementos principales incorporados por esta herramienta vendrán determinados por el editor de procesos y el motor de ejecución de procesos. En nuestro caso, para el editor se utilizará un diseñador instalado como componente adicional al software de desarrollo Eclipse IDE Indigo.

El motor de procesos utilizado se corresponde con el componente por defecto definido en la JBoss Enterprise SOA Platform para jBPM. Este motor se encarga de la interpretación y la

ejecución de los procesos definidos en la etapa de modelado. Para lograr su cometido, el motor de procesos también podrá hacer uso de diversas bases de datos asociadas así como de sus numerosos registros locales. A partir de ellos se podrán efectuar las peticiones y consultas pertinentes requeridas durante la ejecución.

Si bien se pretende obtener una breve visión de conjunto para el funcionamiento interno de jBPM, resulta de interés aclarar algunas de las funciones adicionales que este sistema nos ofrece. Entre estas funciones se destacan:

- ✓ Verificación del estado actual del sistema por medio de la validación de los comandos utilizados en la ejecución de una tarea específica.
- ✓ Autentificación de los usuarios del proceso, estableciendo además la accesibilidad controlada a las distintas tareas definidas en el flujo.
- ✓ Elección del camino correcto por medio de la evaluación de condiciones y reglas en tiempo de ejecución. Para esta evaluación se pueden incorporar mecanismos de toma de decisiones basados en el motor de reglas de jBPM, a través de la definición de archivos específicos o Drools.

Como cabe esperar, el motor de procesos representa un elemento determinante en la automatización de los flujos de trabajo. Este elemento tomará como entrada los distintos procesos definidos durante la etapa de modelado. Dichos elementos, desarrollados gráficamente por el diseñador instalado en Eclipse, establecerán el flujo secuencial de trabajo a partir del cual originará el orden estricto de ejecución.

Por este motivo, esta herramienta ofrece el puente de unión entre los modelos gráficos de negocio, y su correspondiente implementación software; o, desde otro punto de vista, entre los analistas de negocio y los desarrolladores de BPM. Este concepto, unido a otros comentados en capítulos anteriores, ponen de manifiesto la estrecha relación entre todos los actores implicados en la automatización de los procesos de negocio, ya sean los propios clientes, los administradores y empleados de la empresa, los analistas del negocio o los desarrolladores de software.

Una vez ejecutado el proceso de negocio, el motor será el encargado de determinar todas las conexiones necesarias para llevar a cabo las distintas tareas requeridas, ya sea con sus participantes humanos o con servicios y aplicaciones externas. A modo de recordatorio,

jBPM distinguirá entre dos tipos de actividades dependiendo de su modo de ejecución. Atendiendo a esta clasificación podremos encontrar actividades automáticas donde la ejecución se realiza de forma directa y automática por el software de aplicación; y actividades con tiempos de espera, donde su ejecución dependerá del resultado obtenido de servicios automáticos externos o de participantes humanos. Esta variedad de actividades, así como un desglose interno de la plataforma jBPM, se representa de forma esquemática en la siguiente figura.

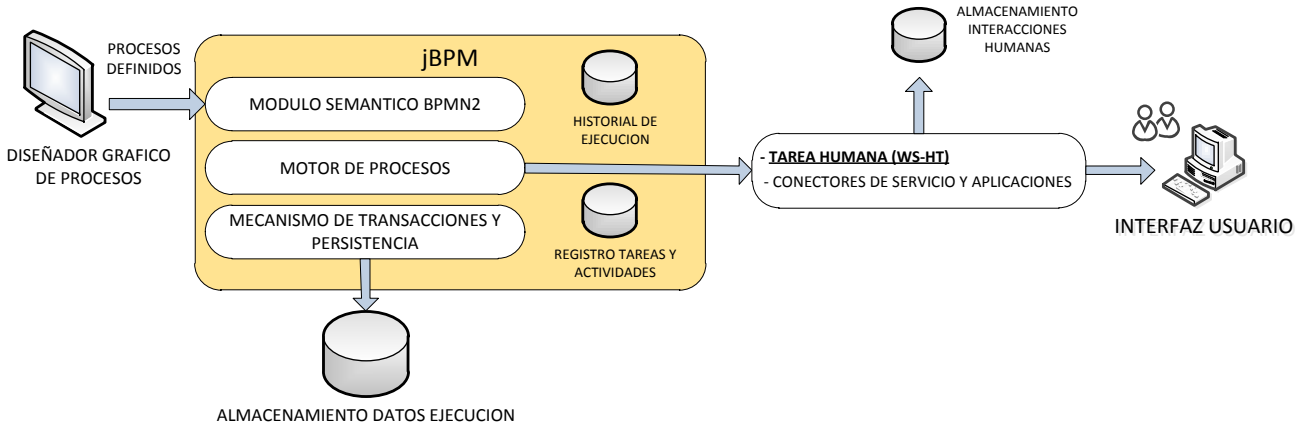


Figura 11: Esquema interno de la plataforma jBPM

Aunque jBPM como herramienta permite la conexión con servicios y aplicaciones externas, este proyecto se centra principalmente en la interacción coordinada con actores humanos. Para esta interacción, jBPM utiliza un estándar denominado Web Service Human Task (WS-HT). Mediante este componente de establecen, mantienen y administran los ciclos de vida asociados a la tareas de interacción humana. A través de este estándar no sólo se definen los posibles estados alcanzables en su ciclo de vida, sino también las estructuras de datos manejables por ellas.

Por otro lado, en el esquema anterior también está recogida una subdivisión interna de varios componentes dentro del conjunto jBPM. Estos elementos se encuentran agrupados según la funcionalidad específica de cada uno de ellos. Dentro de esta agrupación se distinguen tipos esenciales:

- ✓ Módulo semántico BPMN2: este componente será el encargado de interpretar los procesos definidos en el programa de modelado gráfico. Para ello, el

proceso deberá cumplir con las especificaciones establecidas en el estándar de notación BPMN2.

- ✓ Núcleo del motor de procesos de negocio: este elemento se ocupa de crear instancias de los procesos interpretados por el módulo semántico y ejecutar sus definiciones. Además, el motor de jBPM proporciona un conjunto de APIs de Java para su integración con otras aplicaciones.
- ✓ Mecanismos de transacción y persistencia: este módulo es el encargado de almacenar y mantener el estado de cualquier proceso de negocio ejecutado en el sistema.
- ✓ Historial de ejecución: se corresponde con un módulo de memoria dedicado al almacenamiento de toda información adicional relativa a la ejecución de los procesos de negocio.

Dentro del motor de procesos jBPM se establece también la posibilidad de ejecución múltiple de procesos. Esta característica, que permite tener activos varios procesos a la vez de forma concurrente, se plantea como una interesante ventaja a tener en cuenta por los desarrolladores atraídos por esta plataforma. A partir de esta ejecución múltiple se conseguirá establecer instancias independientes de los procesos para una ejecución autónoma y simultánea. Debido a la importancia operacional de esta característica, este aspecto será discutido con mayor profundidad en posteriores apartados de la memoria.

2.3.2. JPDL (jBPM Process Definition Languaje)

Como se comentó en el capítulo anterior, la definición de procesos se realiza mediante un editor gráfico instalado como componente del software de desarrollo Eclipse. A través de este componente se generarán los distintos archivos correspondientes al proceso en cuestión sirviendo, posteriormente, de referencia en ejecución al motor central de jBPM. Aunque esta aclaración es correcta, no resulta del todo completa. Para completarla es necesario puntualizar que el motor de proceso jBPM no entiende de entornos gráficos, siendo necesaria una correspondencia entre el modelo gráfico y el correspondiente lenguaje de programación empleado.

Bajo esta premisa nace el concepto de JPDL. Estas siglas, que hacen referencia a jBPM Process Definition Language, se asocian a una notación para el lenguaje de definición de

procesos de acuerdo con un XML Schema. Este lenguaje, equivalente por tanto a XML, representa específicamente lo editado de forma gráfica mediante el componente de diseño gráfico de Eclipse. Si bien existe una relación casi total entre lo editado gráficamente en Eclipse y lo definido en JPDL, existen también ciertas entidades que no pueden ser representadas en el componente gráfico aunque sí definidas por JPDL. Al margen de ello, la relación entre ambos métodos de definición, ya sea el método gráfico o el programado, resulta bastante directa y completa. De este modo, se puede apreciar que el editor gráfico no sólo es capaz de leer e interpretar el fichero definido en JPDL, sino que también puede generar de forma automática los cambios editados gráficamente en el correspondiente lenguaje de definición.

A continuación se añade el esquema genérico de definición de procesos estableciendo, para ello, la correspondencia entre el modelado gráfico y la definición en JPDL.

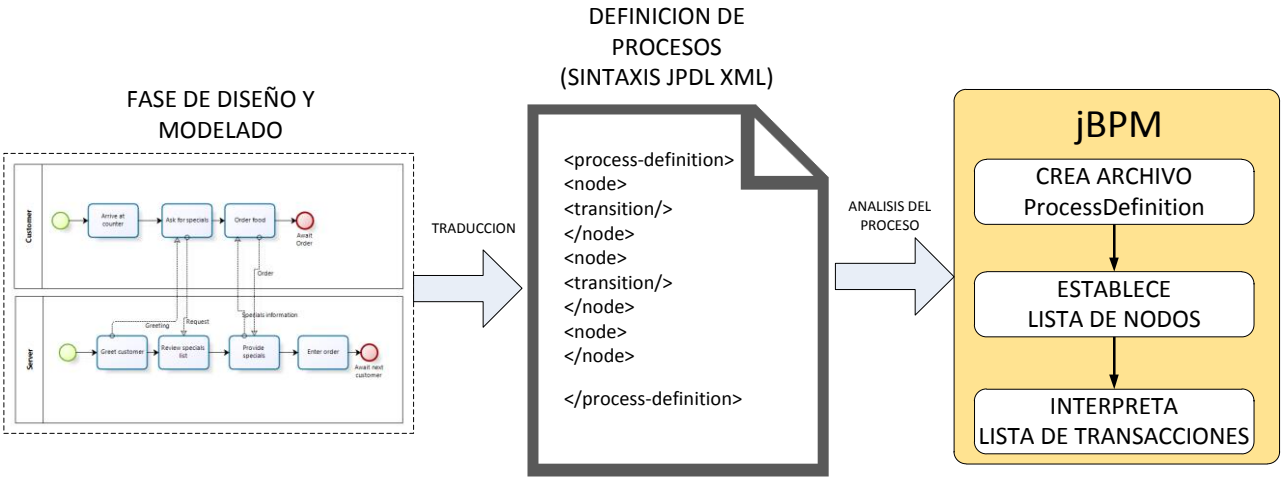


Figura 12: Asociación entre los métodos de definición de procesos y el comportamiento interno de jBPM

Como cabe esperar, en la definición de procesos en JPDL se hará uso de las correspondientes etiquetas XML. Todas estas etiquetas permanecen recogidas en un esquema XML fácilmente accesible en la red. Para localizar cualquier información acerca de este esquema, sólo es necesario navegar a través de páginas como la siguiente: http://docs.jboss.com/jbpm/v4/schemadocs/index.html.

3. Definición del entorno de trabajo

3.1. Descripción del proceso de negocio

El proceso productivo para la elaboración de vino ha representado una ciencia en sí misma sujeta a continuas variaciones en métodos y técnicas a lo largo de toda su historia. Mediante la experimentación, el conocimiento intrínseco del producto y de la materia prima, y la evolución tecnológica se han logrado desarrollar procesos altamente eficientes y mejorados. Los métodos actuales de elaboración y producción plasman claramente esta evolución, representando además la unión entre conceptos modernos y tradicionales.

En el transcurso de esta sección se realizará el análisis cualitativo de un proceso vinícola genérico, atendiendo a las distintas etapas que lo componen. Con objeto de mejorar la comprensión del proceso simulado en este proyecto, se realizará además el seguimiento conceptual del producto, partiendo desde su recolección en los viñedos, pasando por los procesos intermedios de fermentado y molienda, hasta su embotellado final y distribución.

3.2. Fases del proceso de elaboración de vino

La primera etapa que inicia el proceso de elaboración de vino es la **vendimia**. Esta etapa, comúnmente realizada entre los meses de julio y octubre, establece la temporada de recolección de las uvas de vino. La elección del momento óptimo de cosecha de la uva vendrá determinada por una relación entre su grado de maduración y el tipo de vino que se desea producir. La vendimia se puede llevar a cabo según dos métodos distintos:

- Recolección manual: se dedica principalmente a la producción de vino de alta calidad o ciertas variedades específicas como son los vinos espumosos. Permite la selección del producto así como mejorar sustancialmente su conservación (la uva se conserva entera, no se estruja ni mostea). Por todo ello, resulta el método más habitual y extendido entre los productores de uva.
- Recolección mecanizada: esta recolección reduce significativamente los costes limitando, en gran medida, la calidad final del producto. Para utilizar este tipo de recolección será necesario que el viñedo se distribuya en espaldera.

Figura 13: Recogida de la vendimia

Como puede observarse, la elección de la técnica de recolección no debe tomarse a la ligera, ya que repercutirá directamente en las características intrínsecas a la materia prima recogida.

Una vez obtenido el flujo inicial de uva procedente de los viñedos, será necesario trasladar esta materia prima a las plantas de producción ubicadas en la bodega. Para el **transporte de la uva**, será necesario garantizar unas estrictas condiciones de calidad e higiene evitando, de este modo, cualquier alteración de la materia prima durante su traslado. Este transporte, que deberá ser realizado a la mayor brevedad posible, representa otro de los puntos clave para asegurar la calidad final del producto terminado. Por un lado, el traslado debe evitar la fermentación prematura de la uva. Este efecto estará principalmente causado por dos motivos principales: el propio aplastamiento de la materia prima en los contenedores de transporte y las condiciones ambientales existentes durante el trayecto. Con el fin de minimizar este desagradable efecto, se suele recurrir al uso de pequeños contenedores de plástico convenientemente apilados y aireados.

Ya transportada la uva a la bodega se realiza el proceso correspondiente a la **recepción y el pesaje**. Como la recepción de mercancías no es constante, este proceso hay que tomarlo con cuidado. Posibles fallos en la organización durante esta recepción pueden ocasionar serias acumulaciones a la entrada de la bodega y degradarse así la calidad de la materia prima durante la espera. Por otro lado, a fin de asegurar la cantidad de uva aceptada, se pesa el vehículo de carga por medio de básculas automáticas. Después del pesaje y, por supuesto, antes de aceptarlo en los almacenes dedicados a la materia prima, se realiza una estricta **inspección** del producto. Entre las prácticas más comunes incluidas dentro de esta inspección

se encuentran el control visual de la uva para garantizar su color, integridad, etc.; el control higiénico del medio de transporte y el control higiénico de los elementos encargados de la recepción de mercancía. De forma adicional, se pueden realizar algunos análisis específicos del producto entrante como la toma de muestras, los controles de acidez o las mediciones del grado alcohólico potencial, entre otros. Las partidas de uva que no cumplan con los requisitos mínimos establecidos en esta fase de inspección serán, por tanto, desechadas.

Figura 14: Recepción de materia prima a la entrada de la bodega

Pasado el proceso de inspección y contabilizada la cantidad de producto entrante en la bodega, se procede a la descarga de la uva. Este proceso, llevado a cabo normalmente mediante el uso de tolvas y cintas transportadoras, traslada la materia prima desde la zona de recepción a su futura ubicación en el interior de la bodega. Como cabe esperar, este proceso se realiza de forma totalmente automática de forma que los operarios de bodega sólo ejercen un papel de asistencia y supervisión.

Aunque esta primera etapa se puede considerar ajena e independiente al proceso interno de producción de vino, su planificación plantea el primer reto a solventar en cuanto a la optimización general del proceso completo. A raíz de ello, fallos durante esta fase inicial podrían repercutir muy negativamente en el desarrollo productivo de la bodega. De este modo se podrían ocasionar imprevistos desde bloqueos en la línea producción por la falta de recursos internos en la bodega hasta periodos de espera por exceso de materia prima en el almacén de entrada.

Dejando a un lado los problemas anteriores y volviendo a las tareas específicas del proceso de producción, la siguiente etapa da comienzo a la fase propia de elaboración del producto. En primer lugar es necesario completar un proceso de preparación antes de comenzar con el fermentado de los lotes entrantes en la bodega. Esta nueva tarea, que gira en

torno a un concepto conocido como el **pie de cuba**, establece la preselección de las primeras partidas de uva de vendimia. A través de esta preselección, que generalmente oscila entre los 3000 y los 5000 kilogramos, se llena un depósito o cuba de acero inoxidable. Una vez depositada la uva se realiza el inicio de la fermentación bajo una atmósfera controlada de presión, temperatura y humedad. El objetivo perseguido por esta fermentación prematura será establecer una población inicial de levaduras que acelere el proceso de fermentación en los siguientes lotes de uva. Como cabe esperar, para realizar todo el proceso de preparación de pie de cuba se utilizará el mismo camino establecido para las posteriores partidas entrantes a la bodega.

Tras esta aclaración y volviendo al almacén de materias primas, la uva es traslada hacia las cubas donde realizará su primera fermentación. Durante este traslado se añade dióxido de azufre al producto. Por medio de este proceso, denominado **sulfitado**, se busca mejorar las características de la uva frente a ciertos agentes externos. Entre las características principales, los efectos más determinantes son el antioxidante para limitar la oxidación; el antioxidásico para inhibir las polifenoloxidasas y el antiséptico frente a la acción de las bacterias. La correcta dosificación de este producto resulta de vital importancia ya que una dosis inferior dejaría al vino desprotegido, mientras que una superior ocasionaría serios problemas en su fermentación.

Una vez sulfitado el producto se procede al a la fase de **despalillado**. En esta etapa se buscará la separación entre el raspón y el grano de la uva. La maquinaría encargada de esta tarea consta de un cilindro perforado colocado horizontalmente y un eje dotado de paletas. Los correspondientes racimos de uva son transportados hasta una tolva que los deja caer en el interior del cilindro. Mediante el giro continuo de la maquinaria se realizará la esperada separación entre el raspón y el grano expulsándose, este último, gracias a los orificios perforados del cilindro.

Los granos de uva obtenidos en este proceso pasarán directamente a una estrujadora de acero inoxidable que realizará el primer prensado del producto. Este **prensado** estará caracterizado por sólo desprender la pulpa de la uva ya que, en ningún caso, se buscará romper ni dilacerar sus partes sólidas.

Ya despalillada y estrujada la uva procedente de la vendimia se inicia el proceso de **encubado** y maceración. Este proceso, realizado mediante tuberías y bombas automáticas, trasladarán las pastas obtenidas tras el estrujado a las cubas inoxidables dedicadas a su reposo

e inicio de fermentación. Las características del traslado de las pastas deberán ser muy restrictivas puesto que repercutirá de forma significativa en la calidad final del producto. Como se comentó anteriormente, para favorecer el inicio de la fermentación alcohólica se adherirá parte del pie de cuba cosechado en etapas anteriores. Las cubas utilizadas durante esta etapa se corresponderán generalmente con depósitos cilíndricos verticales de acero inoxidable dotado de elementos automáticos para la refrigeración interna de su contenido. Por otro lado, también contarán con mecanismo automáticos para el llenado y vaciado, así como elementos interiores para la agitación de las pastas.

Figura 15: Periodo de fermentación y reposado en cubas

Durante el periodo de encubado, que suele durar en torno a 10-15 días, no sólo se realiza parte de la fermentación alcohólica del vino, sino que se determinan además gran parte de las características organolépticas finales del producto.

Finalizado el tiempo de encubado se realiza el **descubre** del producto. Esta nueva etapa marcará el fin del periodo de maceración, vaciando las cubas correspondientes y extrayendo así el líquido contenido en su interior. El enólogo de la bodega será el encargado de determinar el momento óptimo del descubre así como establecer la futura calidad del producto. Para ello se deberá realizar un **análisis** en el laboratorio a fin de determinar algunas características como la densidad, el grado de alcohol, etc. Por otro lado y con objeto de establecer algunas de las características organolépticas del producto, también será necesario llevar a cabo una cata. A partir de los todos los resultados obtenidos en esta etapa se establecerá el destino final de la partida de vino correspondiente ya sea para vino joven, vino crianza o vino reserva.

Antes de la extracción del vino de las cubas se realizará la **separación de las pastas sobrantes** suspendidas en el depósito anterior. Estas pastas pasarán a su vez a una **segunda etapa de prensado** donde, en primer lugar, se les aplicará un presión reducida. El resultado de este estrujado producirá vino de gran calidad que podrá ser adherido a la partida principal de vino. Tras este leve estrujado se aplicarán presiones mayores que dará como resultado un vino degradado de peor calidad.

Volviendo a la partida principal y ya incluido el vino procedente de la segunda prensada, se procede a las fases de **clarificación y filtrado**. Este proceso constará de un periodo previo, dedicado a la sedimentación natural del producto, así como de una etapa posterior dedicada a la clarificación por filtrado. Para este proceso, generalmente, se suelen utilizar técnicas basadas en el uso de tierras diatomeas capaces de separar las partículas no deseadas del vino. Una vez superada esta etapa, algunas bodegas optan por un periodo de estabilización por frío. Esta clase de técnicas busca eliminar algunas sustancias sensibles a las bajas temperaturas tales como cristales, tartratos, etc. Con el fin de efectuar esta estabilización se realiza el enfriamiento del producto a temperaturas cercanas a su punto de congelación provocando, de este modo, la precipitación por cristalización del tartrato. Tras este proceso se filtra nuevamente el vino eliminando así las posibles partículas aparecidas durante la fase de estabilización.

Pasado todo el periodo de reposo en cuba y superada además todas las fases de clarificación y filtrado, se procede al **llenado de las barricas** de roble para iniciar así el largo periodo de crianza. Esta etapa, sólo alcanzable para vinos que pretendan la denominación de crianza, reserva o gran reserva, marca el comienzo de la maduración del vino. La duración de este proceso dependerá directamente de la calidad inherente al vino, pudiendo variar entre seis y doce meses para los vinos de crianza; más de un año para los vinos de reserva o más de dos años para los vinos de gran reserva. El enólogo de la bodega, mediante un análisis sensorial del vino, deberá ser el encargado de determinar el momento óptimo de finalización de la crianza. Durante este periodo será de vital importancia controlar algunas características ambientales del entorno de crianza como la temperatura o la humedad, entre otras.

Figura 16: Periodo de crianza en barrica

Tras la **crianza en barrica**, el grupo de enólogos de la bodega deberá realizar un exhaustivo análisis de calidad en busca de enfermedades o alteraciones en sus características. Esta adulteración del producto, que abarca desde defectos en el sabor o el olor hasta fallos en el color, la pureza o la textura, pueden ser ocasionadas por múltiples causas. A su vez, algunas de ellas pueden ser disimuladas e incluso corregidas mediante el uso de técnicas o medidas correctoras. En estas situaciones, la rápida toma de decisiones del enólogo podrá salvar una partida defectuosa, limitando así las posibles pérdidas de la bodega.

Una vez conseguida la acreditación de calidad del producto tras su periodo de crianza, el siguiente paso será proceder a su **embotellado**. En el caso concreto de los vinos jóvenes que carecen de este tiempo de crianza, este proceso se realizará justo después de su correspondiente estabilización en frío. Para realizar correctamente el embotellado del producto se deberán controlar especialmente las condiciones higiénicas de las instalaciones, ya que éstas podrían alterar las características finales del vino elaborado. Para realizar esta tarea generalmente se hace uso de embotelladoras automáticas que permanecen asistidas por los propios operarios de la bodega. De este modo, al automatizar este proceso se asegura no sólo unas condiciones estables del producto sino también la estricta repetitividad de la operación. El embotellado del vino engloba dos tareas principales, el llenado y el taponado. Éste último se corresponde con un proceso delicado donde un tapón de corcho es introducido de forma hermética en el gollete de la botella, dejando una cámara de aire ente la superficie del líquido y el tapón.


Figura 17: Proceso de embotellado del vino y envejecimiento en bodega

La siguiente operación en el proceso de elaboración de vino se corresponde con el **envejecimiento en botella**. Este proceso generalmente se practica a vinos de reserva y gran reserva, aunque existen partidas de vino de crianza que la también admiten. A lo largo de este proceso de envejecimiento, el vino sufre diversas modificaciones que permiten completar así su afinamiento. Durante este periodo, las botellas son almacenadas en unas naves dedicadas exclusivamente al envejecimiento. Estas naves son diseñadas de forma que se aporten unas condiciones óptimas de humedad, luminosidad y temperatura. Normalmente, esta operación se realiza disponiendo las botellas en jaulones horizontales de acero inoxidable. La duración de este periodo varía enormemente dependiendo tanto de la calidad del vino que se está produciendo, como de los requisitos específicos establecidos por la bodega.

Tras el correspondiente periodo de maduración por el envejecimiento y un nuevo análisis de calidad del producto, las botellas de vino quedarán listas para su distribución. Sin embargo, éstas deben realizar antes el **encapsulado** de la botella. En el caso particular de los vinos jóvenes esta fase a los vinos jóvenes les llega inmediatamente después del embotellado. Para cumplimentar este proceso resulta imprescindible asegurar una serie de necesidades que se presentan a continuación. Entre las más representativas se encuentran:

- Aplicar el sello fiscal a la botella.
- Garantizar el correcto taponado del producto. En este apartado se permite cierta personalización del producto.
- Se debe asegurar la fácil apertura de la botella, así como la limpieza del tapón y del gollete.

Finalizada la etapa anterior, ya sólo queda el **etiquetado** y **empaquetado** final del producto. Esta fase del proceso marca el final del flujo de elaboración y se realiza justo antes

de la salida del producto al mercado. La fase de etiquetado suele desarrollarse de forma automática utilizando, para ello, cintas de transporte, carruseles giratorios, paletas, etc. Tras acabar el etiquetado se inicia la operación de empaquetado donde se agrupan las distintas botellas en cajas donde quedarán precintadas y preparadas para su venta. Generalmente, esta tarea será realizada de forma manual por los operarios de la bodega.

Una vez concluido todo el proceso de elaboración de vino se procede a la etapa de **almacenamiento** y **distribución**. Las distintas cajas precintadas en el proceso anterior deberán pasar al pertinente almacén dedicado al producto elaborado. Este lugar, convenientemente diseñado para este fin específico, deberá a su vez establecer unas condiciones adecuadas a las necesidades de almacenamiento del producto, garantizando así su conservación hasta el momento de su venta. Para la distribución y venta será necesaria la intervención de departamentos ajenos al proceso de elaboración como puede ser el departamento de marketing; así como una adecuada red de transporte y distribución que pueda hacer llegar el producto final a los clientes.

4. Desarrollo del proyecto

4.1. Esquema conceptual

A fin de mejorar la comprensión de este apartado se ha optado por realizar una planificación previa de la metódica perseguida durante esta implementación. Por medio de este estudio preliminar se pretende mejorar la percepción de todo el conjunto y asimilar mejor las distintas etapas contempladas en el desarrollo.

Para llevar a cabo el presente proyecto se han tenido en consideración una serie de factores que han desembocado finalmente en el modelo conceptual alcanzado por esta solución. En esta primera etapa se presentará este modelo, así como algunos aspectos interesantes a tener en cuenta durante la evaluación del diseño implementado.

El concepto interpretado del proceso real permanece representado en el esquema expuesto a continuación.

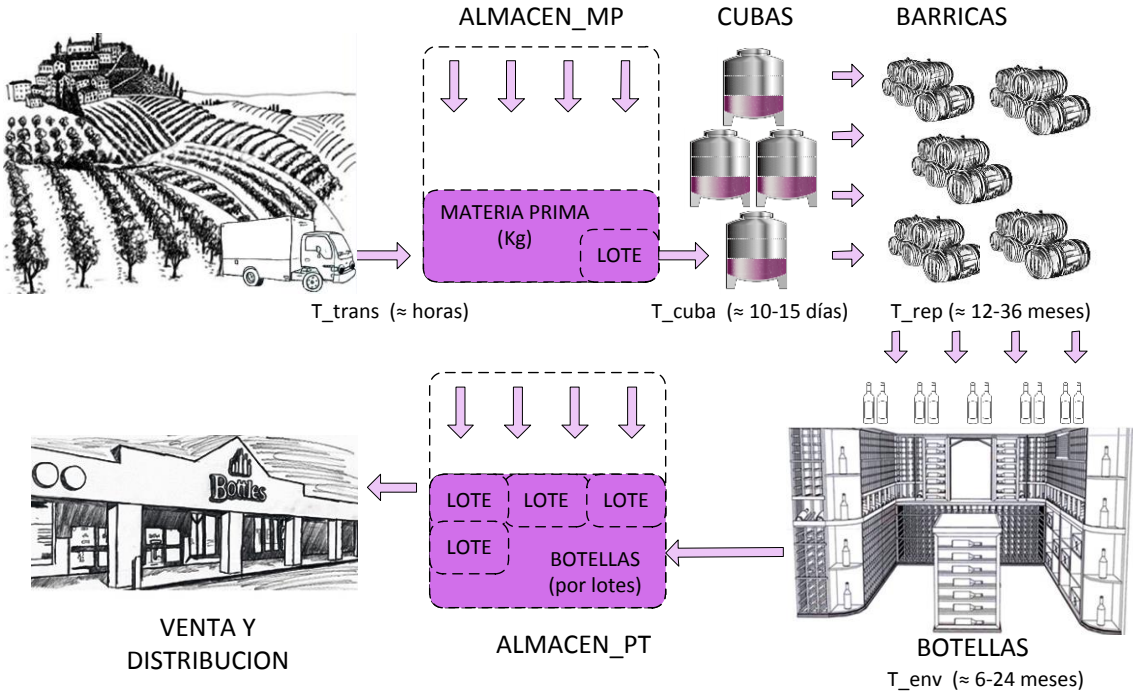


Figura 18: Esquema conceptual de elaboración de vino

Como puede observarse existe una asociación directa entre los distintos nombres establecidos en el diagrama y los correspondientes objetos o entradas en las tablas especificadas en la base de datos del sistema. Dentro de este modelo se aglutina tanto la

interpretación del proceso real descrita al inicio del capítulo como las distintas consideraciones, restricciones o simplificaciones establecidas para acotar la complejidad de la solución. Bajo este marco, representado esquemáticamente en la figura anterior, se pretende extender la percepción global del proceso acercándola, aún más, a la implementación final concebida en este proyecto.

Al igual que el proceso real, todo comienza en los viñedos. En esta situación se localiza el punto inicial del proceso donde con cada ejecución se lanzará una nueva petición de envío de uva hacia el almacén correspondiente. Esta petición quedará recogida en un elemento tipo pedido que quedará almacenado en la correspondiente base de datos del sistema.

Tras el tiempo establecido para el transporte de la uva, los distintos camiones irán llegando a la bodega y, tras los correspondientes controles, descargará la mercancía en el almacén establecido para la materia prima entrante. Este almacén puede considerarse como un elemento desbordable que irá aumentando su contenido con cada nueva iteración del proceso. Mientras no se cumplan las condiciones establecidas, todo proceso ejecutado morirá en este punto de su simulación. Entre las condiciones necesarias para aceptar la continuación del proceso hacia sus siguientes etapas, será necesario cumplir con una cantidad mínima de materia prima en el almacén. Esta cantidad, perfectamente configurable en los pasos previos a la configuración de la bodega, queda estipulada por el tamaño recogido para la unidad conocida como LOTE. Este lote representará la unidad mínima de trabajo en la bodega y quedará representada, entre otras cosas, por una cantidad fija de uva. Para la elección de este valor se ha considerado que debe ser igual a la capacidad máxima permitida por el depósito encargado de la maceración del producto, la cuba.

Tras cumplir con la restricción asociada a la cantidad de materia prima en el almacén, deberán además superarse otras condiciones adicionales generalmente relacionadas con la situación exacta de la bodega. Entre estas condiciones, otra de las más evidentes será la posibilidad de alcanzar una cuba libre donde poder realizar la fermentación establecida para el producto.

Una vez superadas todas las condiciones anteriores se procede a la creación de un nuevo elemento lote adscribiendo una entrada nueva en la tabla correspondiente. Este elemento pasará por una serie de etapas previas antes de quedar reposado en la cuba asociada antes de su creación. Durante el periodo de maceración se establecerán las primeras características organolépticas diferenciadoras del vino de forma que, tras su descubre y

análisis, el enólogo marcará la nueva línea a seguir por el lote. Este actor, que cobra una importancia decisiva en la mayor parte de los procesos en que interviene, decidirá tras el descubre la calidad final del vino. De este modo, a partir de este preciso momento, se decidirá sobre el futuro camino a seguir por el lote, así como sobre los tiempos de espera estipulados para ello.

Según las consideraciones establecidas en el inicio de este apartado, el enólogo decidirá sobre si el vino será joven, crianza o reserva. Esta elección se basará principalmente en una relación estricta entre el análisis realizado al producto y los requisitos establecidos en la bodega para establecer cada una de las calidades. Como cabe esperar, los tiempos de crianza y envejecimiento serán radicalmente distintos según la calidad específica del producto a elaborar.

Como concepto particular relacionado con la elección de la calidad del producto, merece la pena destacar una excepción en particular. Esta excepción, que se encuentra estrechamente ligada con la disponibilidad operativa de los recursos de la bodega, admitirá la posibilidad de degradar la calidad de un vino ante la falta de espacio, ya sea para su estancia en barrica o en la bodega para su envejecimiento. De este modo, y gracias a la característica específica de los vinos jóvenes de no admitir ni reposado en barrica ni envejecimiento en botella, cualquier posible inconveniente asociado a la reserva de recursos en las siguientes etapas del proceso podrá verse solventado mediante la degradación del vino a la calidad de vino joven. Esta cualidad simplifica sustancialmente la complejidad de la solución planteada ya que plantea la posibilidad de utilizar un flujo continuo y accesible, a modo de escape, ante posibles cuellos de botella aparecidos durante la producción. Tras esta aclaración, cabe destacar que la toma de decisiones en este punto del proceso será crítica puesto que cualquier fallo en su programación podrá inferir graves daños a la planificación general de la producción en la bodega.

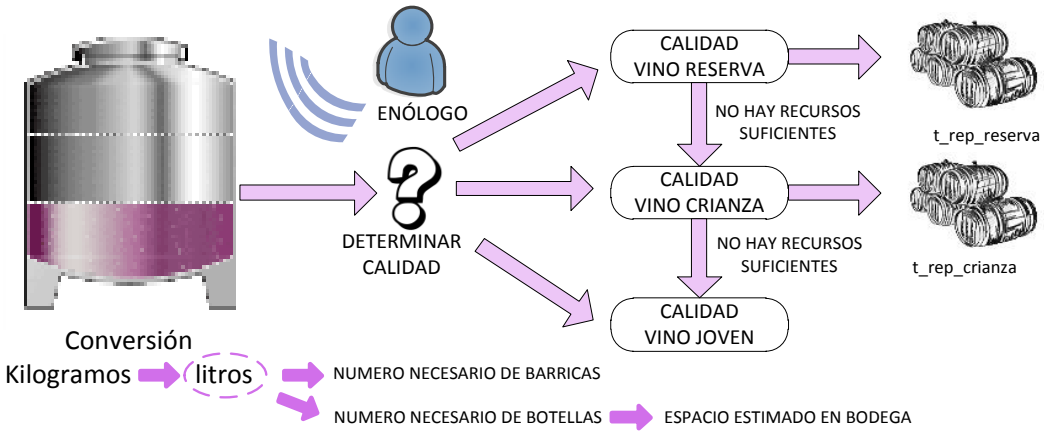


Figura 19: Representación conceptual de la toma de decisión durante el descubre y análisis del vino

En la figura anterior puede observarse la representación esquemática de la toma de decisiones llevada a cabo durante el proceso de descubre de la cuba. Como cabe esperar, esta toma de decisiones vendrá respaldada por una serie de estrictos controles de calidad realizados sobre el producto.

Por otro lado, para la planificación de los recursos de la bodega se ha contemplado la utilización de una serie de testigos característicos a los distintos elementos consumibles del proceso. Tal y como se observó en el apartado correspondiente a la base de datos del sistema, existirán varios elementos que actuarán a modo de recursos finitos del proceso. Estos elementos, ya presentados en secciones anteriores de la memoria, están asociados concretamente con las tablas BARRICA y BOTELLLAS y definidos por las distintas entradas contenidas en ellas. Ambos elementos permanecen fijados como entradas independientes para las correspondientes tablas alojadas en la base de datos del sistema, permitiendo así facilitar la gestión autónoma de los recursos de la bodega y favorecer además la toma de decisiones necesaria en todo el proceso.

Aunque, como cabe esperar, cada elemento barrica tiene correspondencia directa con una barrica física de la bodega, el tratamiento de los elementos tipo botella resulta bastante distinto. Por un lado, el concepto del que parte este elemento no está asociado a una botella real en sí, sino que hace referencia a su espacio figurado en la bodega especificada para el envejecimiento del vino. Por otro lado, la posibilidad de tratar estos elementos de forma individualizada no aporta ni el rendimiento ni la optimización operativa esperada por la implementación de este proyecto. A raíz de ello, la decisión final adoptada permite agrupar una cantidad fijada de espacio para envejecimiento a un conjunto previamente establecido de

botellas. Este conjunto de botellas o, más propiamente dicho, el espacio en la bodega donde cabe el número fijado de botellas para su envejecimiento, será conocido como el elemento botella. El número de botellas abarcable por este elemento será previamente establecido durante la etapa inicial de configuración del sistema. Para determinar este número se podrá o bien asociar con alguna cantidad operativa real manejable por los operarios de la bodega o, si no, establecer el conjunto que el gestor del proceso crea necesario.

Como cabe esperar, cada uno de estos elementos representará un factor determinante a tener en cuenta en la toma de decisiones asociada a la planificación de la bodega y, por ello, será explicada a continuación con mayor profundidad.

El primer factor de decisión que aparece en este momento del proceso viene asociado por los litros de vino obtenidos tras la maceración. Esta cantidad, que puede preverse mediante la utilización de un factor de conversión almacenado bajo el nombre de relación, marcará el número de barricas y botellas a reservar en las siguientes etapas del proceso. Una vez establecida la cantidad específica de vino obtenida de la cuba, y conociendo la calidad intrínseca del producto, ya se encuentra en condiciones de realizar la correspondiente reserva de recursos. Para llevarlo a cabo se deberá realizar un barrido de la base de datos en busca de los recursos a reservar. Una vez localizados, éstos quedarán marcados mediante la asociación del lote en cuestión con el campo especificado como id_lote_siguiente. Tras realizar este proceso, el recurso permanecerá reservado de forma que, una vez complete las tareas intermedias, pueda consumirlo sin espera alguna.

De forma adicional, esta técnica también plantea la posibilidad de utilizar un recurso mientras está reservado. Este concepto, que permite incrementar notablemente su rendimiento operativo, se basa en el uso de otro de los campos contenidos en las tablas descritas como BARRICA y BOTELLAS. Dichos campos, conocidos como id_lote_actual, establecen, como su propio nombre indica, el lote que utiliza el recurso en un momento específico de la ejecución. Mediante la utilización conjunta de ambos campos se consigue adoptar una solución potente sólo abarcable mediante el uso de testigos.

Estos testigos marcarán los casos específicos de reserva para cada una de las calidades de vino y asegurarán así la consistencia general de la ejecución del proceso. Todo ello permanece representado de forma esquemática en la siguiente figura, mostrando las distintas posibilidades abarcables en este punto de la ejecución.

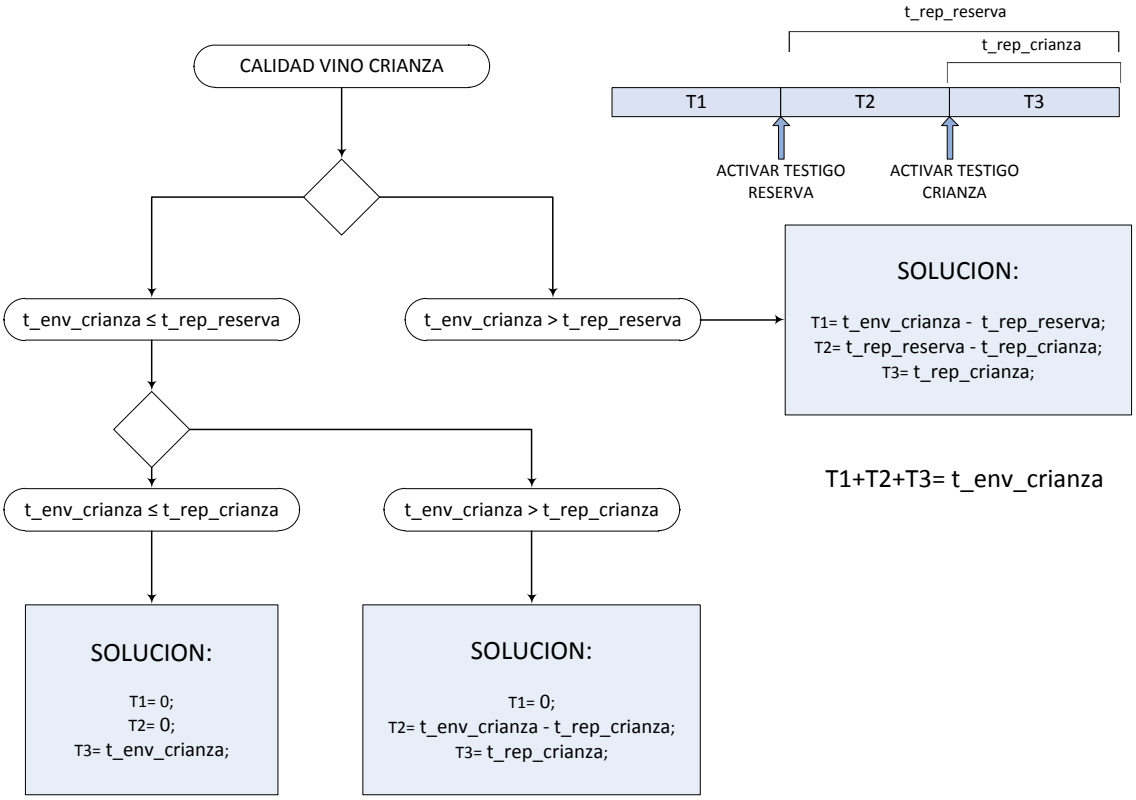


Figura 20: Diagrama de activación de testigos para vino crianza

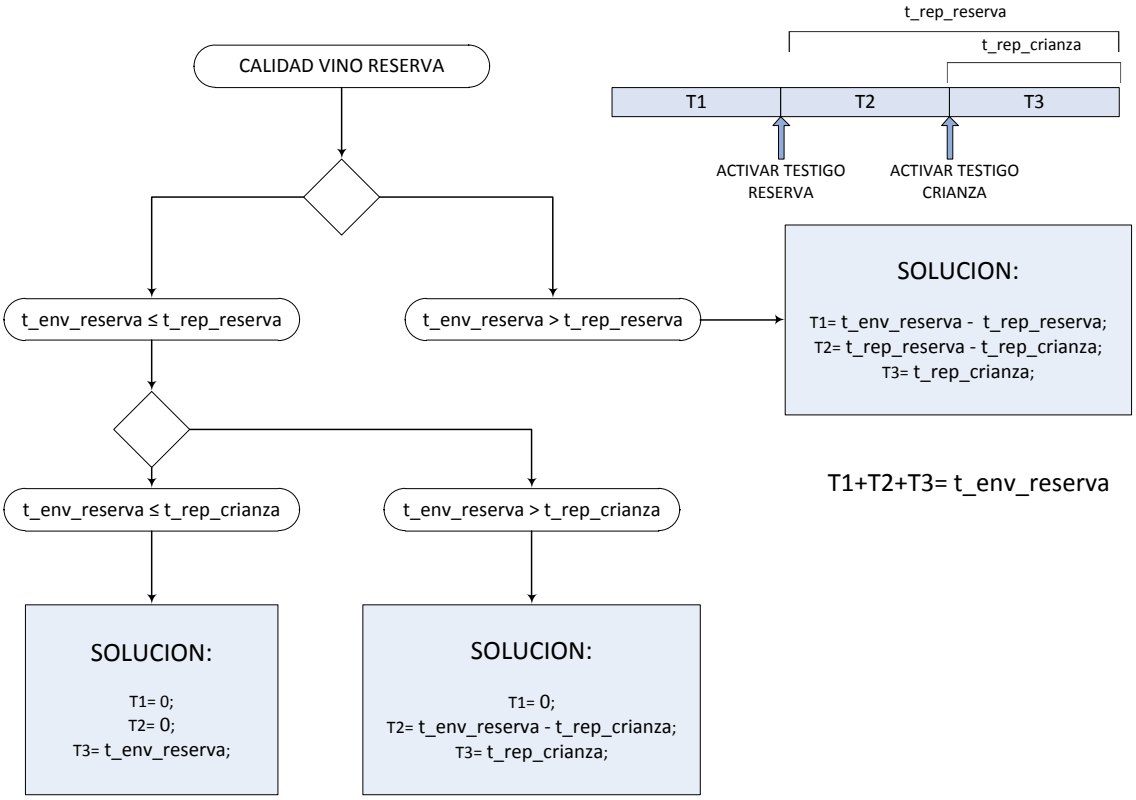


Figura 21: Diagrama de activación de testigos para vino reserva

En ambos esquemas se puede observar los tiempos específicos para para la activación de los testigos asociados a los distintos elementos tipo BOTELLAS del proceso. Esta asignación resulta completamente equivalente a la correspondiente con los elementos de tipo BARRICAS sólo que, en ese caso, los tiempos contemplados estarán asociados al periodo de maceración en cuba.

Para aclarar con mayor profundidad la toma de decisiones llevada a cabo tras finalizar el descubre y el análisis del vino, se representa el siguiente diagrama de flujo. En él se establecen todas las posibilidades asociadas a esta toma de decisión tras el correspondiente análisis, especificando además algunas de las acciones a realizar en relación con el camino establecido.

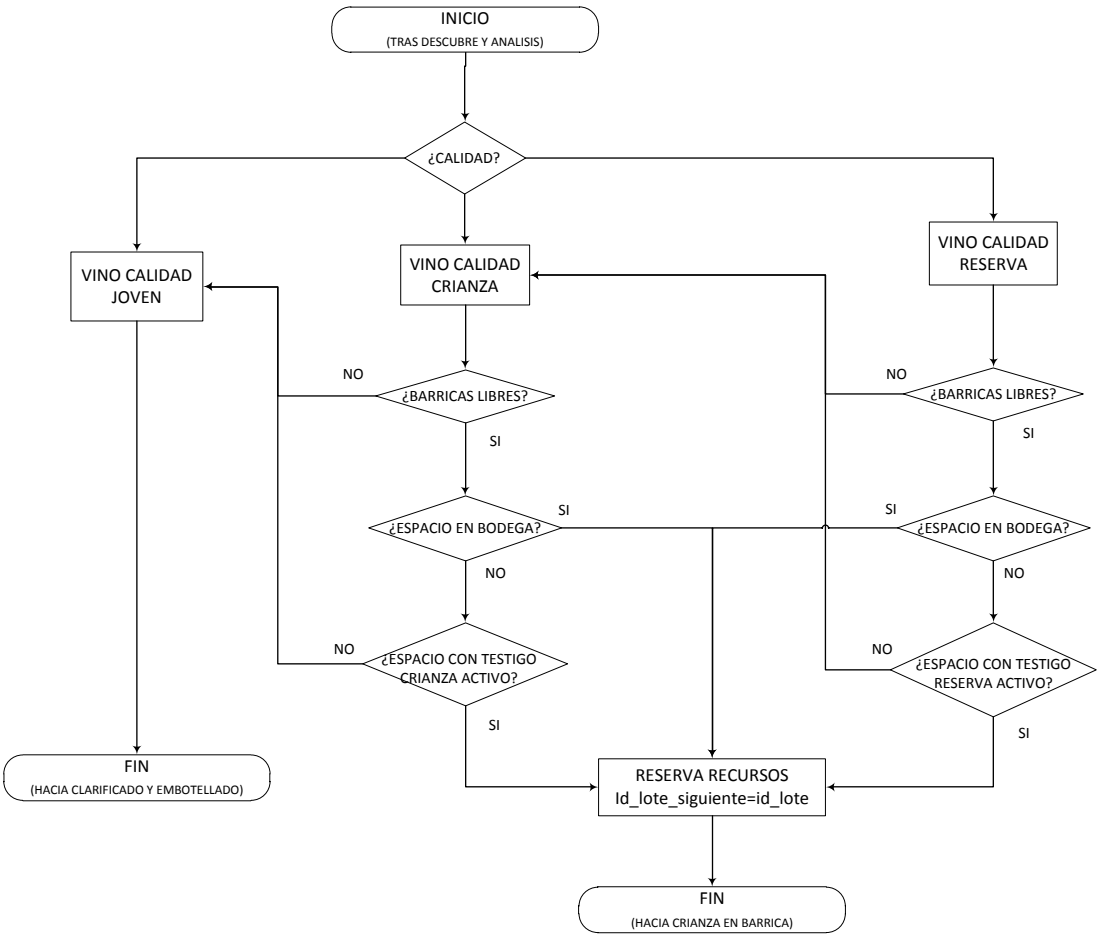


Figura 22: Diagrama de flujo para la toma de decisiones sobre la calidad del vino

Una vez determinados los siguientes pasos a seguir por el lote y asignados los recursos necesarios para ello, se procede a continuar con la ejecución secuencial del proceso. Esta

ejecución quedará completamente marcada por el conjunto de tiempos y tareas especificadas para la calidad específica del lote en ejecución. Por otro lado, la reserva de recursos permitirá garantizar la integridad del proceso, evitando paradas o bucles de inactividad. Tras la correspondiente toma de decisiones comentada anteriormente, el lote específico realizará, como bien se ha descrito en otros apartados del documento, la secuencia ordenada de tareas establecidas durante el modelado. De este modo, el lote en ejecución acabará finalmente en el almacén destinado al producto terminado para su posterior venta y distribución.

4.2. Simplificaciones y consideraciones

Una vez establecido el primer contacto con el modelo definido anteriormente y antes de proceder a abordar su implementación, conviene aclarar una serie de conceptos que han marcado el curso del desarrollo de este proyecto. Entre estos aspectos se encuentran tanto las simplificaciones y restricciones establecidas para acotar la complejidad del proceso, como el conjunto de consideraciones y fundamentos en torno a los cuales se ha desarrollado la implementación final de la solución.

Entre las simplificaciones más relevantes a tener en cuenta se encuentran las siguientes:

- Las tareas que impliquen actividades automáticas o de duración fija y conocida se han tomado como elementos de tipo Script con el tiempo fijado.
- El tiempo inicial para el transporte de la materia prima es fijo y conocido. Este concepto podría ser fácilmente modificable, trasladando una consulta sobre el tiempo estimado del trayecto al formulario cumplimentado para el pedido.
- Para la llegada y la recepción de camiones con materia prima a la entrada de la bodega no se incluye ningún mecanismo para su optimización. La contemplación de este mecanismo podría ser incluida en posteriores versiones del proyecto.
- La degradación de la uva en el almacén de materia prima no se tiene en consideración, permitiéndose su almacenamiento sin necesidad de desechar ninguna de las partidas de uva.
- Por regla general, los formularios de consulta incluidos para la interacción humana han sido reducidos y simplificados. Para posteriores versiones del modelo se plantearán mejoras sustanciales referidas a este punto concreto. Como caso particular, y debido a

la inexperiencia en algunos aspectos de la temática en cuestión, los formularios relativos a la inspección de materia prima, así como otros controles vinculados al proceso como lo son los controles de calidad, se han visto seriamente reducidos. Este hecho se ha realizado a fin de no complicar demasiado ni el proceso correspondiente ni la información almacenada en la tabla de la base de datos.

- Respecto al espacio de almacenamiento tanto para materia prima como para producto terminado, se contempla la posibilidad de arrendar siempre espacio en un almacén auxiliar. Este elemento auxiliar se considera siempre disponible ejerciendo, de este modo, de cortafuegos ante posibles desbordes de los productos a almacenar.
- La cantidad contenida en cubas, barricas, y botellas permanece fijada en la configuración inicial de la bodega permaneciendo, además, igual para los mismos elementos de su clase.
- La realización del marketing del producto se sale un poco de los objetivos principales del proyecto. Por este motivo, su modelado e implementación se definen de forma bastante simplificada a la complejidad real que este subproceso podría ofrecer. Este podría ser otro de los apartados interesantes en futuras versiones del proyecto.
- No se contempla detener la producción de vino aun cuando el almacén establecido para el producto terminado quede lleno. Sin embargo, para aliviar la carga del almacén, se contempla poder realizar promociones a la venta del producto y así potenciar su rápida distribución.
- En la elaboración del vino sólo se contemplan tres posibles calidades: vino joven, vino crianza y vino reserva. Además los tiempos estipulados para cada sus etapas de fermentación, crianza y maduración se encuentra fijados desde la configuración inicial de la bodega.
- El tiempo de envejecimiento para el vino joven se considera nulo, pasando desde su reposado y fermentación directamente al filtrado, clarificado y embotellado final. Esta consideración se plantea cercana a la realidad del producto ya que resulta aplicable a un alto porcentaje del vino joven producido.
- En ciertos procesos de producción, el vino joven no reposa en barrica sino que lo hace directamente en depósitos especiales dedicados a ello. En el proyecto se ha desechado esa oportunidad, estableciéndose la barrica como el recurso común destinado al reposo del vino.

- Los tiempos asociados al reposo y al envejecimiento del vino resultan totalmente configurables aunque se han permitido ciertas licencias. Estas consideraciones se recogen en el siguiente cuadro.

t_rep_joven ≤ t_rep_crianza ≤ t_rep_reserva
t_env_crianza ≤ t_env_reserva

  Aunque esta afirmación resulta bastante similar a la tendencia real del producto conviene remarcar su aplicación en el desarrollo del proyecto.

- Los controles de calidad realizados al término del periodo de crianza y envejecimiento se han considerado, por simplicidad, idénticos. De esta forma, la interacción en este control con el actor correspondiente se realizará dos veces en las partidas de vino crianza y reserva.
- La cantidad en kilogramos fijada para establecer los lotes se define en función de la capacidad máxima permitida en las cubas, siendo ambas cantidades equivalentes.
- Para la valoración de la reserva de recursos de la bodega se utiliza, principalmente en las fases iniciales del proceso, una variable nombrada como relación que establece la transformación entre kilogramos de uva y litros de vino. Para ajustar al máximo esta transformación, la variable se irá ajustando con cada iteración.

4.3. Modelado gráfico del proceso

Antes de comenzar con el apartado de modelado del proceso merece la pena incidir un poco sobre el desarrollo final del modelo del proceso. Para alcanzarlo se partió de una primera aproximación desarrollada mediante un software de Bizagi. Para hacer un seguimiento más exhaustivo del desarrollo previo planteado con esta herramienta se propone un capítulo adicional en el apartado correspondiente a los anexos de la memoria.

Como cabe esperar, a continuación se define la solución final resultante del proceso correspondiente con la elaboración de vino descrita anteriormente. Aunque para todo el proceso se seguirá un esquema general parecido al definido en Bizagi, esta solución contempla algunos cambios y, por tanto, resulta imprescindible la revisión completa de toda su estructura.

El proceso final establecido para la elaboración de vino se recoge en la figura presentada a continuación. En ella, el proceso incorpora todas las etapas descritas secuencialmente para la elaboración de vino

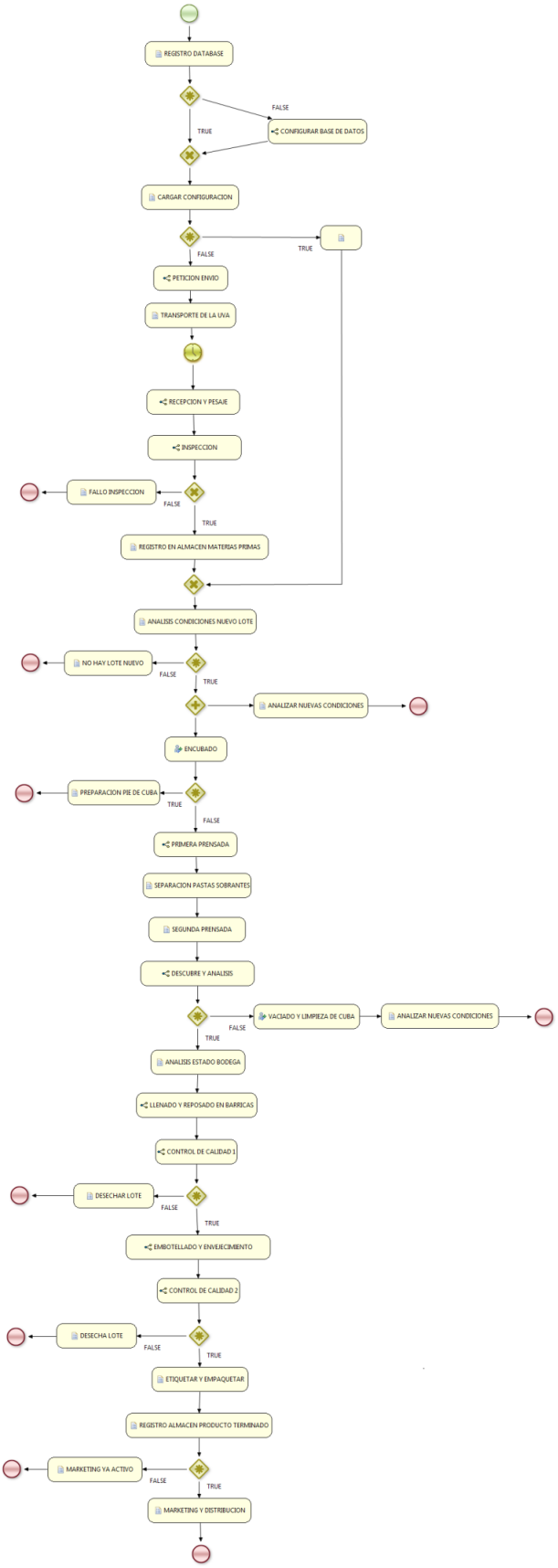


Figura 23: Proceso general completo para el modelado de la elaboración genérica de vino.

Como puede observarse en el diagrama anterior, existen algunas etapas que no se corresponden con el proceso descrito en el apartado 5.1. Esto se debe a la necesidad de incluir ciertas etapas adicionales como las dedicadas a los registros en la base de datos; las encargadas de evaluar condiciones o a la toma de decisiones, entre otras.

Por otro lado, y de forma análoga a la descripción realizada para el proceso en Bizagi, el proceso general incorpora también subprocesos, algunos de los cuales se describen a continuación. Antes de iniciar la descripción de los subprocesos, conviene destacar que gran parte de la explicación interna relativa al modelo quedará recogida en el apartado correspondiente a la implementación, ya que este apartado se asocia exclusivamente al modelado gráfico.

- *Petición envío* (petición.bpmn): este subproceso hace referencia a la petición de envío de materias primas por parte de la empresa encargada del viñedo. De este modo, tal y como se observa en el proceso general, cada nueva ejecución simboliza una nueva petición en el diagrama, siendo el punto de partida de toda modificación realizada en la bodega. Dentro del subproceso se implementará la tarea humana asociada a la petición de envío evaluando, para ello, la viabilidad de la respuesta en relación a las condiciones internas del sistema.

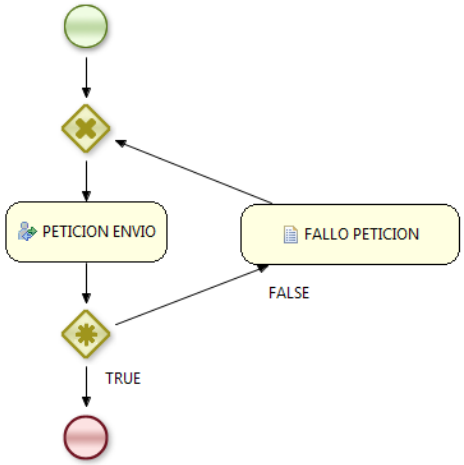


Figura 24: Subproceso correspondiente a la petición de envío de materias primas.

- *Inspección* (inspección.bpmn): este subproceso resulta bastante análogo con el representado en el modelo Bizagi. En este caso, se evaluarán las condiciones necesarias para determinar si una partida entrante de materia prima es capaz de

pasar la inspección de la bodega. Tras esta evaluación, se realiza el registro de la inspección en la base de datos para finalizar el subproceso.

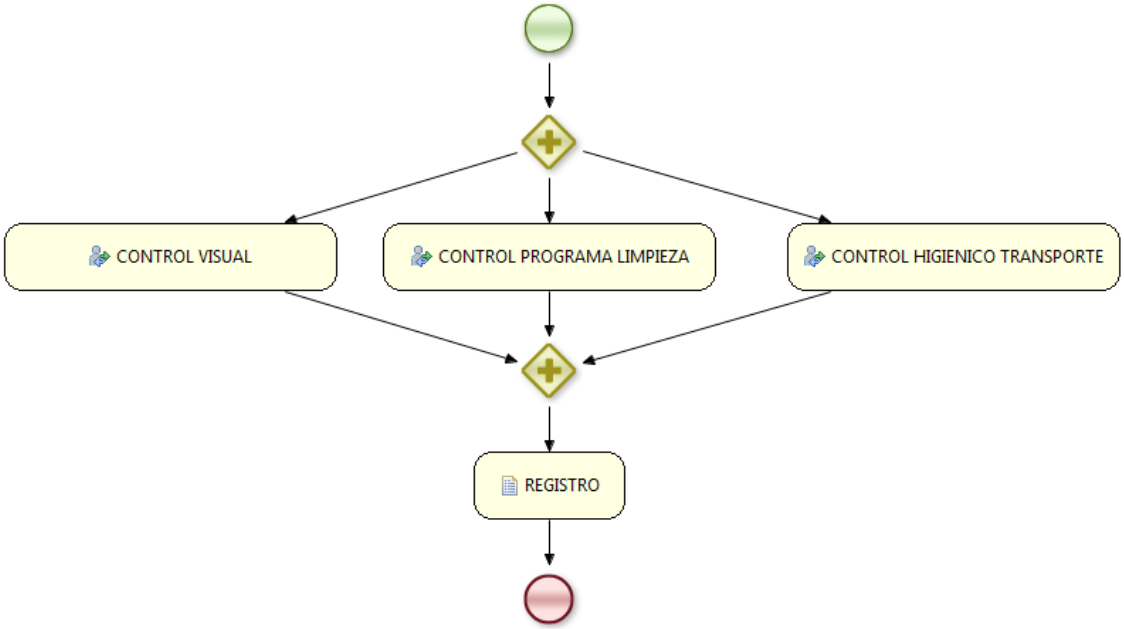


Figura 25: Subproceso de inspección para la materia prima entrante en la bodega.

- *Prensada* (prensada.bpmn): durante esta etapa se realizarán de forma secuencial tanto tareas tales como el sulfitado, despalillado, prensado o añadir el pie de cuba, como el correspondiente reposo en depósito para el reposado de las pastas resultantes.

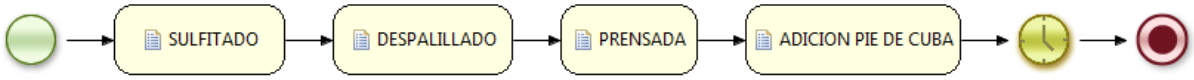


Figura 26: Subproceso de prensado y reposo de vino.

- Llenado y reposado en barricas (llenado.bpmn): durante la ejecución de este subproceso, como su propio nombre indica, se procede al llenado de las barricas de roble para la crianza del vino según el tiempo establecido para la calidad del producto. Tras este periodo, el proceso continúa con su correspondiente clarificación, filtrado y control de sulfitado. Durante esta etapa se añaden nodos adicionales que se comentarán con mayor profundidad en el apartado dedicado a la implementación.

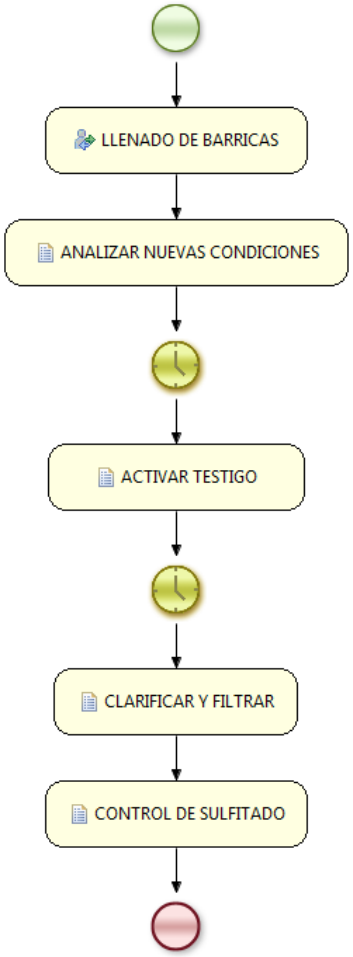


Figura 27: Subproceso para el llenado y reposo en barricas.

- *Control de calidad* (calidad.bpmn): en esta etapa, los enólogos realizan los controles de calidad necesarios para garantizar las características finales del producto elaborado. Este subproceso, al igual que en el análogo definido en Bizagi, plantea la realización de controles físicos y químicos al producto, permitiendo además implementar medidas correctoras siempre que los análisis no sean concluyentes.

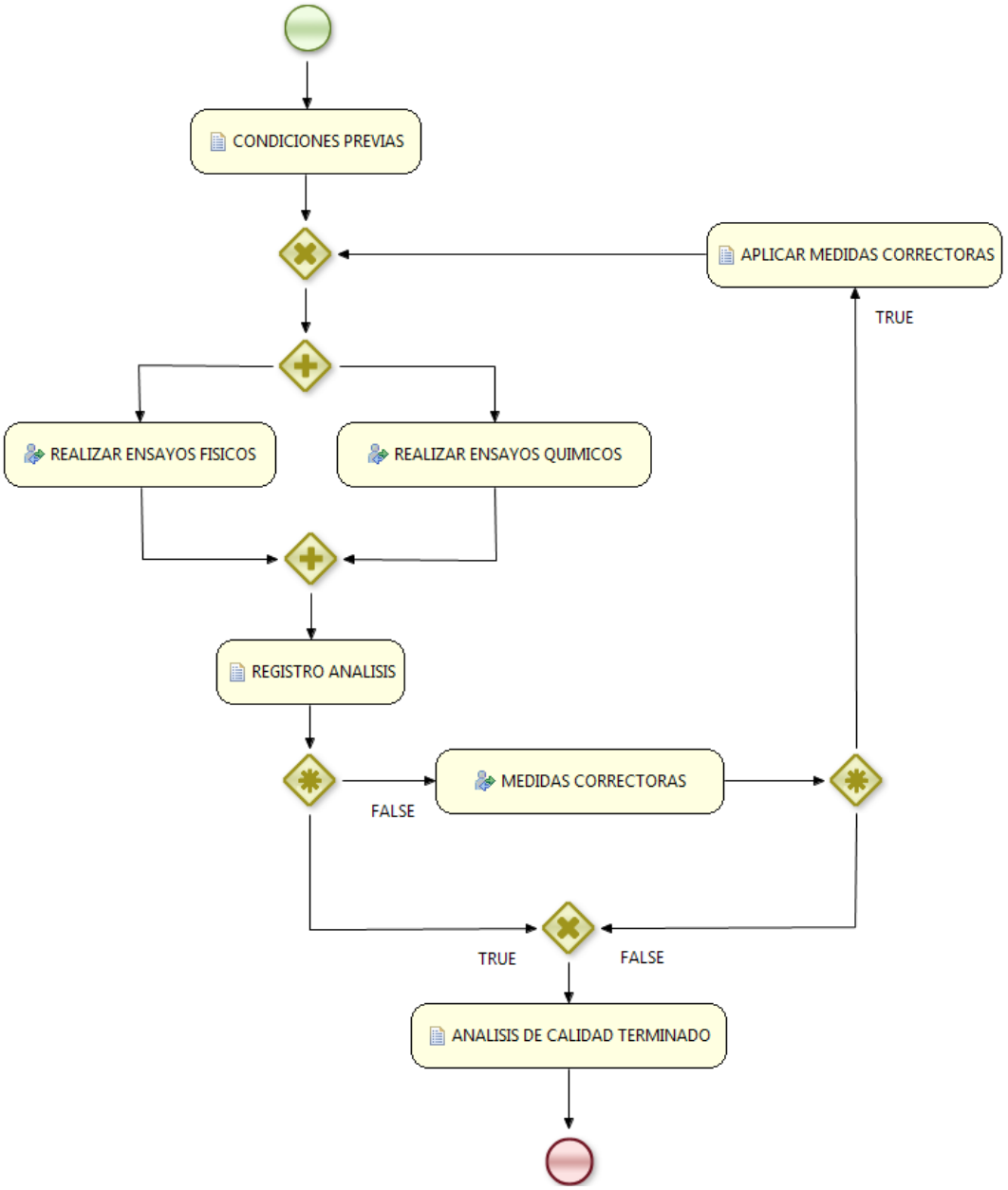


Figura 28: Subproceso dedicado al control de calidad del vino.

- *Embotellado y envejecimiento* (embotellado.bpmn): para completar este subproceso, se procederá al embotellado del vino y a su correspondiente envejecimiento en botella. Durante este proceso también se llevarán a cabo consultas y modificaciones a algunas variables internas del proceso realizando además conexiones a la base de datos de la bodega. Todas estas modificaciones quedarán ampliamente recogidas en el apartado correspondiente a la implementación.

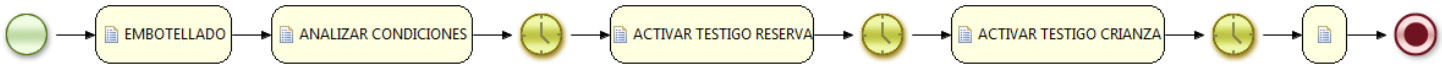


Figura 29: Subproceso correspondiente al embotellado y envejecimiento del vino en botellas.

- *Marketing y distribución* (marketing.bpmn): este subproceso será el encargado de evaluar las condiciones para llevar a cabo el marketing del producto, así como elaborar y enviar los oportunos informes de distribución para el posterior envío del producto a los clientes.

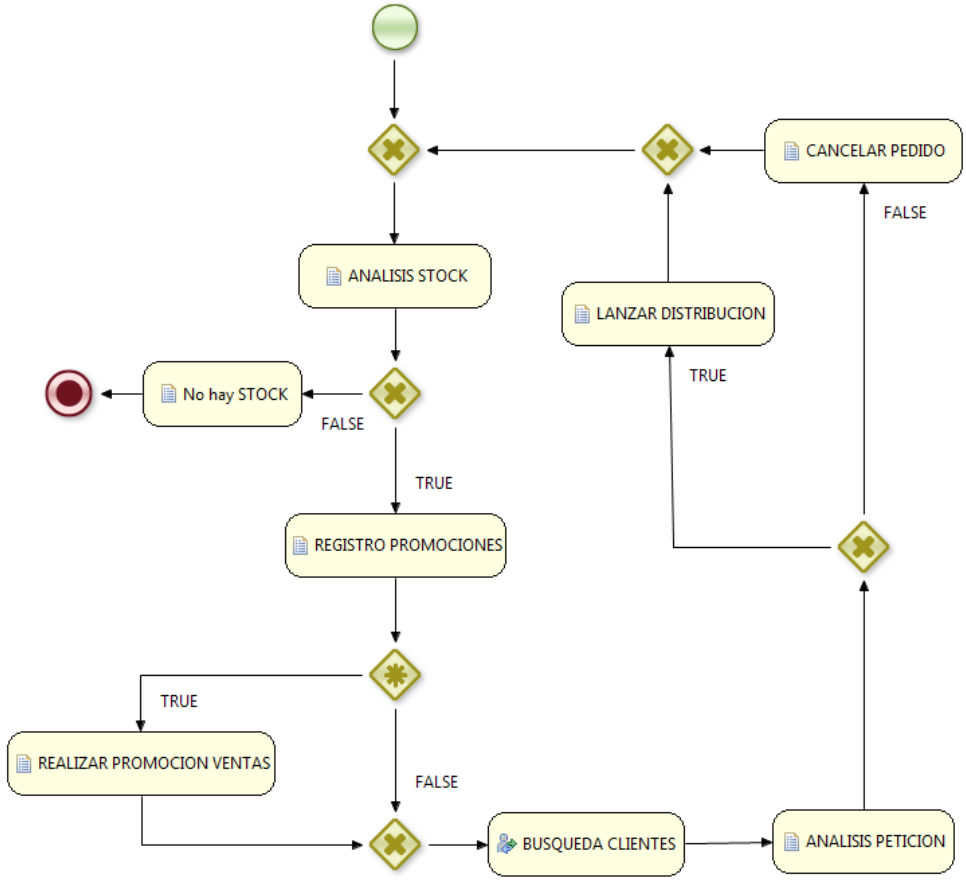


Figura 30: Subproceso dedicado al marketing y la distribución del producto elaborado.

4.4. Base de datos del sistema

Como se comentó en el apartado correspondiente al servidor de aplicación JBoss, la base de datos implementada en el proyecto se corresponde con HyperSQL. Esta base de datos, fácilmente accesible y configurable desde la interfaz JMX para la gestión del servidor, aporta una potente herramienta para el desarrollo y la implementación de este proyecto. Mediante la

librería estándar de JDBC se irán realizando distintas conexiones a través del driver de la base de datos, consiguiendo una interacción directa con la información almacenada en su interior.

A lo largo de esta sección se propone un acercamiento a la estructura interna de la información contenida en la base de datos. Para ello se presentará no sólo un análisis detallado de sus elementos, sino también una descripción exhaustiva de las principales relaciones existentes entre ellos y con el proceso general para el que se definen. Una vez terminada la presentación de la base de datos se presentará además el modelo entidad relación del conjunto estableciendo, de forma gráfica, la estructura general de todo el conjunto.

Para la definición de cada una de las tablas implementadas en la base de datos se ha seguido un esquema ordenado por su secuencia de interacción durante la ejecución del proceso. Como cabe esperar existirán además otras entradas de la base de datos dedicadas tanto a la configuración como a la gestión del sistema. Estos elementos, fácilmente reconocibles por su apelativo, serán frecuentemente utilizados a lo largo de toda la ejecución del proceso y, por este motivo, se describirán en primer lugar.

La primera de las entradas a definir en la base de datos hace referencia a la tabla *INDICE*. Esta tabla, contemplada como elemento auxiliar de ejecución, almacenará un registro de los índices para la creación de nuevos registro en las principales tablas albergadas en la base de datos. El principal objetivo de esta definición radica en evitar consultas y bucles de búsqueda innecesarios. Como cabe esperar, todos los elementos albergados será números enteros que se corresponderán con el siguiente identificador libre en la tabla especificada por el campo nombre.

Otra de los elementos encargados de auxiliar la ejecución del proceso es la tabla *VARIABLES_BOOLEANAS*. Dentro de esta tabla quedarán recogidos algunos testigos encargados de modificar el comportamiento de la ejecución según el estado específico del sistema. Entre ellos se contemplan algunos testigos dedicados al aviso de desbordamientos en los almacenes, la activación de marketing y las promociones, o la gestión de la correcta ejecución multihilos sin fallos, entre otros.

Para recoger adecuadamente los parámetros establecidos durante la configuración se define la tabla *CONFIGURACION*. Esta tabla se encargará de almacenar los valores fijados durante la etapa de configuración previa del sistema.

Una vez establecidas las tablas encargadas de dar soporte a la ejecución del sistema, los siguientes elementos de la base de datos se adecuarán al curso previsto por el proceso general. Para llevarlas a cabo, resulta necesario, en primer lugar, establecer qué integrante del proceso realizará la tarea que corresponda durante la ejecución del flujo de trabajo. Con objeto de establecer esta asociación se propone almacenar a los distintos actores y empleados de la bodega como entradas de una tabla denominada *EMPLEADO*. Como se puede esperar esta tabla albergará algunos datos relacionados con la actividad, estado u otros aspectos específicos que se tenga a bien añadir como nombre, dirección, email, etc.

De vuelta a la secuencia establecida para el proceso de elaboración, y una vez cumplimentados los correspondientes formularios de aceptación del envío, se procederá a la creación de dos entradas nuevas en tablas alojadas en la base de datos. La primera de ellas hace referencia a la tabla *PEDIDO*. En esta tabla se recogerán los nuevos envíos entrantes a la bodega identificando para ello al empleado involucrado en su manipulación; la cantidad de materia prima que incluye y una referencia hacia a un informe de inspección realizado antes de aceptar el pedido.

La otra entrada generada tras aceptar una nueva petición de envío se localiza en la tabla *CAMIONES*. Esta tabla se establece con objeto de dejar no sólo un histórico de los envíos aceptados en la bodega, sino también de dejar constancia de los pedidos aún sin llegar al destino. De esta forma podrán tenerse en cuenta en las futuras peticiones de envío y tener así un control más real de la capacidad útil del almacén a la entrada.

Para poder aceptar la mercancía será necesario pasar un control exhaustivo a la entrada de la bodega. Este control deberá asegurar una serie de condiciones de seguridad e higiene generando un informe de incidencias asociado a la inspección. A fin de dejar constancia de este evento y relacionarlo así con el correspondiente pedido entrante, se añade una entrada nueva en la tabla denominada *INSPECCION*. Dentro de esta tabla se recogerán los datos más significativos resultantes de la inspección de la materia prima que llegue a la bodega. Los parámetros incorporados en esta tabla se recogen a continuación:

Después de aceptar el pedido se procederá a su recepción el almacén de materia prima. Para llevar un control directo sobre la actividad real de este almacén y realizar la toma de decisión necesaria y realizar así su correcta gestión se define la tabla *ALMACEN_MP*. Dentro

de esta tabla quedarán recogidos todos los almacenes destinados a albergar la materia prima entrante en la bodega, especificando además algunos parámetros asociados así como su estado interno.

Una vez cumplidas las condiciones establecidas para la creación de un nuevo lote, se agrupa una cantidad fijada de materia prima para continuar con las siguientes etapas en la elaboración del vino. Esta agrupación, establecida como cantidad conocida desde la configuración inicial del sistema y almacenada convenientemente en la tabla *CONFIGURACION*, se representará como una entrada en la tabla conocida como *LOTE*. Esta tabla permitirá asociar a dicho lote las subsiguientes etapas remarcables del proceso y realizar así un control completo del producto elaborado.

Tras establecer un nuevo lote y completar las siguientes etapas previas al reposado del vino se procede al llenado de las cubas. Estos elementos consumibles dentro del sistema completo formado por la bodega permanecen representados individualmente por entradas en la tabla *CUBA*. Dentro de esta tabla se encuentran recogidas todas las cubas registradas en la configuración inicial de la bodega, además de otros parámetros relacionados con su gestión y manipulación.

Ya terminado el tiempo estipulado para el reposo y fermentado en la cuba se procede al primer análisis del producto. Este análisis, encargado de determinar las características potenciales del vino, quedará enlazado inequívocamente con la correspondiente partida o lote. Para ello se añade una entrada nueva a la tabla *ANALISIS* recogiendo en ella algunos de los aspectos más interesantes incluidos en los ensayos realizados al producto en cuestión.

Después de completar el análisis anterior y determinar así la calidad potencial del producto, se procederá a la siguiente etapa del proceso. Esta etapa, directamente dependiente de la calidad establecida al producto, será consecuencia de la toma de decisiones asociada a la planificación optimizada de la producción en la bodega. De este modo, y exclusivamente para vinos con calidades de crianza o reserva, el siguiente paso en el proceso les llevará a las barricas de roble, iniciando así su periodo de crianza. Estas barricas quedarán recogidas individualmente en la base de datos como entradas en la tabla *BARRICA*. Cada una de estas entradas tendrá asociada una serie de características específicas de la barrica así como otros elementos orientados a facilitar la administración de las distintas barricas.

Tal y como se comentó en apartados anteriores de esta memoria, tras completar el tiempo de crianza se procede a realizar un análisis de calidad del producto. Esta etapa de control quedará recogida nuevamente en la base de datos como una entrada a una tabla especificada como *CALIDAD*. Entre los parámetros contemplados dentro de esta tabla se encontrará la información obtenida al realizar los correspondientes ensayos físicos y químicos al producto.

A partir de los ensayos anteriores se establece un seguimiento controlado al producto, dando paso a la siguiente etapa del proceso. Esta nueva etapa, alcanzada tras el filtrado y embotellado, representa el inicio del periodo de envejecimiento del vino. Las limitaciones operacionales en este periodo vendrán descritas por la capacidad de almacenamiento de la bodega de envejecimiento. Para facilitar la administración de sus elementos se ha segmentado la capacidad total en agrupaciones fijas de botellas recogidas como entradas de la tabla *BOTELLAS*. Estos elementos, al igual que las barricas, serán elementos consumibles por el proceso quedando reservados durante el envejecimiento. Por este motivo, también será necesario incluir campos adicionales para facilitar la administración de estos elementos.

Una vez finalizado el periodo dedicado al envejecimiento en botella, y replicando el control de calidad comentado anteriormente, se realiza el registro del producto terminado en su correspondiente almacén. En su interior, el producto elaborado se irá almacenado a la espera de realizar su marketing y distribución. Para la gestión interna del almacén de productos terminados se ha optado también por la creación de una tabla en la base de datos denominada, en este caso, *ALMACEN_PT*.

Tras haber sido registrado en el almacén correspondiente, el producto estará preparado para su venta. Para llevarla a cabo se han tenido en cuenta tres conceptos clave. El primero de ellos se corresponderá con un registro exhaustivo de los clientes de la bodega mediante la tabla *CLIENTE* para poder enlazar así las posibles ventas que tengan asociadas. El siguiente aspecto quedará marcado por el propio proceso de venta recogiendo, en la tabla *VENTA,* ciertas características del pedido como la cantidad, la fecha o información relativa al lote del que parte. El último aspecto vendrá representado por el parte de distribución del producto a entregar al transportista que quedará asociado mediante una nueva entrada en la tabla *DISTRIBUCION*.

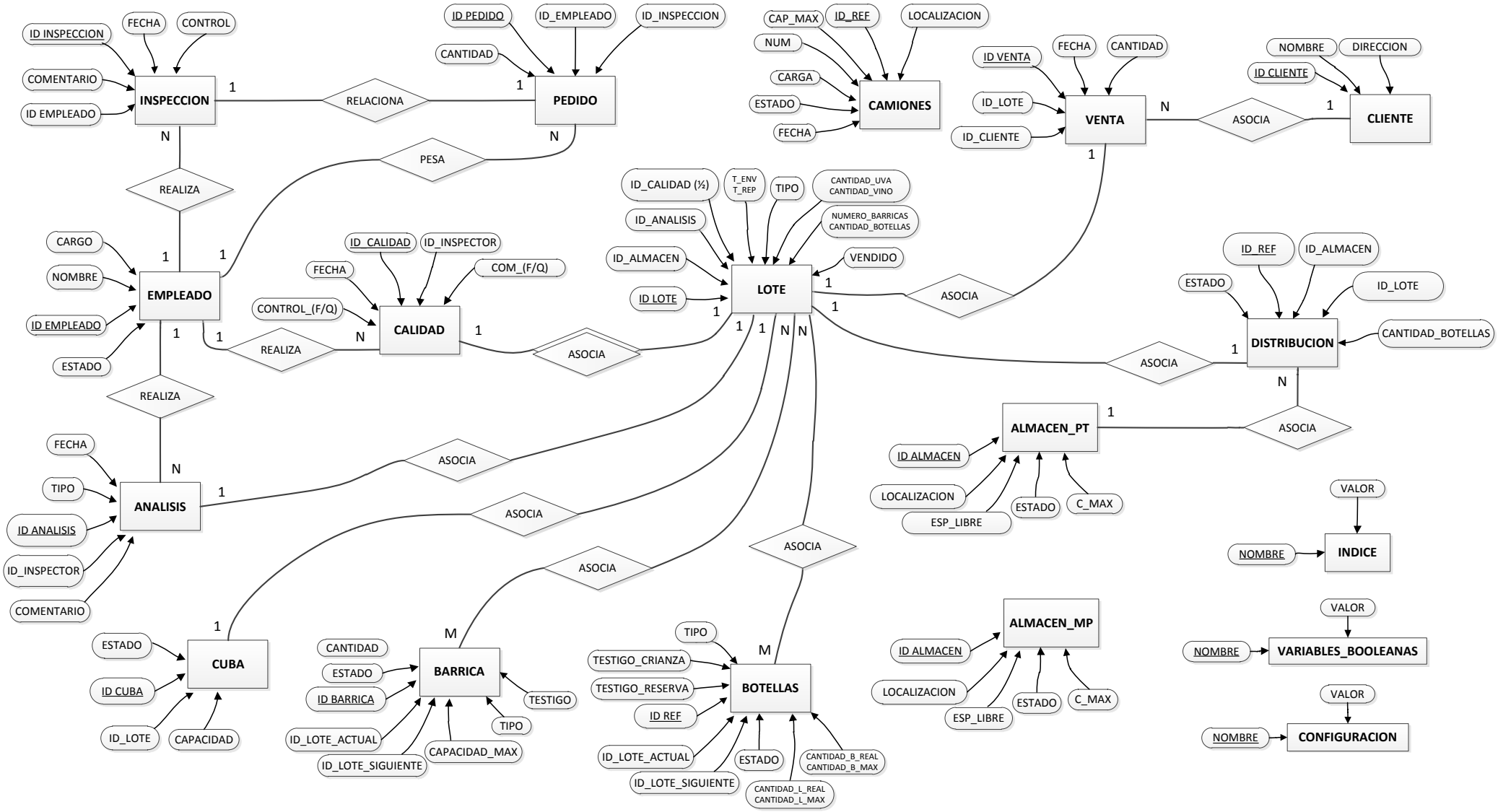


Figura 31: Representación del modelo de entidad relación de la base de datos del sistema

4.5. Implementación del sistema

4.5.1. Etapa de configuración

En la siguiente sección de la memoria se realizará un seguimiento detallado de la implementación de la solución presentada en este proyecto para la planificación general de una bodega. A lo largo de este apartado también se detallarán algunos aspectos críticos de la programación así como otros factores a tener en cuenta en la implementación final del modelo. Para llevar a cabo este seguimiento se ha seguido el esquema de modelado definido en la correspondiente sección de modelado en BPMN de la memoria y representado en la figura 25. Para una visualización más detallada de este modelo se puede cómodamente acceder al adecuado anexo enlazado en la sección final de esta memoria.

Como aclaración previa, el conjunto de librerías utilizadas durante la programación de los nodos del proceso resulta común a la gran mayoría de ellos. Esta agrupación de librerías permanece recogida en el siguiente cuadro:

```
import org.drools.KnowledgeBase
import org.drools.builder.KnowledgeBuilder
import org.drools.builder.KnowledgeBuilderFactory
import org.drools.builder.ResourceType
import org.drools.io.ResourceFactory
import org.drools.runtime.StatefulKnowledgeSession
import java.sql.*
import java.util.HashMap
import java.util.Map
import java.sql.Connection
import java.sql.DriverManager
import java.sql.ResultSet
import java.sql.Statement
import java.util.Date
import org.drools.logger.KnowledgeRuntimeLogger
import org.drools.logger.KnowledgeRuntimeLoggerFactory
import org.drools.logger.KnowledgeRuntimeLoggerFactory
import org.jbpm.process.instance.*
import org.jbpm.process.workitem.wsht.WSHumanTaskHandler
```

Entre las más representativas se encuentran algunas como las asociadas a las conexiones remotas con la base de datos mediante conectores JDBC; otras como las relacionadas con la ejecución de sesiones para la ejecución de procesos u otras para la generación de registros de funcionamiento para la gestión de esta ejecución. Algunas de estas librerías quedarán convenientemente recogidas con mayor profundidad en el apartado

dedicado a la simulación del proceso de negocio mientras que otras se irán viendo a lo largo de la presentación de esta implementación.

Otro de los conceptos comunes definidos por esta implementación lo representa el uso de un conjunto de variables globales asociadas a cada ejecución del proceso. Estas variables quedarán definidas desde el inicio de la ejecución, permitiendo un espacio de memoria útil e independiente. Entre las principales variables especificadas al inicio se destacan las siguientes:

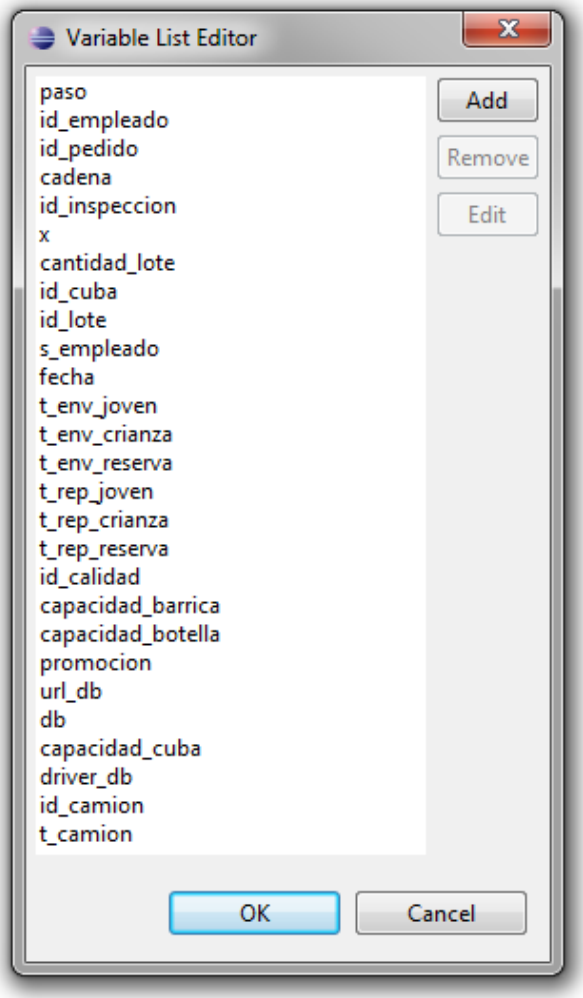


Figura 32: Captura de pantalla sobre el editor de variables de Eclipse

El principal objetivo de este conjunto de variables se centra en reducir el número de conexiones a la base de datos mediante JDBC y mejorar así la accesibilidad general para todos los procesos simultáneos ejecutados en el servidor. Entre este conjunto de variables presentados en la figura anterior se pueden diferenciar algunas como las destinadas a almacenar los tiempos de transporte, reposo y envejecimiento; otras como las orientadas a

almacenar datos específicos para la conexión con la base de datos u otras utilizadas a modo de registros auxiliares de funcionamiento, etc.

A partir de esta definición previa de los elementos comunes al sistema se procederá al seguimiento secuencial del funcionamiento, detallando para ello los puntos de interés más determinantes para llevar a cabo su implementación.

La fase inicial del sistema contempla la definición y configuración de su base de datos. Esta comprobación, realizada convenientemente con cada ejecución del proceso, se encargará de asegurar la consistencia operativa del proceso durante su ejecución. Esta primera etapa del proceso quedará recogida por el siguiente diagrama en BPMN:

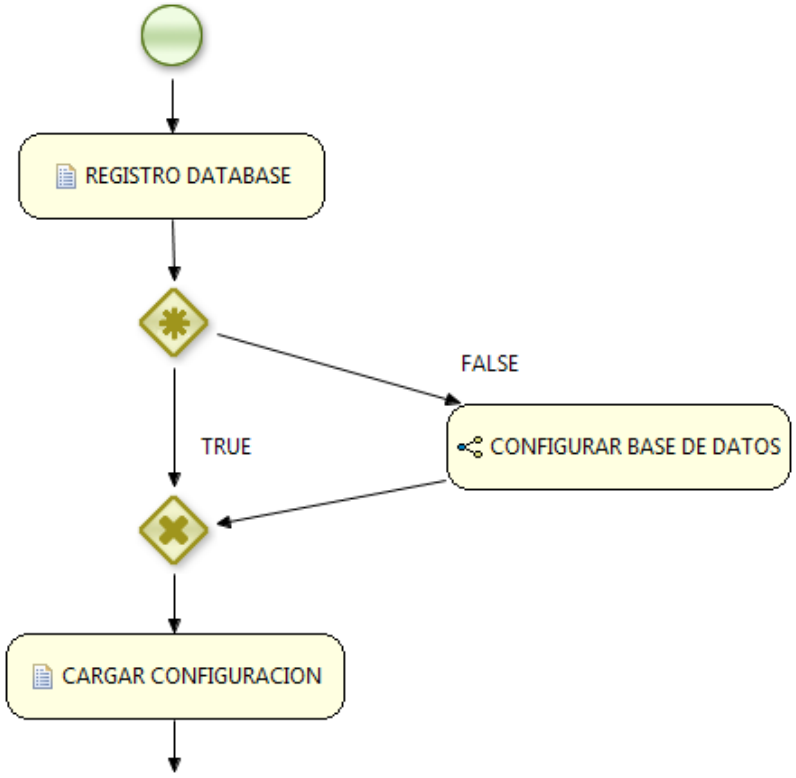


Figura 33: Etapa inicial del proceso para la configuración general del sistema

Como se puede observar, esta etapa representa un estadio previo a la propia elaboración de vino y se realizará, por tanto, de forma automática mediante la ejecución de una serie de nodos de tipo *script task*. El primero de ellos, descrito en las líneas de código que se presentan a continuación, se encargará de comprobar la existencia de una base de datos asociada al sistema. En caso de no existir, el proceso procederá a su creación así como a la configuración de ciertos parámetros relevantes relacionados con esta definición inicial. Para rellenar buena parte de los parámetros operativos de la bodega se realizará una consulta al

administrador del proceso, permitiendo de este modo plena libertad en su configuración. Tal y como se observa en el código presentado en el cuadro, tras la creación de la base de datos se activa el camino hacia el subproceso denominado configurar base de datos.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
try {       // Se carga el controlador para JDBC
            Class.forName(driver_db).newInstance();
            conn = DriverManager.getConnection(url_db,"sa","");
            st  = conn.createStatement();
}
catch (Exception ex){
            System.out.println("PROBLEMA CONEXION");
            System.out.println( ex );
}
try {
            rst1 = st.executeQuery("select * from variables_boleanas where nombre='base datos'");
            rst1.next();
            paso=rst1.getBoolean("valor");
 }
catch (Exception ex){  // No existe base de datos previa
            System.out.println("PRIMERA CONEXION A BASE DE DATOS");
            sql = "create table variables_boleanas (nombre char(20) not null primary key, valor boolean );";
            st.executeUpdate(sql);
            .executeUpdate( "insert into variables_boleanas values ('base datos', false)");
            paso=false;
}
try {
            if (paso==false)
                        {
                        System.out.println("CREANDO BASE DE DATOS ..... ");
                        sql = "CREATE TABLE EMPLEADO ( id_empleado INTEGER NOT NULL PRIMARY KEY ,
                        nombre CHAR (30) not null, cargo CHAR(1) not null, estado BOOLEAN not null );";
                        st.executeUpdate(sql);

                        sql = "CREATE TABLE CALIDAD ( id_calidad  INTEGER NOT NULL PRIMARY KEY ,
                        id_inspector  INTEGER not null,  fecha  DATE , control_f BOOLEAN ,control_q BOOLEAN,
                        com_f CHAR (100), com_q CHAR (100), control_correct BOOLEAN,com_correct CHAR
                        (100), FOREIGN KEY (id_inspector) REFERENCES EMPLEADO (id_empleado));";
                        st.executeUpdate(sql);
                        sql = "CREATE TABLE ANALISIS ( id_analisis   INTEGER NOT NULL PRIMARY KEY ,
                        id_inspector  INTEGER,  fecha DATE, tipo CHAR (1) , comentario varCHAR (100),
                        FOREIGN KEY (id_inspector) REFERENCES EMPLEADO (id_empleado));";
                        st.executeUpdate(sql);

                        sql = "CREATE TABLE ALMACEN_MP ( id_almacen  INTEGER NOT NULL PRIMARY KEY ,
                        localizacion CHAR (50) not null,  esp_libre FLOAT not null,  estado BOOLEAN not null,
                        c_max FLOAT );";
                        st.executeUpdate(sql);

                        sql = "CREATE TABLE ALMACEN_PT ( id_almacen  INTEGER NOT NULL PRIMARY KEY ,
                        localizacion CHAR (50) not null,  esp_libre FLOAT not null,  estado BOOLEAN not null,
                        c_max FLOAT );";
                        st.executeUpdate(sql);
```

```
sql = "CREATE TABLE LOTE ( id_lote  INTEGER NOT NULL PRIMARY KEY  , id_almacen
INTEGER,id_cuba INTEGER , id_analisis INTEGER  , id_calidad_1 INTEGER,
id_calidad_2 INTEGER , t_env FLOAT, t_rep FLOAT, tipo CHAR(1), cantidad_uva
FLOAT, numero_barricas INTEGER, cantidad_vino  FLOAT, cantidad_botellas
INTEGER, vendido BOOLEAN, FOREIGN KEY (id_almacen) REFERENCES ALMACEN_PT
(id_almacen), FOREIGN KEY (id_calidad_1) REFERENCES CALIDAD (id_calidad),
FOREIGN KEY (id_calidad_2) REFERENCES CALIDAD (id_calidad), FOREIGN KEY
(id_analisis) REFERENCES ANALISIS (id_analisis));";
st.executeUpdate(sql);

sql = "CREATE TABLE INSPECCION ( id_insp INTEGER NOT NULL PRIMARY KEY ,
id_inspector INTEGER, fecha DATE, com_l CHAR(500), com_t CHAR(500), com_v
CHAR(500), control_lBOOLEAN, control_t BOOLEAN, control_v  BOOLEAN, FOREIGN
KEY (id_inspector) REFERENCES EMPLEADO (id_empleado));";
st.executeUpdate(sql);

= "CREATE TABLE PEDIDO ( id_pedido INTEGER  NOT NULL  PRIMARY KEY ,
id_empleado INTEGER, id_inspeccion INTEGER, cantidad FLOAT, FOREIGN KEY
(id_empleado) REFERENCES EMPLEADO(id_empleado), FOREIGN KEY (id_inspeccion)
REFERENCES INSPECCION (id_insp));";
st.executeUpdate(sql);

sql = "CREATE TABLE CUBA ( id_cuba INTEGER NOT NULL PRIMARY KEY , capacidad
FLOAT, estado BOOLEAN , id_lote INTEGER, tipo CHAR(1),  FOREIGN KEY (id_lote)
REFERENCES LOTE (id_lote) );";
st.executeUpdate(sql);

sql = "CREATE TABLE BARRICA ( id_barrica    INTEGER NOT NULL PRIMARY KEY ,
capacidad_max    FLOAT , cantidad FLOAT , estado    BOOLEAN ,testig  BOOLEAN ,
tipo CHAR(1) , id_lote_actual INTEGER , id_lote_siguiente INTEGER,  FOREIGN KEY
(id_lote_actual) REFERENCES LOTE (id_lote),  FOREIGN KEY (id_lote_siguiente)
REFERENCES LOTE (id_lote) );";
st.executeUpdate(sql);

sql = "CREATE TABLE BOTELLAS ( id_ref INTEGER NOT NULL primary key,
cantidad_b_max INTEGER, cantidad_b_real INTEGER , cantidad_l_real FLOAT,
cantidad_l_max FLOAT , estado BOOLEAN ,testigo_crianza BOOLEAN
,testigo_reserva BOOLEAN , tipo CHAR(1) , id_lote_actual INTEGER , id_lote_siguiente
INTEGER,  FOREIGN KEY (id_lote_actual) REFERENCES LOTE (id_lote),  FOREIGN KEY
(id_lote_siguiente) REFERENCES LOTE (id_lote) )";
st.executeUpdate(sql);
 sql = "CREATE TABLE CLIENTE ( id_cliente INTEGER NOT NULL PRIMARY KEY ,
nombre  CHAR(50) not null, direccion CHAR(100) not null );";
st.executeUpdate(sql);

sql = "CREATE TABLE VENTA ( id_venta INTEGER NOT NULL PRIMARY KEY , id_lote
INTEGER not null, id_cliente INTEGER not null, fecha DATE not null , cantidad
FLOAT ,FOREIGN KEY (id_lote) REFERENCES LOTE (id_lote), FOREIGN KEY (id_cliente)
REFERENCES CLIENTE (id_cliente));";
st.executeUpdate(sql);

sql="CREATE TABLE DISTRIBUCION ( id_ref    INTEGER NOT NULL    , id_almacen
INTEGER NOT NULL , id_lote INTEGER , cantidad_botellas          INTEGER, estado
BOOLEAN,  FOREIGN KEY (id_almacen) REFERENCES ALMACEN_PT (id_almacen),
FOREIGN KEY (id_lote) REFERENCES LOTE (id_lote), PRIMARY KEY (id_ref,
id_almacen));";
```

```
                    st.executeUpdate(sql);

                    sql="CREATE TABLE CONFIGURACION ( nombre char(20) not null primary key, valor
                    float );";
                    st.executeUpdate(sql);

                    sql="create table indice (nombre char(20) not null primary key, valor integer );";
                    st.executeUpdate(sql);

                    sql="create table camiones (id_ref    INTEGER NOT NULL primary key, localizacion
                    char(20), num INTEGER, cap_max FLOAT, carga FLOAT, estado BOOLEAN, fecha
                    DATE);";
                    st.executeUpdate(sql);

                    st.executeUpdate( "update variables_boleanas set valor=true where nombre='base
                    datos'");
                    db=false;
            }
            else{
                    rst1 = st.executeQuery("select * from variables_boleanas where nombre=
                    'almacen_mp'");
                    rst1.next();
                    paso=rst1.getBoolean("valor");
                    paso=true;
            }
}
catch (Exception ex){
            paso=false;

}
            java.util.Map contentParam = new java.util.HashMap();
            kcontext.setVariable("paso", paso);
            kcontext.setVariable("db", db);
 try {
          st.executeUpdate("SHUTDOWN");
}
 catch (Exception e) {
            System.out.println(e);
 }
```

Tras analizar brevemente el código se pueden observar dos caminos posibles. Por un lado se plantea la creación desde cero de una base de datos relacional según el esquema ya presentado en esta memoria, o bien la omisión de todo este proceso declarando al nodo CARGAR CONFIGURACION como el siguiente nodo en la ejecución.

En el primer caso, el valor de la variable paso se establecerá como *false*, pasando por tanto a la ejecución del subproceso CONFIGURAR BASE DE DATOS. Dentro de este subproceso se procederá a inicializar gran parte de los contenidos iniciales de la base de datos así como

definir el estado inicial de la bodega. Para realizar este procedimiento se seguirá la secuencia definida en el siguiente diagrama BPMN.

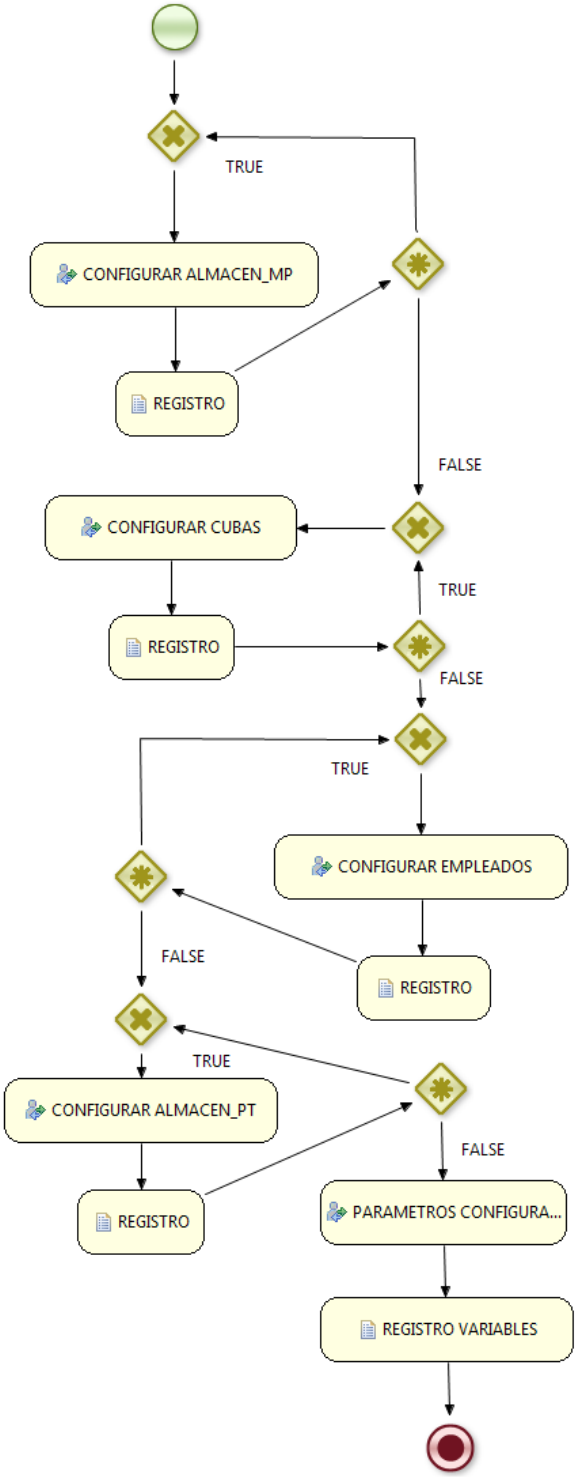


Figura 34: Subproceso para la configuración de las condiciones iniciales de la bodega.

A lo largo de este subproceso se irán lanzando distintos formularios asociados a la configuración general del sistema. Toda esta sucesión de nodos se encargará de definir

secuencialmente aspectos relacionados con los almacenes de la bodega, la capacidad de todos los elementos o la cantidad y las características de los consumibles del proceso. De forma adicional también definirá un registro inicial de todos los empleados de la bodega a fin de realizar las futuras asociaciones en las siguientes etapas del proceso.

Durante todo el periodo de ejecución, las tareas estipuladas como *human task* se encargarán de lanzar una serie de formularios por medio de la interfaz web del servidor, realizando de este modo las interacciones necesarias con los distintos actores del proceso. En el caso particular de la configuración inicial del sistema, esta interacción se realiza con el administrador del sistema, asegurando así la personalización de los elementos fundamentales de la bodega. En la siguiente figura se muestra un esquema visual de algunos de los contenidos lanzados hacia el administrador para su inicialización del sistema.

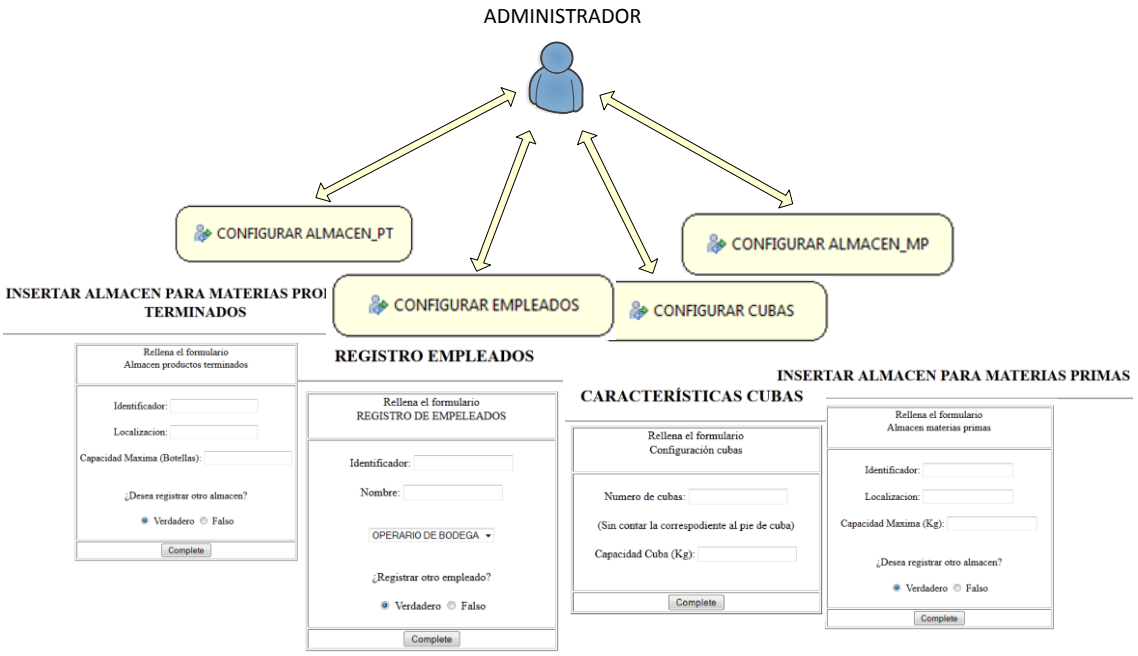
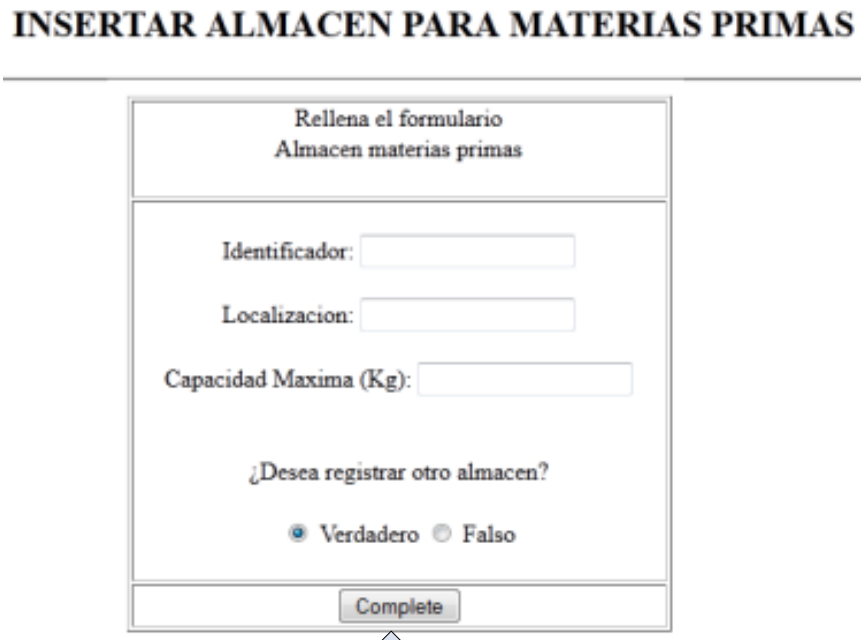


Figura 35: Representación sobre la interacción del administrador en la configuración del sistema.

La definición de estos formularios de interacción se basa en las plantillas FTL (Freemaker Template Language), que representa un motor de plantillas basado en java y optimizado para software implementado bajo arquitecturas MVC (Model View Cotroler). Como ejemplo real de esta clase de programación se presenta a continuación una asociación directa entre uno de los formularios escritos en FTL y su correspondiente visualización en un navegador web.




```
<html>
<head>
</head>
<body>
<center><h2>INSERTAR ALMACEN PARA MATERIAS PRIMAS</h2></center>
<hr>
<form name="Conf1"action="complete" method="POST"
enctype="multipart/form-data">
<TABLE BORDER="1" ALIGN="center">
<br/>
<TR>
<TD WIDTH="350">
<center>Rellena el formulario  </center>
<center> Almacen materias primas</center>
<br/>
</TD>
<form>
</br>
<TR>
<TD WIDTH="350">
</br>
<center>Identificador:<input name="string1" maxlength="25"
type="text"></center>
</br>
<center>Localizacion:<input name="string2" maxlength="25"
type="text"></center>
</br>
<center>Capacidad Maxima (Kg):<input name="string3" maxlength="25"
type="text"></center>
</br>
</br>
<center>¿Desea registrar otro almacen?</center>
</br>
<center>
<input checked="checked" name="s1" type="radio" value="true" />
Verdadero
<input name="s1" type="radio" value="false" /> Falso
</center>
</br>
</TD>
</br>
</br>
</br>
<TR>
<TD WIDTH="350">
<center><input type="submit" value="Complete"></center>
</TD>
</form>
</body>
</html>
```

Figura 36: Asociación entre las plantillas FTL y el código específico sobre el que se definen.

A lo largo de la ejecución del proceso, ésta representará la interacción predominante con los actores de la bodega. En la sección correspondiente a la simulación del proceso se analizará en profundidad esta clase de interacción, describiendo para ello los distintos formularios lanzados en el proceso. En este apartado merece la pena centrarse en la comunicación entre las plantillas FTL y el proceso en ejecución. Para trasladar la información introducida por los usuarios en la plantilla al proceso interno en ejecución se hace uso de la siguiente sentencia:

```
<input name="string1" maxlength="25" type="text">
```

Esta declaración permitirá almacenar, con formato String, los datos introducidos en el formulario en la variable string1. Esta variable será introducida como parámetro de entrada al nodo *human task,* realizando además el correspondiente mapeo a la salida una vez completada la plantilla. Para asociar este dato con el correspondiente valor y formato adecuado para el proceso será necesario realizar una conversión a la salida del nodo.

Tanto esta asociación como la conversión necesaria a la salida del nodo quedan recogidas en la figura presentada a continuación:

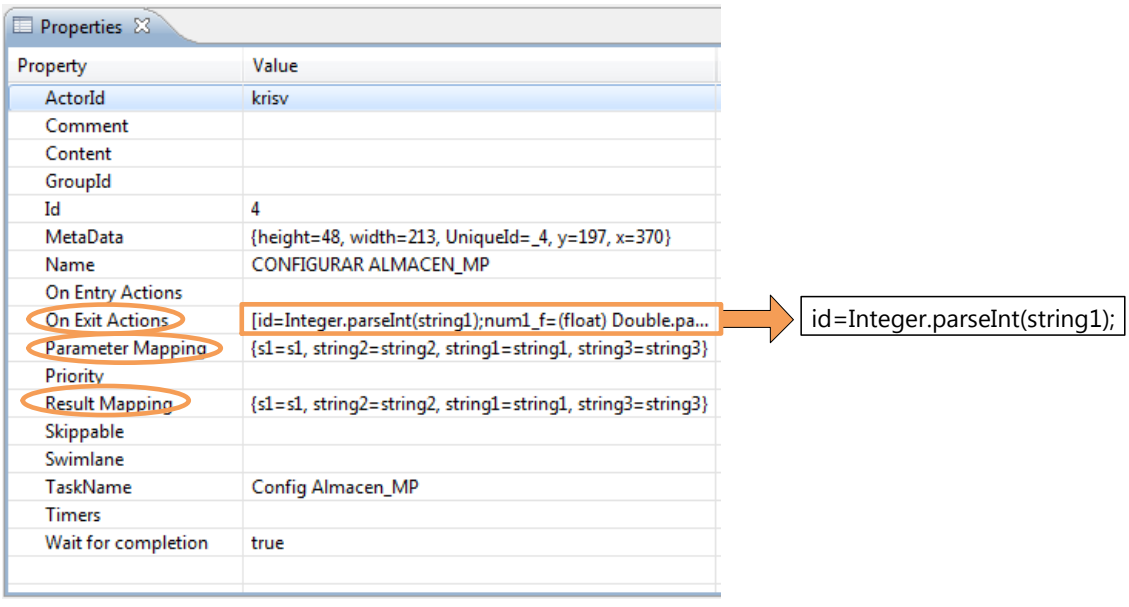


Figura 37: Propiedades principales de las tareas de usuario y de su interacción con las variables del proceso.

A través de los distintos métodos definidos como parseInt, parseFloat, parseBoolean, etc., se realizarán las correspondientes asociaciones entre los datos recogidos por las plantillas y las variables internas del proceso en ejecución.

Una vez establecidas las correspondencias necesarias para el proceso activo se procederá al registro de las variables obtenidas en la base de datos del sistema. Esta acción permanece reflejada en cada uno de los nodos automáticos denominados REGISTRO, incluidos a la salida de cada una de las tareas humanas. Las principales acciones llevadas a cabo por estas tareas automáticas consistirán en establecer la secuencia de sentencias necesarias para estructurar convenientemente cada una de las tablas almacenadas en la base de datos. En el cuadro representado a continuación se detalla, a modo de ejemplo, cómo sería la programación de uno de estos nodos REGISTRO.

```
int a=0;
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
try {        // Cargamos el controlador JDBC
            Class.forName("org.hsqldb.jdbcDriver").newInstance();
}
catch (Exception ex){
            System.err.println("Se pa producido un error al cargar el controlador JDBC");
            return;
}
try{
            conn = DriverManager.getConnection(url_db,"sa","");
            st  = conn.createStatement();
}
catch (Exception ex){
            System.out.println("PROBLEMA CONEXION");
}
try {
            st.executeUpdate( "insert into LOTE (id_lote) values (0)");
             st.executeUpdate( "insert into configuracion values ('capacidad_cuba', "+num1_f+" )");
            a=0;
            while (id>=a) {
                        st.executeUpdate( "insert into cuba (id_cuba,capacidad,estado,tipo,id_lote) values
                        ("+a+","+num1_f+", true, 'F',0)");
                        a++;
            }
}
catch (Exception e) {
            System.out.println(e);
}
try {
            st.executeUpdate("SHUTDOWN");
} catch (Exception e) {
             e.printStackTrace();
}
```

En el caso concreto expuesto en el cuadro anterior se muestra la secuencia de acciones necesaria para definir, en la tabla nombrada como CUBA, un número de entradas equivalente al número fijado por el administrador para las cubas de reposo en la bodega.

Tras completar convenientemente cada una de las etapas, tanto de consulta como de registro, incluidas en el subproceso de configuración, se pasará al nodo de REGISTRO DE VARIABLES. Este nodo será el encargado de realizar los últimos detalles de configuración antes de dar por completado el subproceso de creación. En su interior, representado en el cuadro suministrado a continuación, se definirán las entradas establecidas para las tablas BARRICA y BOTELLAS, así como incluir las variables globales de ejecución a las tablas INDICE, VARIABLES_BOOLEANAS y CONFIGURACION.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
float cap_litros=0;
int a=0;
float mini_lotes=0;
int resto=0;
float resto_l=0;
float litros=0;
try {       // Cargamos el controlador JDBC
            Class.forName(driver_db).newInstance();
            conn = DriverManager.getConnection(url_db,"sa","");
            st  = conn.createStatement();
}
catch (Exception ex){
            System.err.println(ex);
}
try{
            cap_litros=capacidad_botella*num_botellas+capacidad_barrica*num_barricas;
            mini_lotes=num_botellas/lote_botellas;
            id=0;
            resto=num_botellas-(id*lote_botellas);
            while (resto>lote_botellas) {
                        mini_lotes=mini_lotes-1;
                        id++;
                        resto=num_botellas-(id*lote_botellas);
            }
            resto=num_botellas-(id*lote_botellas);
            resto_l=resto*capacidad_botella;
             litros=lote_botellas*capacidad_botella;
            st.executeUpdate( "insert into botellas (id_ref, cantidad_b_max, cantidad_l_max,
            id_lote_actual, id_lote_siguiente, tipo,estado,testigo_crianza,testigo_reserva) values
            (0,"+resto+","+resto_l+",0,0, 'F', true, false, false)");
            a=1;
            while (id>=a) {
                        st.executeUpdate( "insert into botellas (id_ref, cantidad_b_max ,cantidad_l_max,
                        id_lote_actual, id_lote_siguiente, tipo,estado,testigo_crianza,testigo_reserva) values
                        ("+a+","+lote_botellas+","+litros+",0,0, 'F', true, false, false)");
                        a++;
            }
```

```
            a=0;
            id=num_barricas;
            while (id>a) {
                        st.executeUpdate( "insert into barrica (id_barrica, capacidad_max, id_lote_actual,
                        id_lote_siguiente, tipo,estado, testigo) values ("+a+","+capacidad_barrica+",0,0, 'F',
                        true, false)");
                        a++;
}
catch (Exception e) {
            System.out.println(e);
}
try {  // Se insertan los elementos recogidos en la configuracion del sistema
            st.executeUpdate( "insert into configuracion values ('cantidad_lote', "+capacidad_lote+" )");
            st.executeUpdate( "insert into configuracion values ('capacidad_botella',
            "+capacidad_botella+" )");
            st.executeUpdate( "insert into configuracion values ('capacidad_barrica',
            "+capacidad_barrica+" )");
            st.executeUpdate( "insert into configuracion values ('capacidad_litros', "+cap_litros+" )");
            st.executeUpdate( "insert into configuracion values ('relacion', "+relacion+" )");
            st.executeUpdate( "insert into configuracion values ('procesando_litros', 0)");
            st.executeUpdate( "insert into configuracion values ('t_env_joven', "+t_env_joven+" )");
            st.executeUpdate( "insert into configuracion values ('t_env_crianza', "+t_env_crianza+" )");
            st.executeUpdate( "insert into configuracion values ('t_env_reserva', "+t_env_reserva+" )");
            st.executeUpdate( "insert into configuracion values ('t_rep_joven', "+t_rep_joven+" )");
            st.executeUpdate( "insert into configuracion values ('t_rep_crianza', "+t_rep_crianza+" )");
            st.executeUpdate( "insert into configuracion values ('t_rep_reserva', "+t_rep_reserva+" )");
            st.executeUpdate( "insert into configuracion values ('t_cuba', "+t_cuba+" )");

            // Se insertan los elementos necesarios para la tabla indice
            st.executeUpdate( "insert into indice values ('analisis', 1 )");
            st.executeUpdate( "insert into indice values ('pedido', 1 )");
            st.executeUpdate( "insert into indice values ('calidad', 1 )");
            st.executeUpdate( "insert into indice values ('cliente', 1 )");
            st.executeUpdate( "insert into indice values ('crianza', 1 )");
            st.executeUpdate( "insert into indice values ('hoja_d', 1 )");
            st.executeUpdate( "insert into indice values ('inspeccion', 1 )");
            st.executeUpdate( "insert into indice values ('joven', 1 )");
            st.executeUpdate( "insert into indice values ('lote', 1 )");
            st.executeUpdate( "insert into indice values ('id_camion', 1)");
            st.executeUpdate( "insert into indice values ('reserva', 1 )");
            st.executeUpdate( "insert into indice values ('barricas_max', "+num_barricas+" )");
            st.executeUpdate( "insert into indice values ('barricas_libres', "+num_barricas+" )");
            st.executeUpdate( "insert into indice values ('num_botellas', "+num_botellas+" )");
            st.executeUpdate( "insert into indice values ('lote_botellas',"+lote_botellas+" )");

            //Se añaden los elementos necesarios de variables boleanas
            st.executeUpdate( "insert into variables_boleanas values ('almacen_mp', true )");
            st.executeUpdate( "insert into variables_boleanas values ('almacen_pt', true )");
            st.executeUpdate( "insert into variables_boleanas values ('marketing', false )");
            st.executeUpdate( "insert into variables_boleanas values ('crianza', false )");
            st.executeUpdate( "insert into variables_boleanas values ('joven', false )");
            st.executeUpdate( "insert into variables_boleanas values ('lote', false )");
            st.executeUpdate( "insert into variables_boleanas values ('reserva', false )");
            st.executeUpdate( "insert into variables_boleanas values ('v_stock', false )");
            st.executeUpdate( "insert into variables_boleanas values ('barrica', false )");
            st.executeUpdate( "insert into variables_boleanas values ('bodega', false )");
            st.executeUpdate( "insert into variables_boleanas values ('rev', false )");
            st.executeUpdate( "insert into variables_boleanas values ('promocion', false )");
}
```

```
catch (Exception e) {
        e.printStackTrace();
}
          paso=true;
          java.util.Map contentParam = new java.util.HashMap();
          kcontext.setVariable("paso", paso);
try {
          st.executeUpdate("SHUTDOWN");
}
catch (Exception e) {
           e.printStackTrace();
}
```

Una vez cumplimentados todos los pasos definidos por este subproceso, el proceso general estará preparado para continuar con su secuencia normal de ejecución. Esta secuencia le llevará al siguiente nodo del proceso nombrado como CARGAR CONFIGURACION. En él se procederá a volcar los datos generales de la bodega a las variables locales de ejecución del proceso. Todas ellas, que serán replicadas para cada ejecución del proceso, deberán ser inicializadas al comienzo del mismo. Esta acción viene recogida en el siguiente cuadro y marcará el último eslabón antes de comenzar, de forma estricta, con el flujo de elaboración de vino.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
try {  // Cargamos el controlador JDBC
          Class.forName(driver_db).newInstance();
          }
catch (Exception ex){
          System.err.println("Se pa producido un error al cargar el controlador JDBC");
          return;
}
try{
          conn = DriverManager.getConnection(url_db,"sa","");
          st  = conn.createStatement();
}
catch (Exception ex){
          System.out.println("PROBLEMA CONEXION");
}
try {
          rst1 = st.executeQuery("select * from configuracion where nombre='cantidad_lote'");
          rst1.next();
          cantidad_lote=rst1.getFloat("valor");
          rst1 = st.executeQuery("select * from configuracion where nombre='capacidad_barrica'");
          rst1.next();
          capacidad_barrica=rst1.getFloat("valor");
          rst1 = st.executeQuery("select * from configuracion where nombre='capacidad_botella'");
```

```
            rst1.next();
            capacidad_botella=rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='t_env_joven'");
            rst1.next();
            t_env_joven=rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='t_env_crianza'");
            rst1.next();
            t_env_crianza=rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='t_env_reserva'");
            rst1.next();
            t_env_reserva=rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='t_rep_joven'");
            rst1.next();
            t_rep_joven=rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='t_rep_crianza'");
            rst1.next();
            t_rep_crianza=rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='t_rep_reserva'");
            rst1.next();
            t_rep_reserva=rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from variables_boleanas where nombre='rev'");
            rst1.next();
            x=rst1.getBoolean("valor");
}
 catch (Exception e) {
             e.printStackTrace();
}
             paso=true;
            java.util.Map contentParam = new java.util.HashMap();
            kcontext.setVariable("cantidad_lote", cantidad_lote);
            kcontext.setVariable("x", x);
            kcontext.setVariable("capacidad_barrica", capacidad_barrica);
            kcontext.setVariable("capacidad_botella", capacidad_botella);
            kcontext.setVariable("t_env_joven", t_env_joven);
            kcontext.setVariable("t_env_crianza", t_env_crianza);
            kcontext.setVariable("t_env_reserva", t_env_reserva);
            kcontext.setVariable("t_rep_joven", t_rep_joven);
            kcontext.setVariable("t_rep_crianza", t_rep_crianza);
            kcontext.setVariable("t_rep_reserva", t_env_reserva);
try {
            st.executeUpdate( "update variables_boleanas  set valor=false where nombre ='rev'");
            st.executeUpdate("SHUTDOWN");
}
catch (Exception e) {
             e.printStackTrace();
}
```

4.5.2. Etapa de elaboración de vino

Ya terminado tanto el registro como la configuración previos al proceso propio de elaboración de vino, se procederá a registrar una nueva petición de envío. Este nuevo nodo se alcanzará no sin antes comprobar la naturaleza específica de la ejecución. Aunque este

concepto se detallará con mayor profundidad más adelante, sólo debemos recordar que la ejecución del proceso podrá partir de una decisión directa del usuario o de una auto-invocación originada dentro del propio proceso.

Al margen de todo ello, el siguiente paso en la secuencia original será la invocación de la tarea de PETICION DE ENVIO. Dentro de este subproceso se originará la plantilla asociada a aceptar un nuevo envío hacia la bodega. Por medio de esta nueva interacción se deberán asegurar que el nuevo envío cumple con el estado actual de la bodega y con los pedidos en ruta hacia la misma, asegurando así evitar cualquier inconsistencia operacional durante la ejecución. Tal y como se observó para el subproceso de petición de envío, cada vez que se envíe un formulario completo se evalúan las condiciones, repitiendo el proceso de forma reiterada hasta cumplir las condiciones asociadas. A la salida de la interacción humana se procederá a esta comprobación, estimando la viabilidad final de la nueva petición.

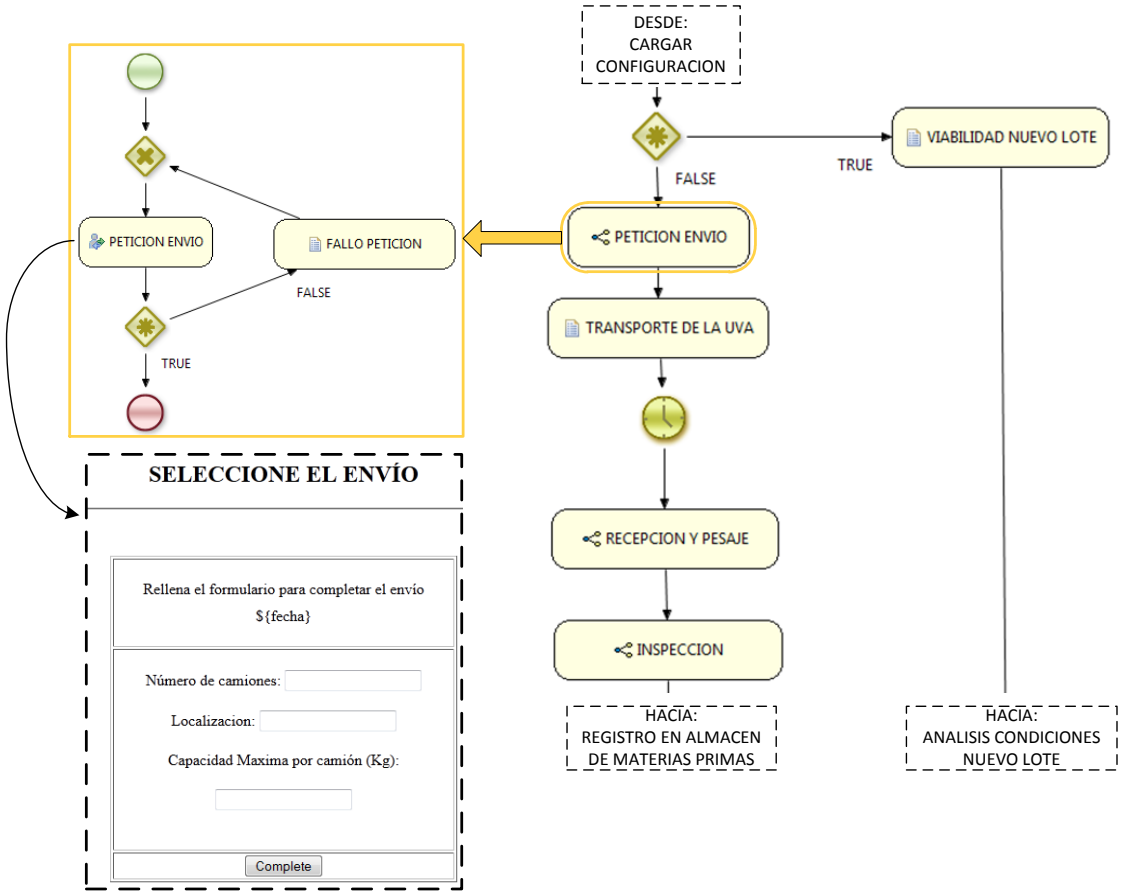


Figura 38: Etapa inicial del proceso de elaboración de vino centrándose en la petición de envío

Tras cumplimentar esta plantilla se habrán especificado los aspectos más relevantes del nuevo envío, como son el número de camiones, el lugar de procedencia o la capacidad de

materia prima por camión. Una vez terminada esta solicitud se supone aceptado el envío y, por tanto, se prosigue con el transporte de la uva. Este periodo vendrá marcado por un tiempo de espera tras el cual se procederá a pesar la partida entrante en la bodega. Para esta tarea se lanzará un nuevo formulario al operario pertinente con el fin de registrar la materia prima entrante.

Una vez terminada esta secuencia, el siguiente paso en el modelo de simulación vendrá marcado por el conocido como proceso de inspección. Dentro de este subproceso, representado en la figura mostrada a continuación, se lanzarán plantillas específicas para cada uno de los controles realizados durante la inspección.

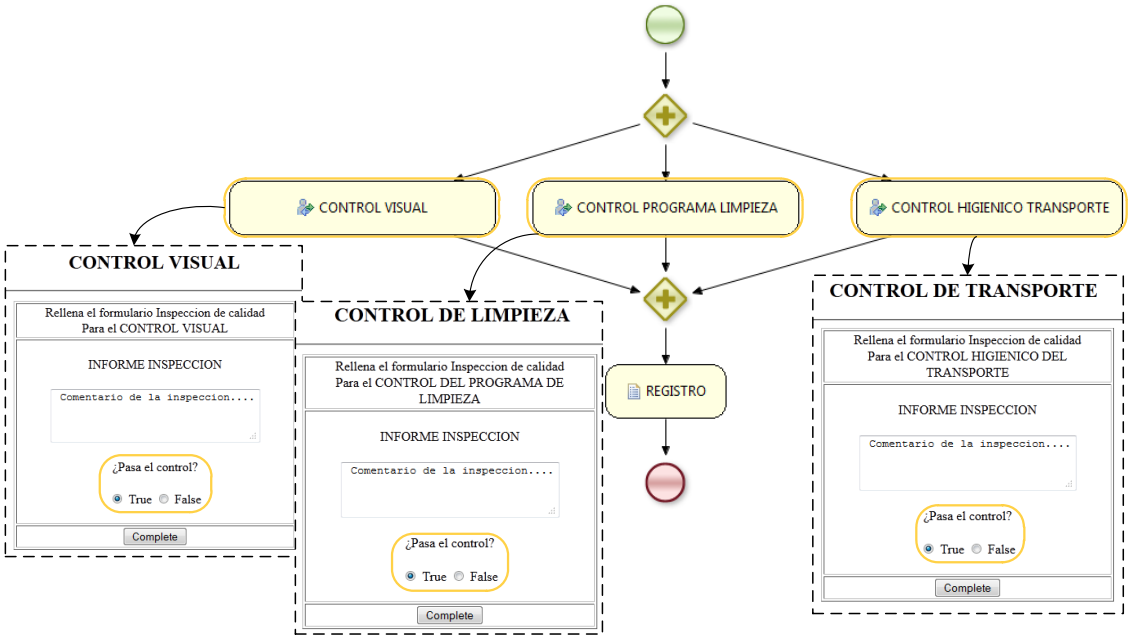


Figura 39: Subproceso de inspección de materia prima y la interacción con el usuario asociada

Para cada uno de los formularios se establecerá un campo a modo de informe o comentario de la inspección así como otro adicional, de tipo booleano, para especificar si la partida superó el control o, por el contrario, debe ser desechada. Toda la información recogida en el subproceso de inspección quedará registrada en la base de datos tras ser procesada en el nodo REGISTRO. Las acciones implementadas en este nodo se muestran en el cuadro de código presentado a continuación.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
int id_inspeccion=0;
try {  // Cargamos el controlador JDBC
            Class.forName("org.hsqldb.jdbcDriver").newInstance();
            conn = DriverManager.getConnection(url_db,"sa","");
            st  = conn.createStatement();
}
catch (Exception ex){
            System.out.println("PROBLEMA CONEXION");
}
      // Se actualiza la variable paso según los valores devueltos por las inspecciones
paso=false;
 if (ctrl_l  &&  ctrl_t  &&  ctrl_v) {
             paso=true;
 }
java.util.Map contentParam = new java.util.HashMap();
.setVariable("paso", paso);
try {
            rst1 = st.executeQuery("select * from indice where nombre='inspeccion'");
            rst1.next();
            id_inspeccion= rst1.getInt("valor");
            st.executeUpdate( "insert into inspeccion (id_insp, control_l,control_t, control_v, com_l, com_t,
            com_v, fecha) values ("+id_inspeccion+",  "+ctrl_l+", "+ctrl_t+", "+ctrl_v+", '"+com_l+"',
            '"+com_t+"', '"+com_v+"', '"+fecha+"');");
            st.executeUpdate( "update pedido set id_inspeccion="+id_inspeccion+" where id_pedido
            ="+id_pedido+";");
            id_inspeccion ++;
            st.executeUpdate( "update indice set valor="+id_inspeccion+" where nombre = 'inspeccion';");
}
catch (Exception e) {
             System.out.println(e);
}
try {
            st.executeUpdate("SHUTDOWN");
}
catch (Exception e) {
             e.printStackTrace();
}
```

El resultado obtenido será el registro ordenado de las inspecciones de entrada realizadas en la bodega así como un seguimiento del inspector que las llevó a cabo. Tras finalizar correctamente este proceso, y evitar así el rechazo del pedido, se considerará la partida entrante apta para ser almacenada en el almacén de la bodega. Tanto la secuencia de entrada en la bodega como la toma de decisiones para la creación de un nuevo lote marcarán, por tanto, las siguientes etapas en el proceso general de negocio, ambas representadas en el diagrama de modelado presentado a continuación.

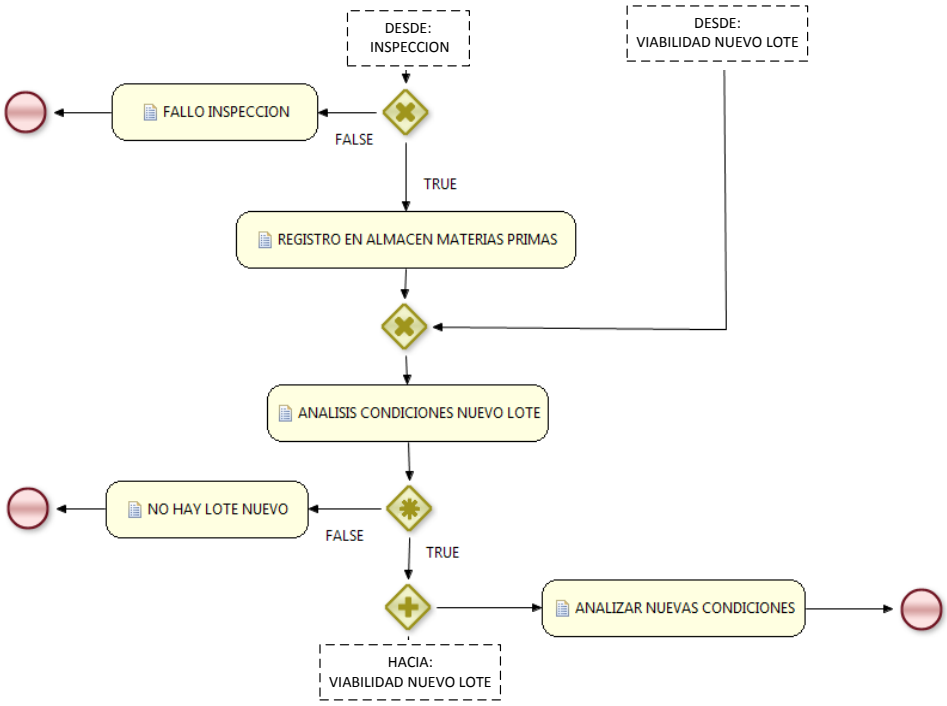


Figura 40: Etapa de registro en almacén y evaluación de condiciones para un nuevo lote

Durante la tarea de REGISTRO EN ALMACEN DE MATERIAS PRIMAS no sólo se registrarán la materia prima entrante, sino que también se valorará la posibilidad de saturación a la entrada, levantando en este caso un testigo para el cese a la hora de aceptar nuevos pedidos en la bodega. La programación de esta tarea automática aparece representada en el cuadro mostrado a continuación.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
float cantidad_pedido=0f;
int i=0;
float a=0f;
float c=0f;
float cantidad_almacen=0f;
int id=0;
float relacion=0f;
boolean pie_cuba=false;
boolean cond1=false;
boolean cond2=false;
float litros=0;
float l_barricas=0;
int barricas=0;
paso=false;
 try {  // Se realiza la conexión
            Class.forName("org.hsqldb.jdbcDriver").newInstance();
            conn = DriverManager.getConnection(url_db,"sa","");
            st  = conn.createStatement();
} catch (Exception ex) {
            System.out.println("PROBLEMA CONEXION");
}
```

```
try {
		rst1 = st.executeQuery("select * from pedido where id_pedido="+id_pedido);
		rst1.next();
		cantidad_pedido= rst1.getFloat("cantidad");
		System.out.println("SE ALMACENA EL PEDIDO id= "+id_pedido+"  cantidad(Kg)=
		"+cantidad_pedido);
}
catch (Exception e) {
		System.out.println(e);
}
try {	// DISTRIBUYE EL PEDIDO ENTRANTE ENTRE LOS DISTINTOS ALMACENES
		// Se establece el camión entrante como camión entregado (estado=false)
		st.executeUpdate( "update camiones set estado=false, carga="+cantidad_pedido+" where
		id_ref ="+id_camion);
		rst1 = st.executeQuery("select * from almacen_mp");

		while (rst1.next()){
			if (cantidad_pedido>0f) {
				a= rst1.getFloat("esp_libre");
				i= rst1.getInt("id_almacen");
				if (a>0f) {
					if(a>cantidad_pedido) {
						System.out.println("Se descarga en el almacen ID= "+i);
						a= a-cantidad_pedido;
						st.executeUpdate( "update almacen_mp set estado=true,
						esp_libre="+a+" where id_almacen ="+i);
						cantidad_pedido=0;
					}
					else {
						System.out.println(" El almacen ID= "+i+"  esta lleno");
						cantidad_pedido= cantidad_pedido-a;
						st.executeUpdate( "update almacen_mp set estado=false,
						esp_libre=0 where id_almacen ="+i);
					}
				}
			}
		}
}
catch (Exception e) {
		System.out.println(e);
}
try {  //Si, por una circunstancia excepcional, no hay espacio suficiente ACTUALIZAMOS almacen=false
(indica que los almacenes de materias prima están LLENOS), poniendo el resto del pedido en un
almacen ficticio arrendado con id_almacen=0
		if (cantidad_pedido > 0f) {
			System.out.println(" Se ha arrendado un almacen extra ID= 0");
			st.executeUpdate( "update variables_boleanas set valor=false where nombre =
			'almacen_mp';");
			st.executeUpdate( "insert into almacen_mp values (0,'almacen arrendado', 0, false,
			"+cantidad_pedido+");");
		}
}
catch (Exception e) {
		System.out.println(e);
}
try {
		st.executeUpdate("SHUTDOWN");
}
catch (Exception e) {
		e.printStackTrace();
}
```

El producto de esta ejecución provocará el almacenamiento ordenado en cada uno de los almacenes definidos para las materias primas de la bodega, o bien, y bajo condiciones excepcionales de organización, el arrendamiento de un almacén auxiliar. Una vez completado este registro, el sistema se encontrará en situación de gestionar la creación de un nuevo lote. Las consideraciones necesarias para esta creación permanecen claramente recogidas en la programación del nodo ANALISIS CONDICIONES NUEVO LOTE presentada en el siguiente cuadro.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
float cantidad_pedido=0f;
int i=0;
float a=0f;
float c=0f;
float cantidad_almacen=0f;
int id=0;
float relacion=0f;
boolean pie_cuba=false;
boolean cond1=false;
boolean cond2=false;
float litros=0;
float l_barricas=0;
int barricas=0;
paso=false;
try {  // Se realiza la conexión
        Class.forName("org.hsqldb.jdbcDriver").newInstance();
        conn = DriverManager.getConnection(url_db,"sa","");
        st  = conn.createStatement();
}
catch (Exception ex) {
        System.out.println("PROBLEMA CONEXION");
}
try { // SE CALCULA LA CANTIDAD TOTAL DE UVA ALMACENADA EN TODOS LOS ALMACENES
        rst1 = st.executeQuery("select * from almacen_mp");
        cantidad_almacen=0f;
        while (rst1.next()){
                a= rst1.getFloat("esp_libre");
                c=rst1.getFloat("c_max");
                _almacen= cantidad_almacen+c-a;
        }
}
catch (Exception e) {
        System.out.println(e);
}      // SE EVALUAN LAS CONDICIONES NECESARIAS PARA CREAR UN NUEVO LOTE
try { // CONDICION 1: DEBE HABER ALMACENADA SUFICIENTE MATERIA PRIMA PARA LLENAR UNA
CUBA
        cond1=false;
        if (cantidad_almacen>=cantidad_lote) {
                System.out.println(" Hay materias primas suficientes");
                cond1=true;
        }
```

```
            // Se crea un lote
            rst1 = st.executeQuery("select * from indice where nombre='lote'");
            rst1.next();
            id_lote = rst1.getInt("valor");

        // CONDICION 2: DEBE HABER ALGUNA CUBA LIBRE PARA INICIAR EL PROCESO
            if(cond1) {
                    st.executeUpdate( "insert into lote (id_lote, cantidad_uva) values ("+id_lote+",
                    "+cantidad_lote+")");
                    rst1 = st.executeQuery("select * from cuba");
                    cond2=false;
                    while (rst1.next()){
                            paso=rst1.getBoolean("estado");
                            if (cond2==false && paso) {  // Se reserva la CUBA
                                    System.out.println("Hay Cubas suficientes");
                                    id_cuba=rst1.getInt("id_cuba");
                                    cond2=true;
                                    st.executeUpdate( "update cuba set estado=false,
                                    id_lote="+id_lote+" where id_cuba ="+id_cuba);
                            }
                    }
                    if (cond2==false) {
                            System.out.println(" NO HAY CUBA LIBRE -> NO SE PUEDE CREAR NUEVO
                            LOTE");
                            st.executeUpdate( "delete from lote where id_lote ="+id_lote);
                    }
            }
    // CONDICION 3: DEBE EXISTIR SUFICIENTES BARRICAS PARA SOPORTAR (DE FORMA ESTIMADA) LOS
    LITROS DE VINO RESULTANTES DEL PASO POR LA CUBA
            rst1 = st.executeQuery("select * from configuracion where nombre='relacion'");
            rst1.next();
            relacion = rst1.getFloat("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='capacidad_barrica'");
            rst1.next();
            l_barricas = rst1.getInt("valor");
            rst1 = st.executeQuery("select * from configuracion where nombre='cantidad_lote'");
            rst1.next();
            cantidad_lote= rst1.getFloat("valor");
            litros=relacion*cantidad_lote;
            litros=  litros/l_barricas;
            barricas= (int) litros;
            if (litros-barricas>0) {
                    barricas++;
            }
            if(cond1 && cond2 && id_cuba!=0) {
                    st.executeUpdate( "update lote set id_cuba="+id_cuba+" where id_lote="+id_lote);
                    rst1 = st.executeQuery("select * from barrica where id_lote_actual=0 AND
                    id_lote_siguiente=0 ");
                    while (rst1.next() && barricas >0 && id_cuba!=0) {
                            // Si existe alguna BARRICA libre y no está reservada
                            id = rst1.getInt("id_barrica");
                            st.executeUpdate( "update barrica set id_lote_siguiente="+id_lote+" where
                            id_barrica="+id);
                            barricas--;
                            System.out.println(" BARRICA LIBRE RESERVADA ID= "+id);
                    }
```

```
                    if(barricas>0 && id_cuba!=0) {
                            // Si existe alguna BARRICA ocupada pero que dentro de t_cuba esté libre
                            rst1 = st.executeQuery("select * from barrica where id_lote_siguiente=0 AND
                            testigo=true");
                            while (rst1.next() && barricas>=0)  {
                                    id = rst1.getInt("id_barrica");
                                    st.executeUpdate( "update barrica set id_lote_siguiente="+id_lote+"
                                    where id_barrica="+id);
                                    barricas--;
                                    System.out.println(" BARRICA OCUPADA RESERVADA ID= "+id);
                            }
                    }
                    java.util.Map contentParam = new java.util.HashMap();
                    if (barricas>0 && id_cuba!=0) { //Si no existen barricas suficientes se liberan las
barricas reservadas, la cuba reservada y la entrada creada en la tabla lote
                            System.out.println(" NO HAY BARRICAS SUFICIENTES -> NO SE PUEDE
                            CREAR NUEVO LOTE");
                            paso=false;
                            st.executeUpdate( "delete from lote where id_lote="+id_lote);
                            rst1 = st.executeQuery("select * from barrica where id_lote_siguiente =" +
                            id_lote);
                            while (rst1.next()){
                                    id = rst1.getInt("id_barrica");
                                    st.executeUpdate( "update barrica set id_lote_siguiente=0 where
                                    id_barrica="+id);
                            }
                            st.executeUpdate( "update cuba set id_lote=0, estado=true where
                            id_cuba="+id_cuba);
                            st.executeUpdate( "delete from lote where id_lote ="+id_lote);
                    }
                    else  { // Se hen cumplido las tres condiciones anteriores -> Se establece el nuevo
lote y se incluye como entrada en la tabla LOTE
                            System.out.println("SE CUMPLEN TODAS LAS CONDICIONES");
                            paso=true;
                            kcontext.setVariable("id_lote", id_lote);
                            // Se actualiza el valor de id_lote en la tabla INDICE
                            id_lote++;
                            st.executeUpdate( "update indice set valor="+id_lote+" where nombre
                            ='lote'");
                            .executeUpdate( "update variables_boleanas  set valor=true where nombre
                            ='rev'");
                    }
            }
}
catch (Exception e) {
            e.printStackTrace();
}
try {
            kcontext.setVariable("paso", paso);
            kcontext.setVariable("id_cuba", id_cuba);
            st.executeUpdate("SHUTDOWN");
            }
catch (Exception e) {
            e.printStackTrace();
}
```

En este punto de la ejecución se lleva a cabo una de las decisiones más importantes, mandar a producción un nuevo lote de materia prima o, por el contrario, dar por finalizada la ejecución. En el primer caso, la secuencia seguirá con las bien conocidas etapas de elaboración mientras que en el segundo caso se concluirá el proceso a la espera de cumplir, en posteriores ejecuciones, las condiciones establecidas en este nodo.

Continuando con la secuencia esperada de ejecución, el lote será creado procediendo de este modo a la elaboración secuencial del producto final. Este periodo vendrá marcado en primer lugar por el encubado. Para la interacción con el operario a la hora de encubar las pastas procedentes del estrujado de la uva se ha optado por la plantilla expuesta en el esquema de la siguiente figura. En ella se puede ver cómo se facilita toda la información relevante al usuario por medio de la edición dinámica de elementos tipo string. Estos elementos serán convenientemente rellenados y presentados de forma cómoda a través de la misma interfaz utilizada por los formularios. Entre la información suministrada se pueden observar detalles relacionados con el identificador de lote, ubicación en el almacén, la cantidad y, por supuesto, la cuba para el reposado.

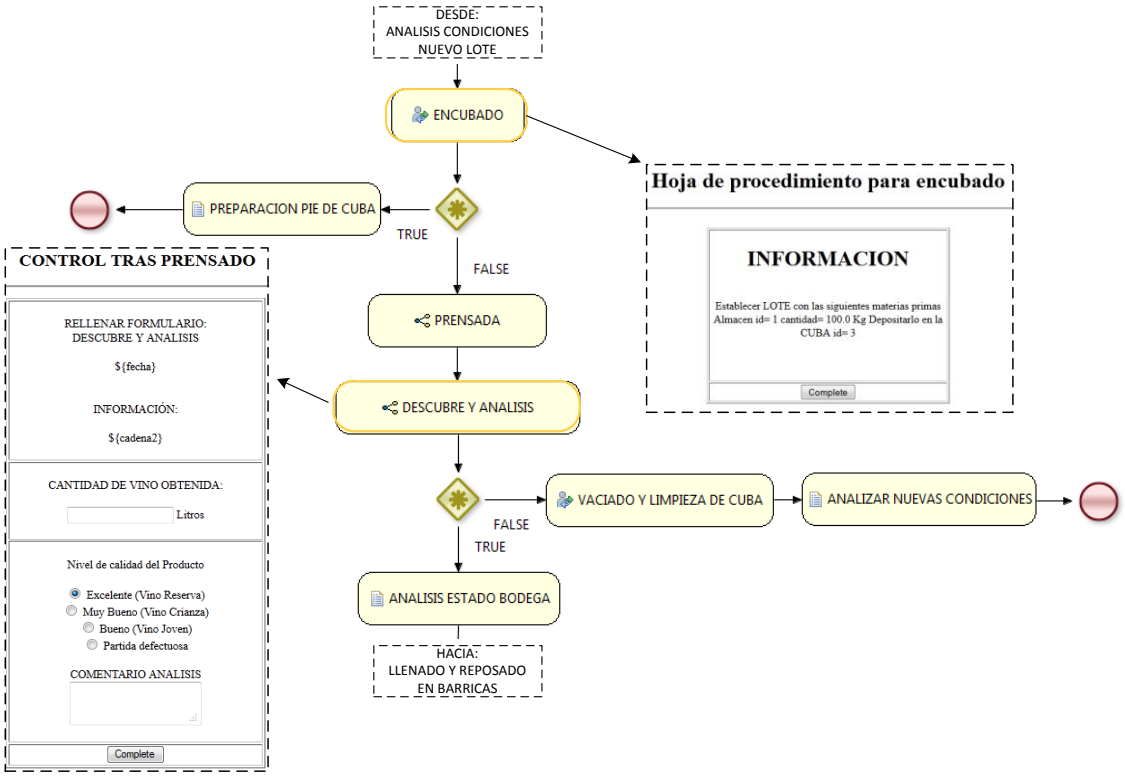


Figura 41: Etapa de encubado y descubre del vino en elaboración

Tras completar todas estas etapas, y transcurrido además el tiempo estipulado para la maceración en cuba, se realizará el descubre y análisis del producto. Para llevar a cabo esta

tarea se implementará un formulario asociado al análisis donde se incluirá, entre otras cosas, detalles específicos de la futura calidad del vino. Una vez completada esta inspección, y registrada toda la información resultante en la tabla correspondiente de la base de datos, se procederá a iniciar otro de los puntos decisivos en la gestión de este proceso de negocio, el denominado ANALISIS DEL ESTADO DE LA BODEGA.

Para llevar a cabo esta comprobación, y proseguir así con la ejecución general del proceso, se ejecutarán una serie acciones programadas que se encuentran reflejadas en el cuadro de código presentado a continuación.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
String calidad;
int id=0;
int b=0;
float litros;
float a=0;
int botellas=0;
try {  // Se realiza la conexion
        Class.forName("org.hsqldb.jdbcDriver").newInstance();
        conn = DriverManager.getConnection(url_db,"sa","");
        st  = conn.createStatement();
}
catch (Exception ex) {
        System.out.println("PROBLEMA CONEXION");
}
try {   rst1 = st.executeQuery("select * from lote where  id_lote="+id_lote);
        rst1.next();
        calidad= rst1.getString("tipo");
        litros= rst1.getFloat("cantidad_vino");
        if  (calidad.equals("J")){
                botellas=0;
        }
        else{   a=litros/capacidad_botella;
                botellas= (int) a;
        }
        if  (calidad.equals("C")) {
                // Se reserva el sitio en la bodega para el envejecimiento de Vino Crianza
                System.out.println("Vino de Calidad TIPO= "+ calidad);
                rst1 = st.executeQuery("select * from botellas where  id_lote_actual=0 AND
                id_lote_siguiente=0");
                while (rst1.next() && botellas>0) { // Se evalúa el epacio que está completamente libre
                        id= rst1.getInt("id_ref");
                        b=rst1.getInt("cantidad_b_max");
                        st.executeUpdate( "update botellas set id_lote_siguiente="+id_lote+" where
                        id_ref ="+id);
                        if (botellas>=b) {
                                botellas=botellas-b;
                        }
                        if (b>botellas && botellas>0) {
                                botellas=0;
                        }
                }
```

```
		if (botellas>0) {
			// Si todavía no se ha reservado suficiente espacio ->  Se evalúa el epacio que
			estará lbre una vez se cumpla el tiempo de estancia en barrica
			rst1 = st.executeQuery("select * from botellas where  testigo_crianza=true
			AND id_lote_siguiente=0");
			while (rst1.next() && botellas>0) {
				// Espacio que quedará libre una vez se pase el tiempo en barrica
				id= rst1.getInt("id_ref");
				b=rst1.getInt("cantidad_b_max");
				st.executeUpdate( "update botellas set id_lote_siguiente="+id_lote+"
				where id_ref ="+id);
				if (botellas>=b) {
					botellas=botellas-b;
				}
				if (b>botellas && botellas>0) {
					botellas=0;
				}
			}
		}
	}
	if  (calidad.equals("R")) {
		// Se reserva el sitio en la bodega para el envejecimiento de Vino Crianza
		System.out.println("Vino de Calidad TIPO= "+ calidad);
		rst1 = st.executeQuery("select * from botellas where  id_lote_actual=0 AND
		id_lote_siguiente=0");
		while (rst1.next() && botellas>0) {
			// Se evalúa el epacio que está completamente libre
			id= rst1.getInt("id_ref");
			b=rst1.getInt("cantidad_b_max");
			st.executeUpdate( "update botellas set id_lote_siguiente="+id_lote+" where
			id_ref ="+id);
			if (botellas>=b) {
				botellas=botellas-b;
			}
			if (b>botellas && botellas>0) {
				botellas=0;
			}
		}
		if (botellas>0) {
			// Si todavía no se ha reservado suficiente espacio ->  Se evalúa el epacio que
			estará lbre una vez se cumpla el tiempo de estancia en barrica
			rst1 = st.executeQuery("select * from botellas where  testigo_reserva=true
			AND id_lote_siguiente=0");
			while (rst1.next() && botellas>0) {
				// Espacio que quedará libre una vez se pase el tiempo en barrica
				id= rst1.getInt("id_ref");
				b=rst1.getInt("cantidad_b_max");
				st.executeUpdate( "update botellas set id_lote_siguiente="+id_lote+"
				where id_ref ="+id);
				if (botellas>=b) {
					botellas=botellas-b;
				}
				if (b>botellas && botellas>0) {
					botellas=0;
				}
			}
		}
```

```
                    if ( botellas >0) {
                            // No hay suficiente sitio en la bodega para Vino Reserva-> Se plantea
                            degradar el Vino Reserva a Vino Crianza
                            rst1 = st.executeQuery("select * from botellas where  testigo_reserva=true
                            AND id_lote_siguiente=0");
                            while (rst1.next() && botellas>0)    {
                                    // Espacio que quedará libre una vez se pase el tiempo en barrica
                                    id= rst1.getInt("id_ref");
                                    b=rst1.getInt("cantidad_b_max");
                                    st.executeUpdate( "update botellas set id_lote_siguiente="+id_lote+"
                                    where id_ref ="+id);
                                    if (botellas>=b) {
                                            botellas=botellas-b;
                                    }
                                    if (b>botellas && botellas>0) {
                                            botellas=0;
                                    }
                            }
                            st.executeUpdate( "update botellas  tipo='C', t_env="+t_env_crianza+",
                            t_rep="+t_rep_crianza+" where id_lote ="+id_lote);
                            System.out.println("Vino degradado a calidad Crianza");
                    }
            }
            if ( botellas >0) {
                    // No hay suficiente sitio en la bodega -> Se libera el espacio reservado y se degrada
                    el vino a calidad Joven
                    rst1 = st.executeQuery("select * from botellas where  id_lote_siguiente="+id_lote);
                    while (rst1.next()){
                            id= rst1.getInt("id_ref");
                            st.executeUpdate( "update botellas  id_lote_siguiente=0 where id_ref="+id);
                    }
                    st.executeUpdate( "update lote  tipo='J', t_env="+t_env_joven+", t_rep= "
                    +t_rep_joven+"  where id_lote ="+id_lote);
                    System.out.println("Vino degradado a calidad Joven");
            }
            else{
                    // Todo está correcto, hay sitio suficiente en la bodega para albergar Vino con estas
                    condiciones
                    System.out.println("Hay sitio en la Bodega, Se procede al llenado de Barricas");
            }
    }
    catch (Exception e) {
            System.out.println(e);
    }
    try {
            st.executeUpdate("SHUTDOWN");
    }
    catch (Exception e)  {
            e.printStackTrace();
    }
```

En este preciso momento de la ejecución se llevará a cabo la toma de decisiones asociada a la aceptación o degradación de la futura calidad del vino. Esta elección, comentada

brevemente al inicio del apartado, se basará en el análisis del estado interno de la bodega, ya sea en las barricas de crianza o en la bodega de envejecimiento. El resultado de esta tarea devolverá la calidad final del lote en ejecución, estableciendo además su futura secuencia de nodos y tiempos.

Independientemente de la calidad del lote, el proceso proseguirá con su ejecución, efectuando para ello el correspondiente llenado de las barricas de crianza. Una vez dentro del subproceso denominado LLENADO Y REPOSADO EN BARRICAS se procederá con la secuencia representada en la siguiente figura.

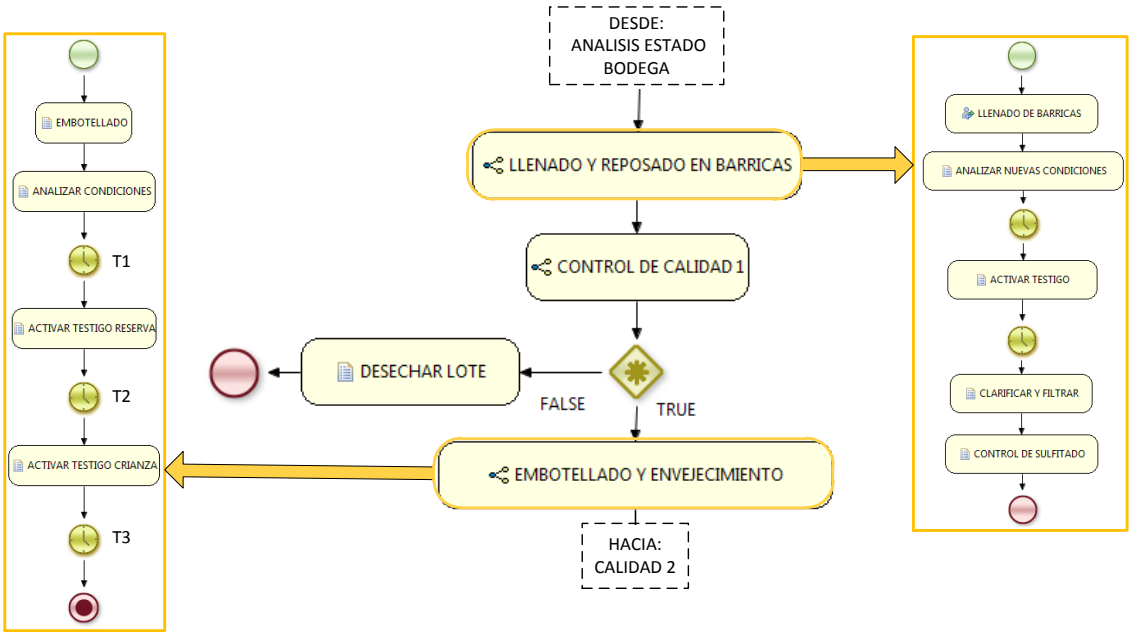


Figura 42: Etapa para el llenado y el reposo en barrica así como el posterior embotellado y envejecimiento

Como se puede observar, la primera operación que se realiza hace referencia a una llamada para la interacción con el operario de la bodega. A través de esta interacción se pretende facilitar la información de control al usuario para completar el trasvase de producto con la mayor brevedad y eficiencia. Tras esta rutina de interacción, la cuba desde donde se hubiera transferido el lote en ejecución quedará totalmente liberada, produciéndose de este modo la necesidad de una llamada en paralelo a la ejecución general del proceso. Como cabe esperar, esta acción resulta necesaria tras toda liberación de recursos en el sistema, por lo que de ahora en adelante la llamada a la ejecución en paralelo para el proceso general permanecerá recogida bajo el script automático ANALIZAR NUEVAS CONDICIONES.

Para llevar a cabo esta acción se ejecutarán la secuencia de comandos descritos presentados en el siguiente cuadro, creándose de este modo un hilo independiente para la ejecución del sistema.

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
kbuilder.add(ResourceFactory.newClassPathResource("main.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("general.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("inspeccion.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("peticion.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("recepcion.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("prensada.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("busqueda.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("descubre.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("llenado.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("calidad.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("embotellado.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("marketing.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("distribucion.bpmn"), ResourceType.BPMN2);
kbuilder.add(ResourceFactory.newClassPathResource("configurar.bpmn"), ResourceType.BPMN2);
KnowledgeBase kbase = kbuilder.newKnowledgeBase();
StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();
ksession.getWorkItemManager().registerWorkItemHandler("Human Task", new WSHumanTaskHandler());
ksession.startProcess("main");
```

Aunque el código representado en el cuadro anterior será explicado con mayor profundidad en el siguiente aparatado, resulta necesario hacer una breve descripción de los elementos más importantes que aparecen en él. En primer lugar, tanto la creación de un elemento KnowledgeBuilder como la asociación de los distintos modelos en BPMN definidos para este proceso marcarán los cimientos para la creación de una nueva sesión de ejecución. Para completar esta definición será necesario además registrar el manejador de tareas humanas, definido en este caso como WSHumanTaskHandler.

El resultado obtenido de esta implementación será la aparición de un nuevo hilo, completamente ajeno al que hizo la llamada inicial. De este modo, el hilo inicial continuará indiferente su ejecución secuencial mientras que el nuevo hilo realizará un testeo sobre la viabilidad de un nuevo lote. Como se puede esperar, la nueva ejecución evaluará paso a paso las distintas etapas descritas en este apartado salvo el momento en que se comprueba el tipo de ejecución que se está llevando a cabo. En este momento se dividirá del camino principal, pasando a una etapa secundaria que le llevará directamente a la evaluación de las condiciones para un nuevo lote. Para hacer distinciones entre los tipos de ejecución se hará uso de variables adicionales contenidas en la base de datos del sistema.

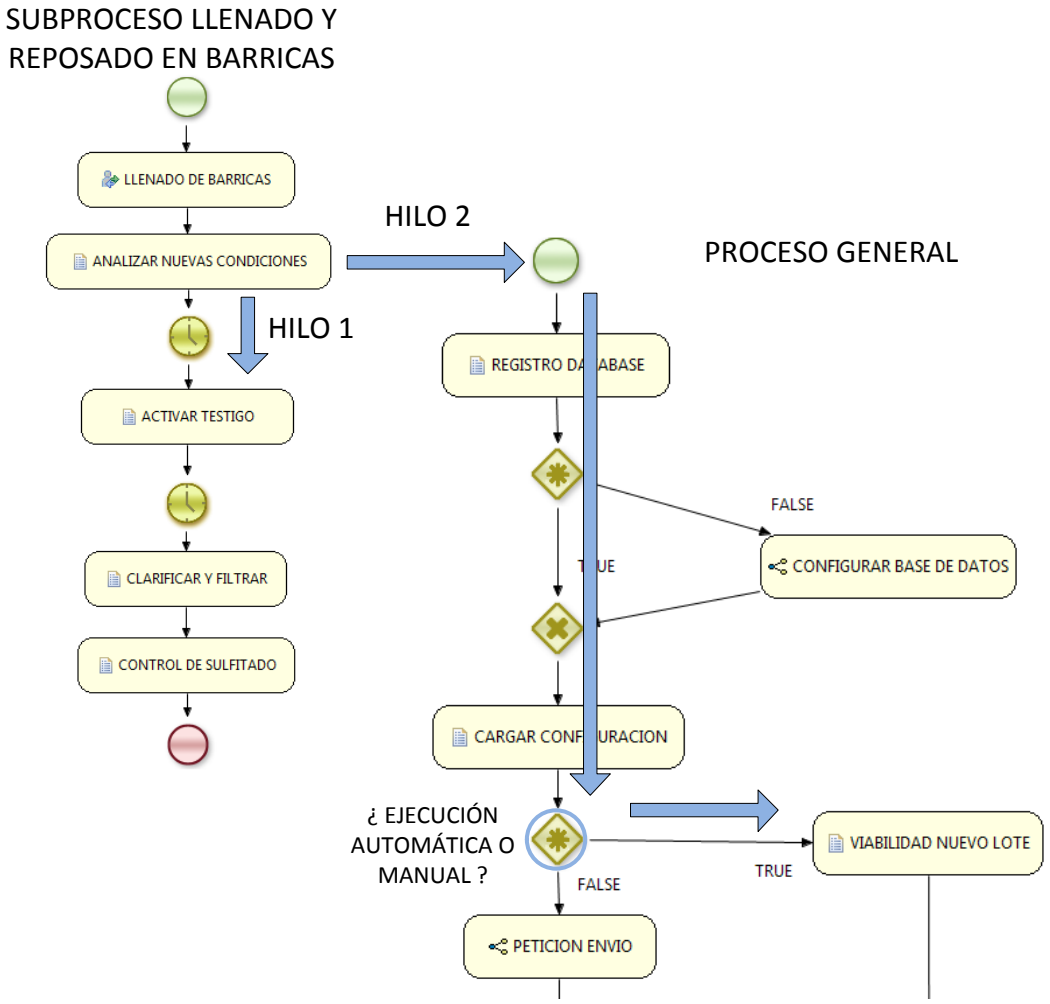


Figura 43: Esquema conceptual sobre el lanzamiento de un nuevo hilo ante la liberación de recursos en la bodega

Al margen de esta aclaración, la secuencia inicial del sistema continuará con la ejecución ordenada de tareas. De este modo, durante su periodo de crianza se actualizarán los diferentes testigos tras pasar el tiempo establecido para ello. Una vez completada esta etapa se iniciará el control de calidad del producto resultante, realizando para ello los análisis físicos y químicos establecidos para ello. Este subproceso, definido convenientemente en apartados anteriores de la memoria, registrará los resultados del estudio en las correspondientes entradas definidas en la base de datos del sistema.

Tras completar satisfactoriamente el primer análisis de calidad del producto, el lote procederá a ser embotellado y trasladado a la bodega de envejecimiento. Cabe destacar que el trasvase del producto a la zona de embotellado y reposo implicará consecuentemente una nueva liberación de recursos. Por este motivo, en esta parte del proceso se añade también otra rutina automática para la ejecución de un nuevo hilo para la comprobación sobre la viabilidad de un nuevo lote. Para el caso específico de la bodega de envejecimiento, el lote en

ejecución permanecerá un tiempo igual a la suma de los tiempos t1, t2 y t3, siendo éste nulo para el vino con calidad joven. Las acciones definidas para establecer el tiempo requerido para el reposo del producto se encuentran recogidas en el cuadro de código que se muestra a continuación.

```
Connection conn;
String sql;
Statement st;
ResultSet  rst1;
int cantidad_botellas=0;
float a=0f;
float a=0f;
int bot=0;
int id=0;
float t_env=0f;
float litros=0f;
try {  // Se realiza la conexión con la base de datos
        Class.forName("org.hsqldb.jdbcDriver").newInstance();
        conn = DriverManager.getConnection(url_db,"sa","");
        st  = conn.createStatement();
}
catch (Exception ex){  System.out.println("PROBLEMA CONEXION");
}
try {     // Se liberan las barricas que contenían el LOTE
        st.executeUpdate( "update barrica set id_lote_actual=0, cantidad=0, tipo='F', estado=true where
        id_lote_actual ="+id_lote);
        rst1 = st.executeQuery("select * from configuracion where nombre='capacidad_botella'");
        rst1.next();
        capacidad_botella= rst1.getFloat("valor");
        rst1 = st.executeQuery("select * from lote where id_lote="+id_lote);
        rst1.next();
        String calidad= rst1.getString("tipo");
        litros= rst1.getFloat("cantidad_vino");
        cantidad_botellas= (int) ( litros / capacidad_botella);
        b=cantidad_botellas
        rst1.getInt("cantidad_botellas");
        t_env= rst1.getFloat("t_env");
        System.out.println("LOTE ID= "+id_lote+"   Se embotellan "+cantidad_botellas+"  botellas de
        calidad =  "+calidad+", dejandolas envejecer "+t_env);
        if (calidad.equals("C")|| calidad.equals("R"))   {
                st.executeUpdate( "update lote set cantidad_botellas="+cantidad_botellas+" where
                id_lote="+id_lote);
                rst1 = st.executeQuery("select * from botellas where id_lote_siguiente="+id_lote);
                while(rst1.next() || cantidad_botellas>0) {
                        id= rst1.getInt("id_ref");
                        bot= rst1.getInt("cantidad_b_max");
                        if(cantidad_botellas<bot && cantidad_botellas>0) {
                                a= cantidad_botellas * capacidad_botella;
                                st.executeUpdate( "update botellas set id_lote_actual="+id_lote+",
                                id_lote_siguiente=0, testigo_crianza=false, testigo_reserva=false, tipo
                                ='"+calidad+"',  estado= false, cantidad_b_rea l="+ cantidad_botellas
                                + ", cantidad_l_real="+a+" where id_ref ="+id);
                                cantidad_botellas=0;
                        }
                        if(cantidad_botellas>=bot) {
                                a= bot * capacidad_botella;
```

```
                                st.executeUpdate( "update botellas set id_lote_actual="+id_lote+",
                                id_lote_siguiente=0, testigo_crianza=false, testigo_reserva=false, tipo
                                = '"+calidad+"', estado=false,cantidad_b_real="+bot+",
                                cantidad_l_real ="+a+" where id_ref ="+id);
                                cantidad_botellas=cantidad_botellas-bot;
                        }
                }
                if (cantidad_botellas>0)  {  //Hay más botellas que las previamente asignadas en el lote
                        rst1 = st.executeQuery("select * from botellas where id_lote_actual=0 AND
                        id_lote_siguiente=0 ");
                        while(rst1.next() || cantidad_botellas>0) {
                                id= rst1.getInt("id_ref");
                                bot= rst1.getInt("cantidad_b_max");
                                        if(cantidad_botellas<bot && cantidad_botellas>0) {
                                        a= cantidad_botellas * capacidad_botella;
                                        st.executeUpdate( "update botellas set id_lote_actual
                                        ="+id_lote+", id_lote_siguiente=0, testigo_crianza=false,
                                        testigo_reserva=false, tipo ='"+calidad+"',  estado= false,
                                        cantidad_b_rea l="+ cantidad_botellas + ",
                                        cantidad_l_real="+a+" where id_ref ="+id);
                                        cantidad_botellas=0;
                                }
                                if(cantidad_botellas>=bot) {
                                        a= bot * capacidad_botella;
                                        st.executeUpdate( "update botellas set id_lote_actual
                                        ="+id_lote+", id_lote_siguiente=0, testigo_crianza=false,
                                        testigo_reserva =false, tipo = '"+calidad+"', estado =false,
                                        cantidad_b_real="+bot+", cantidad_l_real ="+a+" where
                                        id_ref ="+id);
                                        cantidad_botellas=cantidad_botellas-bot;
                                }
                        if (cantidad_botellas>0){  //Se desechan las botellas sobrantes
                                b= b- cantidad_botellas;
                                st.executeUpdate( "update lote set cantidad_botellas ="+
                                cantidad_botellas + " where id_lote="+id_lote);
                        }
                }
        }
        // Se establece los valores temporales para la activación de los testigos (Bajo la restriccion de
        t_env_joven=0)
        if (calidad.equals("C")){
                if (t_env_crianza <= t_rep_reserva) {
                        if (t_env_crianza > t_rep_crianza) {
                                // Más tiempo envejeciendo en bodega que reposando en barrica
                                t1=0;
                                t2=0;
                                t3=(int) t_env_crianza.intValue();
                        }
                        else  { // Más tiempo reposando en barrica que envejeciendo en bodega
                                t1=0;
                                t2=(int) (t_env_crianza-t_rep_crianza);
                                t3=(int) t_rep_crianza.intValue();
                        }
                }
                else  {
                        t1=(int) (t_env_crianza-t_rep_reserva);
                        t2=(int) (t_rep_reserva-t_rep_crianza);
                        t3=(int) t_rep_crianza.intValue();
                }
```

```
		}
		if (calidad.equals("R")){
			if (t_env_reserva > t_rep_reserva) {
				// Más tiempo envejeciendo en bodega que reposando en barrica
				if (t_env_crianza > t_rep_crianza) {
					// Más tiempo envejeciendo en bodega que reposando en barrica
					t1=0;
					t2=0;
					t3=(int) t_env_reserva.intValue();
				}
				else  {	// Más tiempo reposando en barrica que envejeciendo en bodega

					t1=0;
					t2=(int) (t_env_reserva-t_rep_crianza);
					t3= (int)t_rep_crianza.intValue();
				}
			}
			else  {	// Más tiempo reposando en barrica que envejeciendo en bodega
				t1=(int) (t_env_reserva-t_rep_reserva);
				t2=(int) (t_rep_reserva-t_rep_crianza);
				t3=(int) t_rep_crianza.intValue();
			}
		}
}
catch (Exception e) {
		System.out.println(e);
}
java.util.Map contentParam = new java.util.HashMap();
kcontext.setVariable("t1", t1);
kcontext.setVariable("t2", t2);
kcontext.setVariable("t3", t3);
try {
		st.executeUpdate( "update variables_boleanas  set valor=true where nombre ='rev'");
		st.executeUpdate("SHUTDOWN");
}
catch (Exception e) {
		 e.printStackTrace();
}
```

Una vez completado el proceso de crianza y el consiguiente análisis de calidad y validación se procederá a la última etapa de la simulación. En esta etapa se realizará en primer lugar el etiquetado y empaquetado del producto para su posterior registro y almacenamiento en el almacén de productos terminados. En este almacén se aceptarán secuencialmente la llegada de botellas procedentes de los distintos lotes en ejecución para su venta y distribución final. Para el último subproceso de marketing y distribución se ha optado por utilizar un bucle infinito implementado por un hilo independiente de ejecución. Tras ser lanzado, el hilo quedará marcado por una variable auxiliar, evitando ser lanzado más de una vez. La ejecución del hilo será indefinida hasta cumplir con la venta de la totalidad del producto almacenado en

el almacén. Para tener un control directo del producto a la venta se facilitará cierta información en la plantilla presentada en el navegador.

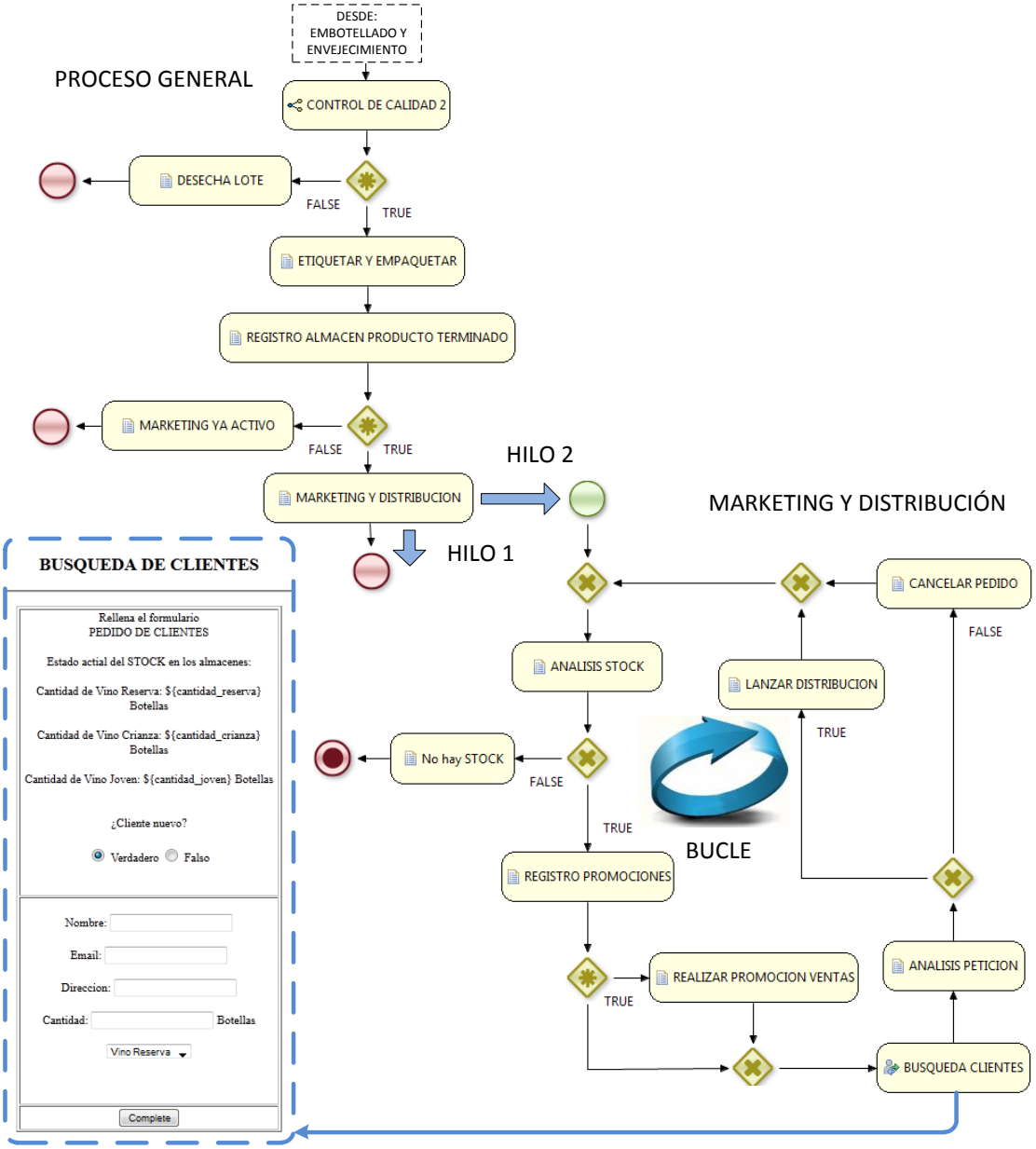


Figura 44: Proceso final de marketing y distribución del producto elaborado

Otro de los aspectos interesantes vendrá dado por la generación de hojas de distribución a los transportistas de la bodega. Este recurso permitirá facilitar a los distribuidores los datos más representativos necesarios para trasladar el producto final a los clientes. Toda esta información será cómodamente presentada al transportista, asegurando de este modo la correcta finalización de la última etapa del proceso descrito en el proyecto.

5. Simulación y resultados

5.1. Creación de una aplicación para ejecución de pruebas

La ejecución y el testeo del proceso en Eclipse se realizará en lenguaje de programación java. Por medio de la implementación y ejecución del archivo definido como ProcessTest.java se iniciará la secuencia definida para el proceso de modelado de vino. A través de la edición directa de este archivo, que define una clase java llamada ProcessTest, será posible especificar los parámetros principales de la ejecución además de dar comienzo al proceso BPMN referenciado en su código. Para el caso concreto comentado a lo largo de esta memoria, el código de ejecución correspondiente se presenta en el cuadro expuesto a continuación.

```
package com.sample;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Date;
import java.util.HashMap;
import org.drools.KnowledgeBase;
import org.drools.builder.KnowledgeBuilder;
import org.drools.builder.KnowledgeBuilderFactory;
import org.drools.builder.ResourceType;
import org.drools.io.ResourceFactory;
import org.drools.logger.KnowledgeRuntimeLogger;
import org.drools.logger.KnowledgeRuntimeLoggerFactory;
import org.drools.logger.KnowledgeRuntimeLoggerFactory;
import org.drools.runtime.StatefulKnowledgeSession;
import org.jbpm.process.instance.*;
import org.jbpm.process.workitem.wsht.WSHumanTaskHandler;
public class ProcessTest {
      public static final void main(String[] args) {
            KnowledgeBuilder kbuilder =
            KnowledgeBuilderFactory.newKnowledgeBuilder();
            kbuilder.add(ResourceFactory.newClassPathResource("general.bpmn")
            , ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource("inspeccion.bpm
            n"), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource("peticion.bpmn"
            ), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource("recepcion.bpmn
            "), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource("prensada.bpmn"
            ), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource("busqueda.bpmn"
            ), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource("descubre.bpmn"
            ), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource(
            "llenado.bpmn"), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource(
            "calidad.bpmn"), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource(
            "embotellado.bpmn"), ResourceType.BPMN2);
            kbuilder.add(ResourceFactory.newClassPathResource(
            "marketing.bpmn"), ResourceType.BPMN2);
```

```
                kbuilder.add(ResourceFactory.newClassPathResource(
                "distribucion.bpmn"), ResourceType.BPMN2);
                kbuilder.add(ResourceFactory.newClassPathResource("
                configurar.bpmn"), ResourceType.BPMN2);
                KnowledgeBase kbase = kbuilder.newKnowledgeBase();
                StatefulKnowledgeSession ksession =
                kbase.newStatefulKnowledgeSession();
                KnowledgeRuntimeLogger logger =
                KnowledgeRuntimeLoggerFactory.newConsoleLogger(ksession);
                KnowledgeRuntimeLoggerFactory.newConsoleLogger(ksession);
                ksession.getWorkItemManager().registerWorkItemHandler("Human
                Task", new WSHumanTaskHandler());

                //CALCULO DE LA FECHA DE EJECUCIÓN
                java.util.Date utilDate = new java.util.Date();
                java.sql.Date sqlDate = new
                java.sql.Date(utilDate.getTime());

                //VARIABLES LOCALES AL PROCESO
                HashMap<String, Object> params = new HashMap<String,
                Object>();
                params.put("paso", true);
                params.put("x", false);
                params.put("db", true);
                params.put("fecha", sqlDate);
                params.put("id_empleado", 0);
                params.put("id_calidad", 0);
                params.put("s_empleado", 'A');
                params.put("id_pedido", 0);
                params.put("id_cuba", 0);
                params.put("id_lote", 0);
                params.put("id_inspeccion", 0);

                //ELEMENTOS MODIFICABLES EN NUESTRO SISTEMA
                params.put("driver_db", "org.hsqldb.jdbcDriver");
                params.put("url_db", "jdbc:hsqldb:file: C:/Users /Jesus/
                Desktop/PROYECTO/jbpm-installer/workspace/Modelado Vino/db");
                params.put("cantidad_lote", 500f);
                params.put("capacidad_barrica", 250f);
                params.put("capacidad_cuba", 250f);
                params.put("capacidad_botella", 0.75f);
                params.put("t_env_joven", 1000f);
                params.put("t_env_crianza", 2000f);
                params.put("t_env_reserva", 3000f);
                params.put("t_rep_joven", 1000f);
                params.put("t_rep_crianza", 2000f);
                params.put("t_rep_reserva", 3000f);
                params.put("t_camion", 300f);
                ksession.startProcess("com.sample.general",params);
        }
}
```

El primer paso a la hora de implementar la ejecución del proceso vendrá descrito por la creación de una nueva instancia para KnowledgeBuilder. Este elemento se encargará de recoger todos los archivos bpmn2 suministrados mediante su método add y crear así un KnowledgePackage consumible por el repositorio de aplicaciones KnowledgeBase. Para lograr esta asociación entre KnowledgeBuilder y los distintos modelos definidos en bpmn se hace necesario el uso de elementos tipo Resource.

Una vez conseguida la asociación anterior se puede proceder a la creación de una instancia de KnowledgeBase. Este elemento representará el repositorio donde permanecerán recogidos los distintos elementos añadidos al KnowledgeBuilder, permitiendo además la creación de sesiones para la ejecución de los procesos. Como se puede observar, la implementación de una nueva sesión resulta bastante intuitiva ya que sólo será necesaria la invocación del método newStatefulKnowledgeSession.

La implementación del elemento StatefulKnowledgeSession permite la interacción iterativa con el engine de JBoss a través de una sesión definida para ello. Dentro de dicha sesión se conseguirá finalmente iniciar la ejecución del proceso a través del uso del método interno definido como startProcess. Entre los parámetros aceptados por este método se encuentran, por un lado, el identificador del proceso en ejecución para el establecimiento inequívoco del inicio de la secuencia y, por el otro, un objeto de tipo HashMap<String, Object> que fijará valores iniciales a las variables locales internas definidas en el proceso.

El último elemento a tener en cuenta en la implementación del archivo java definido anteriormente es el KnowledgeRuntimeLogger. Este elemento, usado a modo de asistente en la ejecución del proceso, irá creando secuencialmente un registro de incidencias a medida que el proceso completa las diferentes etapas establecidas. Tal y como cabe esperar, este registro no aportará por tanto ningún beneficio operacional al sistema, ya que su implementación sólo contempla utilidad al programador, reportando los posibles fallos asociados al mal funcionamiento de la ejecución.

5.2. Human task

Como elemento central de la interacción humana con el proceso, las tareas humanas se han convertido en un punto de interés trascendental en la realización de este proyecto. Por este motivo, y con objeto de mejorar su comprensión, resulta de vital importancia puntualizar con mayor profundidad sus principales propiedades o campos de definición. Entre estos campos se destacan los siguientes:

- ✓ ActorId: este campo hace referencia al participante del proyecto responsable de la ejecución de esta tarea. Para el caso concreto de que haya más de un actor responsable de la tarea se puede añadir más de un identificador, separando cada uno de ellos por comas.

- ✓ GroupId: esta potente propiedad permite asociar una tarea de ejecución a un grupo cerrado de usuarios. De este modo, cada uno de ellos competirá por la adjudicación de la tarea ya que sólo uno de ellos será el encargado de realizarla.
- ✓ Priority: para facilitar la organización del usuario, esta variedad de tareas permite la definición de prioridades específicas de ejecución. A partir de ellas se podrá dotar de mayor o menor importancia a una tarea según las características impuestas por el desarrollador.
- ✓ Skippable: este campo plantea la posibilidad de dejar al usuario la decisión de realizar o no la tarea programada.
- ✓ On entry and on exit actions: estos campos están asociados con la ejecución automática de scripts de código a la entrada y salida de la tarea humana.
- ✓ Parameter mapping: esta lista ordenada de variables permite obtener una relación directa entre los parámetros locales del proceso y los parámetros internos de la tarea. Toda esta asociación se realizará al inicio de la ejecución de la tarea, copiando la información de unos en los otros.
- ✓ Result mapping: esta asociación resulta análoga a la realizada por parameter mapping con la salvedad que, para este caso, la copia de datos se realiza a la inversa durante la finalización de la tarea.
- ✓ TaskName: hace referencia al nombre del formulario o plantilla definido en ftl para la interacción web con el usuario.
- ✓ Timer: este campo permite evitar posibles bloqueos durante la ejecución a través de la asignación de tiempos máximos para la realización de las tareas.

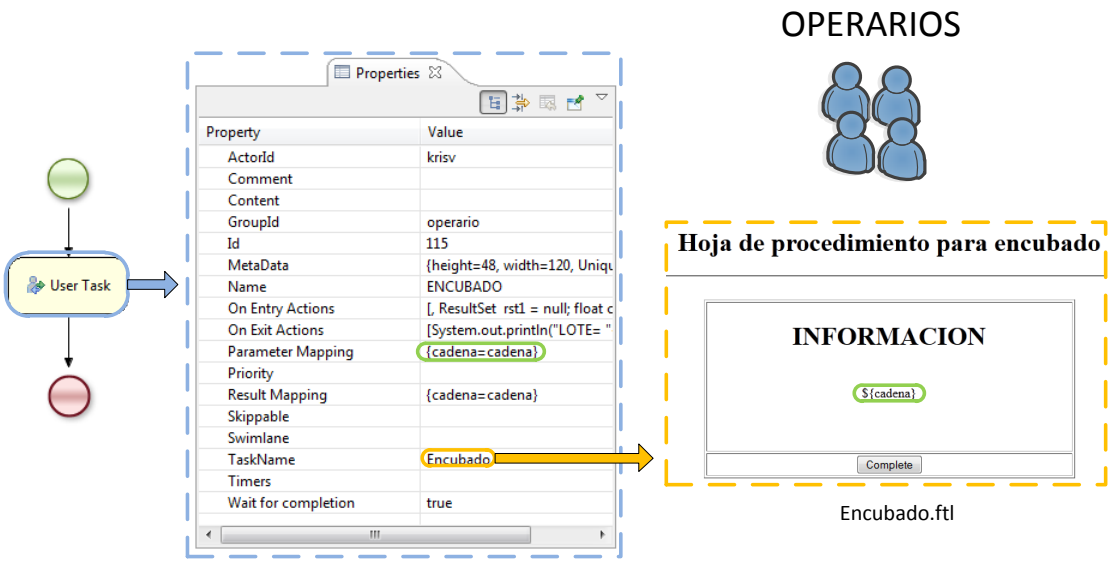


Figura 45: Asociación entre las tareas humanas y su interacción con el usuario

Uno de los aspectos más representativos configurables para estas tareas de usuario lo representa la asignación del actor o actores asociados al nodo. Esta asignación podrá ser realizada según dos puntos de vista completamente independientes. El primero de ellos vendrá determinado por la asignación directa de la tarea a un usuario o conjunto de éstos mediante la utilización del campo ActorId. A raíz del registro interno para los usuarios del servidor de aplicación se podrá implementar esta asignación directa de forma clara y unívoca. Para el caso particular de una asignación múltiple, es decir una tarea asignada a varios usuarios distintos, la tarea en cuestión quedará recogida dentro del cubo de tareas accesible a todos ellos, siendo necesario completar todas ellas para continuar con el proceso.

Respecto al segundo método de asignación, referenciado bajo el campo GroupId, el tratamiento de las tareas se plantea bien distinto. En este caso, la tarea es asignada a un grupo fijo de usuarios definidos según su rol específico en el sistema. Una vez lanzada la tarea, todos los usuarios tendrán visible la actividad a realizar con la salvedad de que para este caso los usuarios competirán por ella. Según esta estructura sólo un usuario de todos los que compartan el mismo rol podrá ser capaz de seleccionar y completar la tarea, implementando así una actuación bajo demanda.

Como aspecto singular asociado a la integración de servicios externos mediante esta tecnología, el manejador de tareas para la interacción humana deberá también ser integrado al motor interno de jBPM. Para realizar esta integración será necesario proceder al registro del manejador a través de la propia sesión definida en java. Para el caso particular implementado

en este proyecto, el manejador escogido es el elemento WSHumanTaskHandler y quedará registrado en la sesión mediante la sentencia:

```
ksession.getWorkItemManager().registerWorkItemHandler("Human Task", new
WSHumanTaskHandler());
```

Bajo esta elección, la comunicación entre el motor jBPM y el manejador de tareas seguirá una arquitectura básica de mensajes servidor-cliente. A través de ella, y utilizando la consola de usuario habilitada en el servidor, se realiza la gestión autónoma de las tareas para los distintos usuarios del sistema.

5.3. Interfaz de trabajo

La interfaz de trabajo establecida para los usuarios del proceso aparece definida por una agradable consola fácilmente accesible por cualquier navegador web. Tras alcanzar la ubicación de la consola, el primer paso para acceder a ella será completar la autenticación como usuario admitido en el proceso. Estos usuarios permanecerán definidos en ficheros locales de autentificación, donde quedarán registrados mediante combinación de nombre de usuario y contraseña. Para la configuración en red local, la ubicación fijada para alcanzar la consola de trabajo responde a la siguiente URL: localhost:8080/jbpm-console. La ventana del navegador para la autenticación del usuario se representa en la captura de pantalla expuesta a continuación.

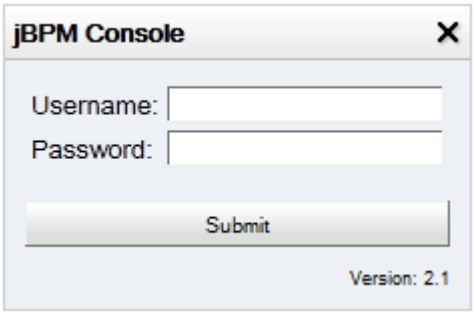


Figura 46: Ventana inicial para la autenticación del usuario

Una vez superado este proceso de autenticación, el usuario tendrá pleno acceso a la interfaz de gestión de tareas asociada a su cuenta. Dentro de esta interfaz existirán una serie de pestañas de elección así como diferentes sub-ventanas encargadas de mostrar la información interna relacionada con cada una de ellas. La interfaz comentada se recoge en la siguiente captura de pantalla.

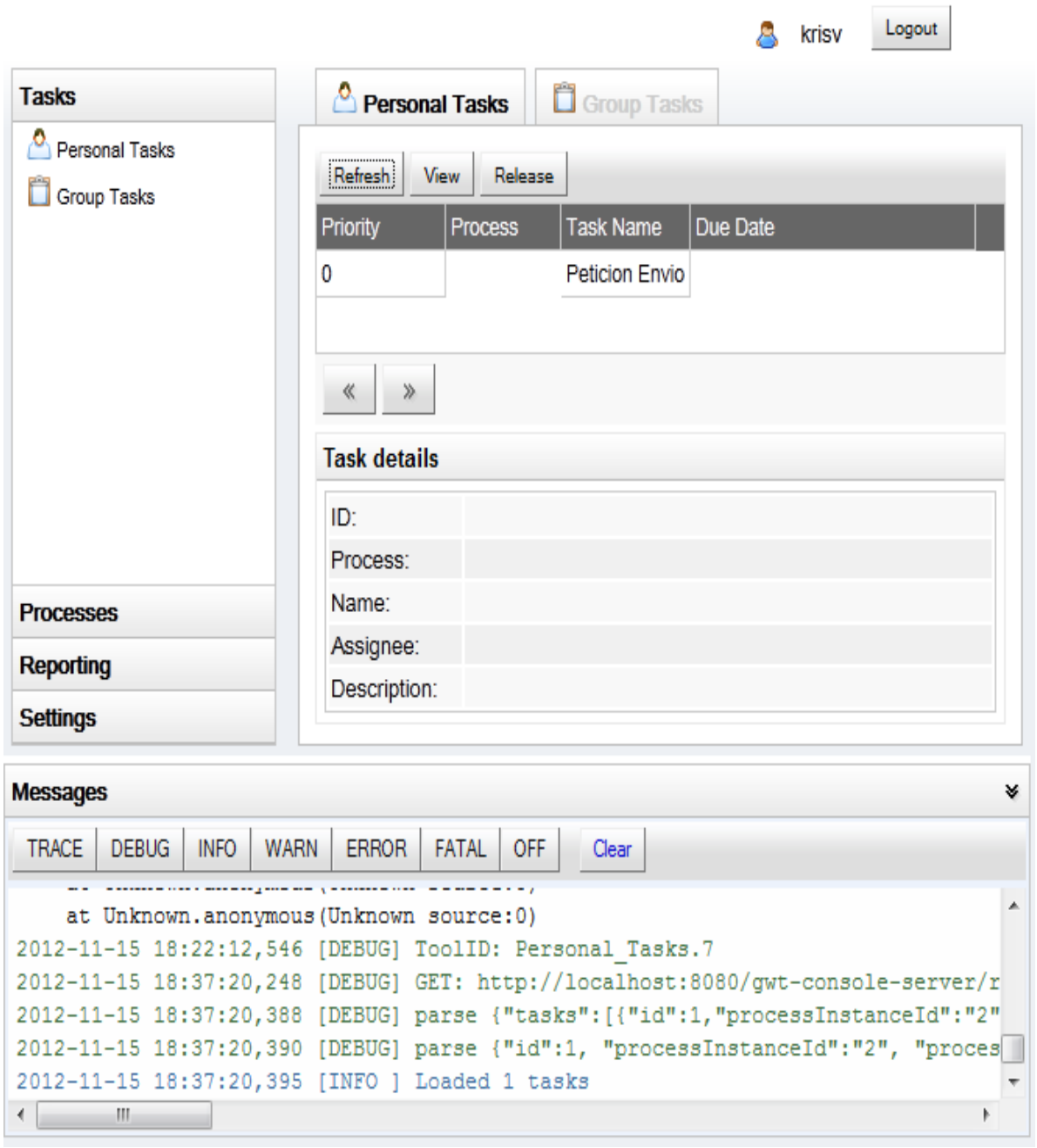


Figura 47: Interfaz de usuario para la gestión de tareas activas

En el índice de la izquierda se puede observar las diferentes ventanas accesibles por el usuario a través de las distintas pestañas definidas en la interfaz. La primera de ellas hace referencia a la ventana de gestión de tareas asignadas al usuario. Como se comentó anteriormente, la clasificación de estas tareas se encontrará segmentada según la naturaleza propia de la tarea, de forma que se podrán encontrar tareas tanto personales como de grupo. Para el caso concreto de las tareas de grupo, la adjudicación de la misma se realizará bajo demanda, siendo únicamente un usuario el encargado de completarla.

Otra de las pestañas incluidas en la interfaz es la que aparece bajo el sobrenombre de “procesos”. Esta nueva ventana, representada en la captura de pantalla de la figura 50, muestra un historial de ejecución de todos los procesos implementados en el servidor. En este caso no sólo mostrará los procesos que fueron ejecutados, sino también los que aún permanecen en ejecución así como algunas características interesantes asociadas a todos ellos. De forma adicional, en esta pestaña también se incluye la posibilidad de visualizar todos los procesos instalados en el sistema. A partir de ellos, y con una sencilla ventana de inicialización, se podrán iniciar desde cero nuevas instancias de los mismos, garantizando así su cómoda ejecución desde el inicio.

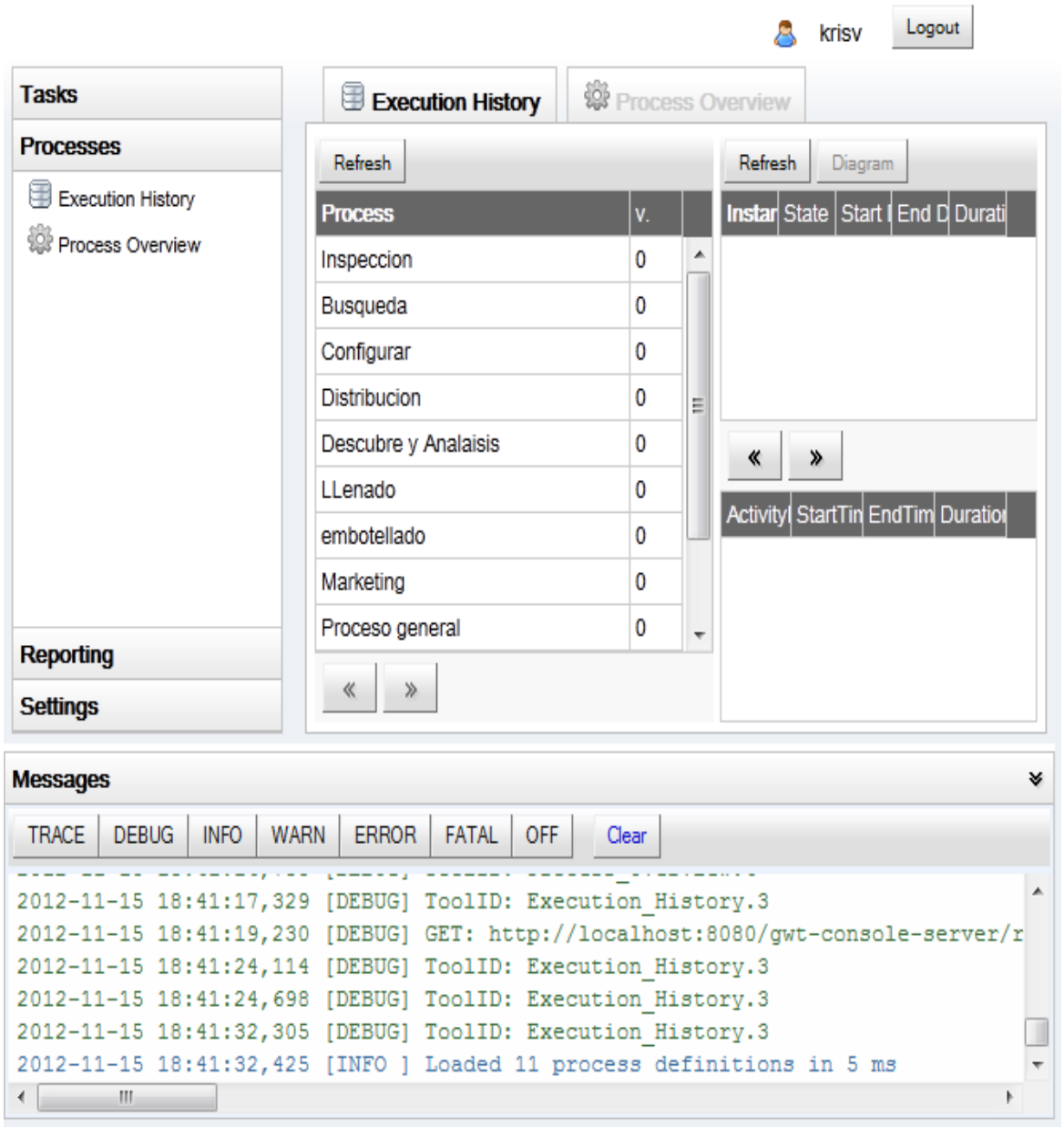


Figura 48: Ventana de gestión de procesos para el historial de ejecuciones

La última pestaña de esta interfaz está orientada principalmente a la gestión general del proceso de negocio por medio de un registro de las incidencias y ejecuciones asociadas al sistema. De este modo, y a través de una representación gráfica de monitorización de la actividad, se podrá realizar un seguimiento pseudo-controlado de las ejecuciones detallando además aspectos adicionales como su ubicación temporal o su duración.

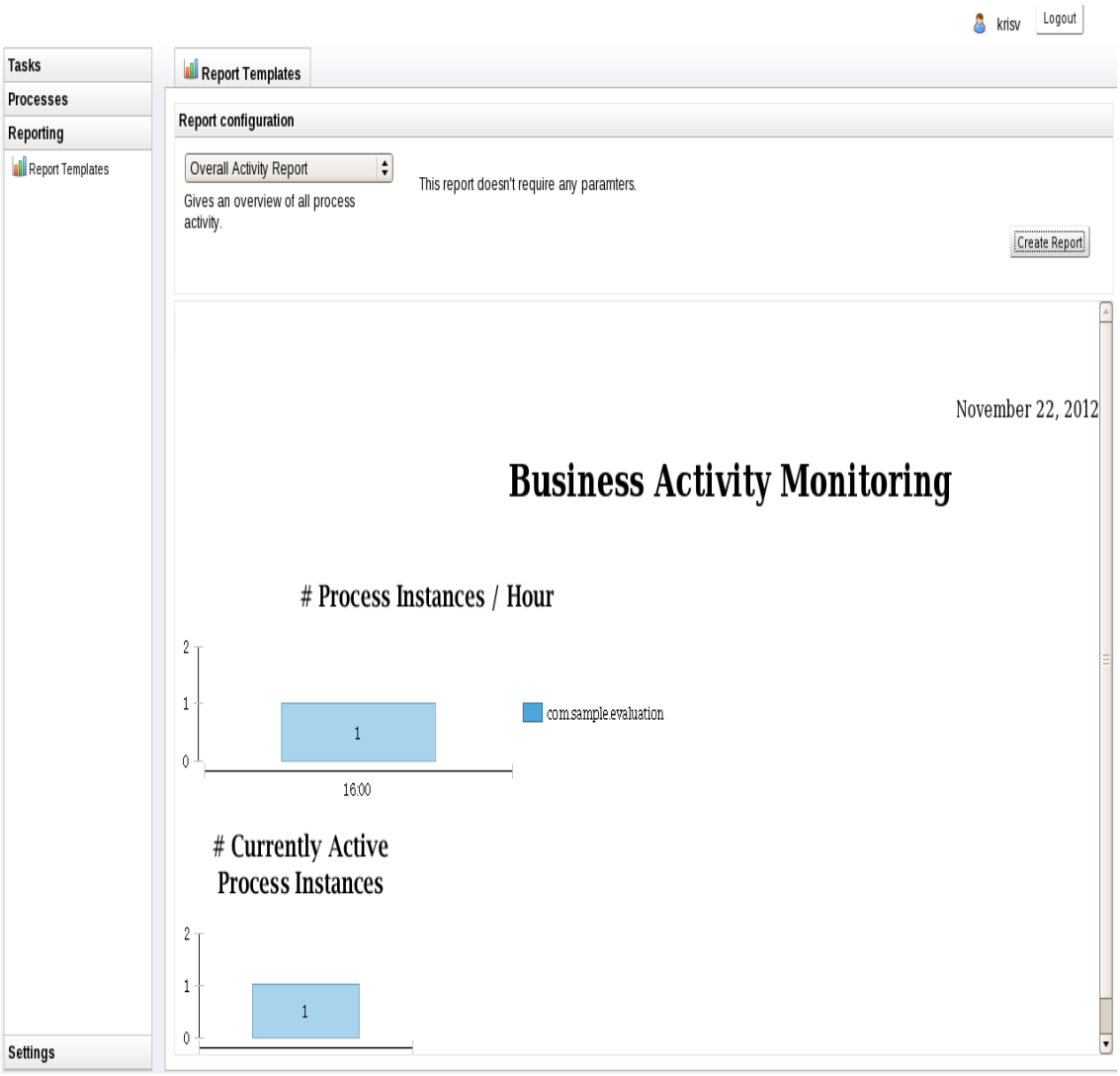


Figura 49: Interfaz de gestión con la monitorización de la actividad en el servidor

6. Conclusiones y líneas futuras

6.1. Conclusiones

Este proyecto es otro ejemplo más de la integración entre tecnologías de la información y de la comunicación y el desarrollo productivo industrial, en este caso la elaboración de vino. En comparación con los métodos tradicionales de gestión y organización se pueden observar algunas cualidades fruto de la utilización de esta tecnología. Ente estos atributos se destacan los siguientes:

- ✓ Permite la aplicación del estándar abierto Business Process Management Notation (BPMN) en la programación orientada a gráficos además de incluir el lenguaje de definición de procesos JPDL.
- ✓ Aporta la posibilidad de personalizar aplicaciones en cualquier entorno empresarial de forma escalable.
- ✓ Garantiza una comunicación sencilla e intuitiva mediante el uso de clases Java.
- ✓ Permite la agregación de servicios web externos como Hibernate, EJB, etc.
- ✓ Ofrece una capa de persistencia de datos independiente, pudiendo implementarla mediante distintas alternativas: MySQL, Postgres, Oracle, etc.
- ✓ jBPM está establemente ligado con el servidor de aplicaciones JBoss, uno de los más utilizados en la actualidad.
- ✓ Permite herramientas de definición, monitorización y ejecución de procesos usando un entorno web.
- ✓ El editor gráfico de procesos genera buena parte del código del programa de forma automática.
- ✓ El software es completamente de código abierto permitiendo una enorme capacidad de configuración, adaptación y personalización según las necesidades específicas de cualquier empresa.
- ✓ Posee una extensa comunidad de usuarios y desarrolladores con una gran trayectoria y experiencia.

Al margen de las cualidades descritas anteriormente, esta herramienta también plantea algunas deficiencias adicionales que conviene tener en cuenta. La primera de ellas se refiere a la interfaz de trabajo para los usuarios adscritos al proceso. Aunque esta interfaz permite una forma cómoda y organizada de identificar, asignar y realizar las distintas tareas

pendientes en el proceso, tanto las diferentes funcionalidades incluidas como el aspecto gráfico arrojan serias deficiencias de cara a posibles futuros proyectos comerciales. En cuanto a la gestión general del proceso, la información adjunta asociada a este fin resulta insuficiente, siendo en algún caso necesario incluso recurrir a mecanismos adicionales para lograr una gestión más acorde con las necesidades propias del sistema.

Respecto al desarrollo del proyecto, la utilización del entorno Eclipse ha resultado de gran ayuda durante todas las etapas del mismo. Su intuitiva forma de manejar procesos, tareas, archivos o variables, ha supuesto un importante apoyo en el cumplimiento de la integración final de esta solución. Como efecto negativo conviene destacar la falta, en ciertas ocasiones, de apoyo ya sea por la comunidad de usuarios de JBoss o por los distintos foros asociados repartidos por la red. A causa de estos motivos ha sido necesario, en algunos casos, sortear el uso de servicios adicionales, como el sistema de gestión de reglas definido como Drools, a fin de evitar extender demasiado la realización final de este proyecto. De forma adicional, también resulta interesante destacar la necesidad de incorporar herramientas para la edición y la personalización de los formularios web lanzados en algunas de sus tareas y actividades.

Otro elemento que conviene destacar ha sido la complejidad de adaptar la herramienta al proceso concreto implementado en este proyecto. Este aspecto, que requiere conocer en profundidad todos los métodos sobre los que se fundamenta la elaboración de vino, hace necesario el apoyo de expertos analistas, capaces de traducir los requisitos específicos del sistema a los desarrolladores finales de la herramienta BPM. Para este caso concreto, la traducción de las condiciones ha recaído sobre el mismo desarrollador resultando, en algunos casos, difícil la adaptación de ambos enfoques al planteamiento final de la solución.

Respecto la ejecución del proyecto a través del motor integrado en el servidor de aplicación, el resultado obtenido cubre convenientemente con todo lo esperado. Aunque tanto el seguimiento en secuencia de los procesos definidos como todas las interacciones implementadas en los mismos se desarrollan según lo deseado, también cabe destacar que existen algunos aspectos que permanecen en tela de juicio a la espera de ser solventados. El principal aspecto a considerar viene marcado por la incertidumbre alcanzada durante la ejecución en paralelo de múltiples hilos. Este efecto, que generalmente se traduce en el bloqueo de algunas tareas del proceso, se plantea como un problema asociado al manejador

de tareas humanas definido en el sistema. El planteamiento de la solución abarcará la edición de un nuevo manejador a través de la programación directa de sus elementos principales. Este ejercicio se planteará más adelante como una de las líneas interesantes para el desarrollo de futuros trabajos relacionados con esta herramienta.

6.2. Líneas futuras de trabajo

Para concluir con esta memoria se proponen algunas posibles líneas de continuación del proyecto que, de forma particular, considero que serían de especial interés para el desarrollo de futuras versiones del mismo.

- ✓ Con objeto de mejorar la interacción con los usuarios, una futura línea de trabajo podría plantear modernizar esta interfaz y ampliar así tanto sus capacidades como su estética.
- ✓ Como se comentó en el apartado 4.2 de la memoria, el desarrollo del proyecto ha estado sujeto a una serie de simplificaciones que pueden en algunos casos cuestionar la veracidad del proceso modelado frente al proceso real. Bajo esta premisa, una nueva línea de trabajo podría implicar el refinamiento del modelo simulado respecto al proceso real de negocio y limitar así las consideraciones impuestas durante el diseño.
- ✓ Otro de los aspectos interesantes de comentar hace referencia a los servicios externos integrables en el proceso. A partir de ellos se ofrece la posibilidad de integrar servicios adicionales como pueden ser, entre otros, el envío de correo electrónico o la transferencia de ficheros; todos ellos alcanzables desde la ejecución secuencial del proceso.
- ✓ Uno de los enfoques que se trató durante la implementación de este proyecto fue la posibilidad de incluir un modelo detallado de planificación de producción, todo ello ajustado para optimizar los beneficios propios de la bodega. Este planteamiento, aunque resulta muy complejo, podría introducir valiosas mejoras al proceso que potenciarían en gran medida la implantación de esta clase de soluciones en el mercado actual.
- ✓ La última de las líneas de continuación está asociada con la gestión general de las ejecuciones. Ésta, que se plantea limitada con jBPM, podría extenderse

sustancialmente mediante el uso de algún software adicional como puede ser MC4J. De esta forma se permitirá un control más completo y exhaustivo de la gestión global del proceso de negocio.

7. Bibliografía

[1] A. Dávila, D. Cecilia Análisis, “Diseño e Implementación de una aplicación Workflow para el seguimiento de procesos de los servicios que ofrece el Área de Comercialización de la Empresa Eléctrica Regional Centro Sur.” Universidad Politécnica Salesiana 2011

[2] J. Coitiño, J. Goyoaga “Diseñar y desarrollar un sistema de workflow para la ANC” Universidad de la República.

[3] Adriana D. Urdaneta, Alex A. Pérez “Sistemas de Información, Workflow” Universidad Central de Venezuela 2008.

[5] J. Casal Ruiz “Desarrollo de un módulo para la gestión gráfica de workflows para procesos software” 2009

[4] Bizagi BPMN 2.0, Bizagi Process Modeler. Disponible en Web: www.bizagi.com

[5] Stephen A. White, Derek Miers “Guía de referencia y modelado BPMN” Future Strategies Inc. 2009

[6] C. Fernandez Llatas “Representación, Interpretación y Aprendizaje de Flujos de Trabajo basado en Actividades para la estandarización de Vías Clínicas” Universidad politécnica de Valencia 2008.

[7] Centro oficial del BPM para España y Latinoamérica. Disponible en Web: http://www.club-bpm.com

[8] A. Cruz Mangas “Análisis de herramientas de simulación de procesos de negocio” Universidad de Sevilla 2007.

[9] “QPR ProcessGuide, BPMN Modeling Guide” QPR Software Plc 2007

[10] C. Mendoza Giraldo “Plug-in de Eclipse para la creación de procesos de negocio” Universidad de Sevilla 2006

[11] S. McGowan, I. Springer “JBoss AS Administration Console User Guide” 2010

[12] JBoss Enterprise Middleware. Disponible en Web: http://www.redhat.com

[13] Curso servidor de aplicaciones JBoss. Disponible en Web: http://globalmentoring.com.mx

[14] C. Colin, C. Dan, S. Koussouris, L. Texier, “The JBoss 4 Application Server Guide.2007. Disponible en Web: http://docs.jboss.org/jbossas/docs

[15] JBoss Community “JBoss AS”. Disponible en Web: https://community.jboss.org

[16] M. Salehie S. Li, “Architectural Recovery of JBoss Application Server” Software Architecture course 2004.

[17] M. Azoff , “Technology Infrastructure. Aplication Server JBoss” Butler Group 2004.

[18] D. Andreadis, “Introduction to Application Server 5.0” Conferencia JBoss World de RedHat 2008

[19] I. García, “La API JMX y Monitorizacion de JBoss” 2008. Disponible en Web: http://www.adictosaltrabajo.com/tutoriales/pdfs/monitorizarJBossJMX.pdf

[20] D. La Sage, S. Dorfield “JBoss Enterprise SOA Platform 5” RedHat. Disponible en Web: access.redhat.com/knowledge/docs

[21] “JBoss Enterprise SOA Platform” RedHat. Disponible en Web: http://www.latam.redhat.com

[22] D, Hernandez, “Explorar Bases de datos HSQLDB” 2008. Disponible en Web: http://www.adictosaltrabajo.com/tutoriales/pdfs/hsqldbBrowser.pdf

[23] L. Costero, S. Oliva, M. Sánchez, “Sistemas informáticos” Universidad Complutense de Madrid 2006

[24] C. Márquez, “JBoss jBPM, Solution Architect” RedHat, JBoss by RedHat 2010 Disponible en Web: www.thinkhemisphere.com

[25] P. Bazán, R. Giadini, F. Díaz, “Tecnologías para implementar un marco integrador de SOA y BPM” Facultad de informática La plata. 2010.

[26] J. Koening “JBoss jBPM White Paper” Universidad de California 2004

[27] T. Surdilovic “Web-based BPM with jBPM 5” JBoss Users ¬ Developers Conference (Boston) 2012

[] “JBoss Frameworks jBPM” RedHat, Alfresco. Disponible en Web: www.vdel.com

[28] K.. Aers “JBoss jBPM Overview” JBoss, a division of RedHat. Disponible en wiki.eclipse.org

[29] “Introduccion a jBPM 5” PlugTree, JBud.com, Universidad de Mendoza 2011

[30] M. Salatino, “JBoss nnn6 Drools” Disponible en Web: salaboy.com

[31] G. Robles, “La elaboracion de los vinos” Disponible en Web: www.asturiasadaptada.org

[32] R. Rodríguez, “Análisis de los Peligros y Puntos de Control Crítico (APPCC)” Mejora de bodega de ValdeFuentes (Cáceres)

[33] “El vino en la botella” Enología. Disponible en Web: www.asturiasadaptada.org

[34] C. Falcó, “Entender de Vino” Editorial mr 1999

[35] “Vinicultura” Vinos y bodegas la sangre de Ronda, Saber Vino. Disponible en Web: http://www.bodegaslasangrederonda.es

[36] R. Rodríguez, “Ingeniería de los procesos” Mejora de bodega de Valdefuentes (Cáceres)

[37] J. Llera, N. Martinengo, “Diagramas de flujo para el diseño de un sistema de control de calidad: Proceso de elaboración de vino blanco” Universidad nacional de Cuyo (Argentina) 2004.

[38] Norma IRAM 14104:2001 “Guía para la aplicación de buenas prácticas de manufactura” Bodegas, Instituto argentino de normalización y certificación, Instituto Nacional de Vitivinicultura, 2001

[39] “Aplicación del sistema de análisis de riesgos y control de puntos críticos en vinos” Federación Española del Vino, Ministerio de Sanidad y Consumo 1998

7. Anexos

7.1. Anexo I: Manual de instalación y configuración

En este apartado de la memoria se incluirán los principales pasos a seguir para obtener la instalación y configuración del entorno de desarrollo utilizado en este proyecto. De forma adicional también se introducirá la línea principal de configuración alcanzada tanto para el software Eclipse como para el propio servidor de aplicación JBoss.

El primer aspecto que se deberá barajar a la hora de querer instalar el entorno de desarrollo y ejecución contemplado en este proyecto vendrá determinado por la capacidad de cumplir con los requisitos mínimos de instalación. Tras comprobar satisfactoriamente este hecho se procederá a la instalación de los tres apartados fundamentales de este manual:

- ✓ La instalación de Java.
- ✓ La instalación del motor JBoss jBPM y el servidor de aplicación JBoss.
- ✓ La instalación del entorno de diseño de procesos, en este caso Eclipse.

La primera pieza de la instalación será Java, más concretamente la versión denominada “Java 2 Software Development Kit”. Para alcanzar esta instalación bastará con acceder al link de descarga de Java y proceder con los pasos establecidos en el ejecutable. Lo más aconsejable será acceder a la versión más actualizada del mismo, todas ellas encontradas en el link de Oracle: http://www.oracle.com/technetwork/java.

Tras descargar e instalar la versión JDK de la página, el siguiente paso consistirá en actualizar las variables de entorno del propio sistema operativo. Para realizar este proceso bastará con acceder al menú de configuración del sistema y alcanzar desde allí la pestaña para la edición de las variables propias del entorno. La pantalla a la que se debe acceder aparece representada en la captura de pantalla mostrada a continuación.

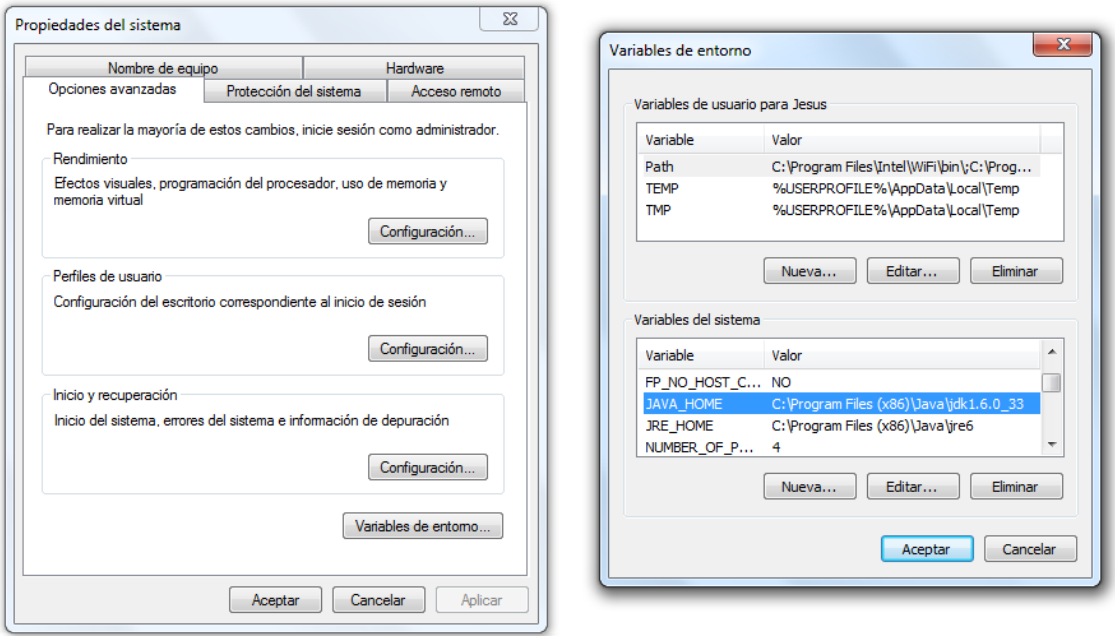


Figura 50: Captura de pantalla para la configuración de las variables de entorno del sistema operativo

Una vez se ha accedido a esta ventana se deberá incluir la ruta correspondiente al fichero \bin instalado para el JDK de Java dentro de la variable JAVA_HOME y PATH. Tras completar este proceso se considerará correctamente instalado el paquete Java en el equipo de trabajo, procediendo así a la instalación de los siguientes módulos comentados.

Para llevar a cabo los siguientes pasos de la instalación será necesario incluir en el equipo una pequeña utilidad de Java denominada Ant. Esta utilidad implementa tareas repetitivas y mecánicas por medio de sentencias de ejecución automática como las que se verán más adelante. Para incluir Ant en el sistema basta con descargar el fichero de instalación de la página de Apache a través de la URL http://ant.apache.org/bindownload.cgi, e instalar su contenido en memoria. Tras completar la instalación será necesario también adecuar las variables del entorno del sistema operativo a la nueva configuración, introduciendo para ello la ruta del fichero \bin de Ant en las variables de estado ANT_HOME y PATH.

En el siguiente paso de la instalación se procederá a la descarga del módulo de JBPM deseado. Para realizar este proceso se accederá a la página de la comunidad oficial de JBoss y se buscará la distribución de JBPM que más se adapte a las especificaciones del usuario. Esta página, accesible desde el enlace http://sourceforge.net/projects/jbpm/files/, ofrece una forma fácil y actualizada de acceder a los paquetes de instalación más utilizados en la comunidad BPM. Para el caso concreto de este proyecto, la versión instalada es la 5.1.0.

Tras la descarga y descompresión del paquete jBPM se procederá con la instalación. Por comodidad, este módulo contiene un método de instalación implementado mediante un archivo denominado build.xml. A partir de este archivo se implementará la secuencia de acciones necesarias para la descarga, descompresión e instalación de los distintos elementos incluidos en este módulo. Una vez finalizado el proceso, el resultado final obtenido será la instalación completa del servidor de aplicación JBoss, varias bases de datos entre las que se encuentran la HSQL y el entorno de desarrollo Eclipse Indigo con los módulos necesarios para la definición de procesos en BPMN. Tras instalar todo ello, el acceso a los distintos recursos instalados también se encuentra accesible dentro del archivo build.xml. A partir de él por tanto se podrá tener acceso a todos estos elementos, ya sea de forma individual o colectiva. Para el segundo caso, la sentencia global de ejecución vendrá descrita por "ant start.demo", comenzando así con la inicialización de todos los elementos instalados.

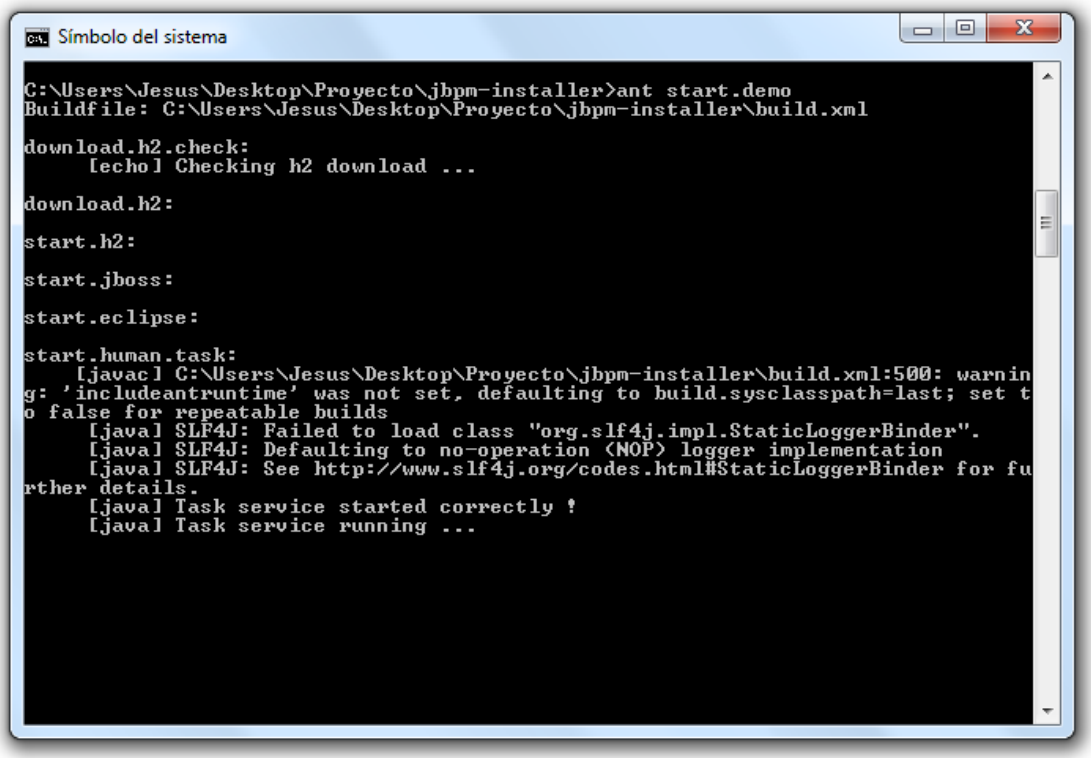


Figura 51: Captura de pantalla de la inicialización de los elementos instalados en el módulo jBPM

7.2. Anexo II: Modelado en BPMN

El conjunto de funcionalidades ofrecidas por BPMN, permite una gran variedad de elementos para la creación e implementación de procesos independientemente de su ámbito de aplicación. Existe además una extensa cantidad de recursos bibliográficos donde los usuarios más noveles de BPMN podrán sumergirse de forma sencilla e intuitiva en la creación de cualquier tipo de proceso de negocio.

De forma genérica, cualquier proceso definido en BPMN estará compuesto por un conjunto de recursos y actividades relacionadas de forma secuencial. Para la definición de modelos bajo este estándar, se hace imprescindible tener un sólido conocimiento sobre los distintos elementos incluidos en él. Por este motivo, en esta sección se detallará un breve resumen de los objetos gráficos implementados por BPMN, así como una resumida definición de sus funciones principales.

ELEMENTOS BÁSICOS DE MODELADO BPMN

ELEMENTO	DEFINICION	NOMBRE BPMN
Elementos de flujo (Flow objects)	Principales elementos gráficos que definen el comportamiento del proceso.	Eventos Actividades Decisión
Conectores (connecting objects)	Permiten la conexión de los objetos de flujo para crear la estructura del proceso.	Transición Flujo de mensaje Asociación
Canales (Swimlane)	Mecanismos de organización de las actividades en categorías virtuales.	Área funcional Fase
Artefactos (Artifacts)	Proveen información adicional sobre el proceso otorgando flexibilidad a la notación.	Objeto de datos Grupo Anotación

ELEMENTOS DE FLUJO

Eventos: elemento que sucede durante el curso del proceso afectando al flujo del mismo. Normalmente tienen una causa o resultado

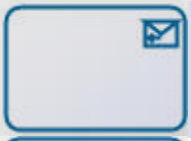
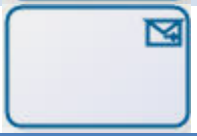
TIPO EVENTO	NOMBRE BPMN	DEFINICION	NOTACION

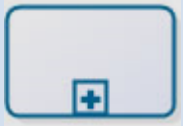
Eventos de Inicio	Start	Representa el punto de inicio del proceso	
	Message start	El proceso activo envía un mensaje a otro proceso para activar su inicio	
	Time Start	Permite configurar la hora de inicio del proceso	
	Signal start	El proceso activo envía una señal que causa el inicio de otro proceso	

TIPO EVENTO	NOMBRE BPMN	DEFINICION	NOTACION
Eventos intermedios	Intermediate	Afecta al proceso pero ni lo inicia ni finaliza	
	Temporizador	Mecanismo de retraso dentro del proceso	
	Compensación	Indica la necesidad de compensación en el proceso	
	Error	Evento que interrumpe la actividad capturando el error intermedio	
	Mensaje	Recepción y envío de mensajes	
	Enlace	Mecanismo de conexión entre dos secciones de un proceso	
	Señal	Recepción y envío de comunicaciones dentro y a través de los niveles del proceso	

TIPO EVENTO	NOMBRE BPMN	DEFINICION	NOTACION
Eventos de Fin	End	Indica la terminación del proceso	
	Terminador	Fin del proceso	
	Cancelación	Indica que el proceso debe ser cancelado pudiendo lanzar el evento intermedio de cancelación	
	Error	Permite capturar errores terminando con todos los threads activos	
	Mensaje	Terminación que incluye el envío de un mensaje a otro proceso	
	Señal	Terminación que incluye el envío de una señal a otro proceso	

Actividades: representan las tareas, automáticas o manuales realizadas por los miembros involucrados en el proyecto ya sea un usuario o un sistema externo al sistema

NOMBRE BPMN	DEFINICION	NOTACION
Tarea de usuario	Elemento donde un humano realiza una tarea en cierta cantidad de tiempo	
Tarea de servicio	Tarea que usa algún tipo de servicio Web o una aplicación automática	
Tarea de recibir	Tarea simple para que llegue un mensaje. Se completa con la recepción del mensaje	
Tarea de enviar	Tarea simple dedicada al envío de un mensaje a un proceso específico	

Script	Tarea automática ejecutada por el servidor sin interacción humana	
Manual	Tarea sin motor de ejecución en el proceso de negocio o alguna aplicación	
Subproceso	Actividad compuesta incluida dentro de un proceso	
Subproceso Múltiple	Subproceso que permite la creación de instancias múltiples. Cada instancia representa una relación dentro del proceso	
Subproceso Transaccional	Permite la implementación de escenarios de negocio con transacciones de duraciones prolongadas	
Subproceso Embebido	Contiene actividades independientes del proceso pariente compartiendo información y datos	

Decisiones: elementos que permiten controlar la divergencia y convergencia en el flujo del proceso. Estas decisiones permiten ramificaciones, combinaciones y fusiones aportando funcionalidades en la creación de procesos más exigentes.

NOMBRE BPMN	DEFINICION	NOTACION
Decisión exclusiva	Decisión basada en los datos del sistema	
Decisión basada en evento	La decisión se basa en los eventos ocurridos	
Decisión Inclusiva	Permite la elección de uno o varios caminos dependiendo de las actividades anteriores	
Decisión compleja	Ofrece un control más complejo del flujo del proceso	
Decisión paralela	Indica distintos puntos en el proceso donde varias ramas se desprenden o convergen en paralelo	

7.3. Anexo III: Primera aproximación al modelado (Bizagi)

Para el modelado gráfico del proceso de elaboración de vino descrito en el apartado anterior se ha optado por dos soluciones bien distintas. La primera de ellas, tal y como se comentó en el capítulo correspondiente a BPMN, se desarrolla a partir de un software propietario para el modelado de procesos de negocio. A través de este software, denominado "Bizagi Process Modeler", se estableció una primera aproximación al modelado general del proceso. Por medio de esta definición se concluyó la correspondiente secuencia de tareas que debían aparecer, desde la recogida inicial de la uva en la vendimia hasta la distribución final del producto elaborado, todo ello interpretado desde el punto de vista de los distintos participantes del proceso.

Este primer modelado, aunque precario, plasma correctamente la complejidad del proceso global, mostrando además la carga relacional e interactiva de todos los elementos implicados en él. Aunque el desarrollo del modelado siguió una segunda versión bastante más avanzada y completa, resulta interesante remarcar esta aproximación a fin de comprender el desarrollo final de la solución planteada en este proyecto.

Al ser Bizagi un software orientado al modelado gráfico, la definición planteada resulta bastante intuitiva, sobre todo atendiendo a la descripción detallada del proceso real que se ha suministrado en el apartado anterior. Por este motivo y en busca de no alargar demasiado esta sección previa al modelado final, se representa a continuación el modelado en BPMN del proceso completo para la elaboración de vino.

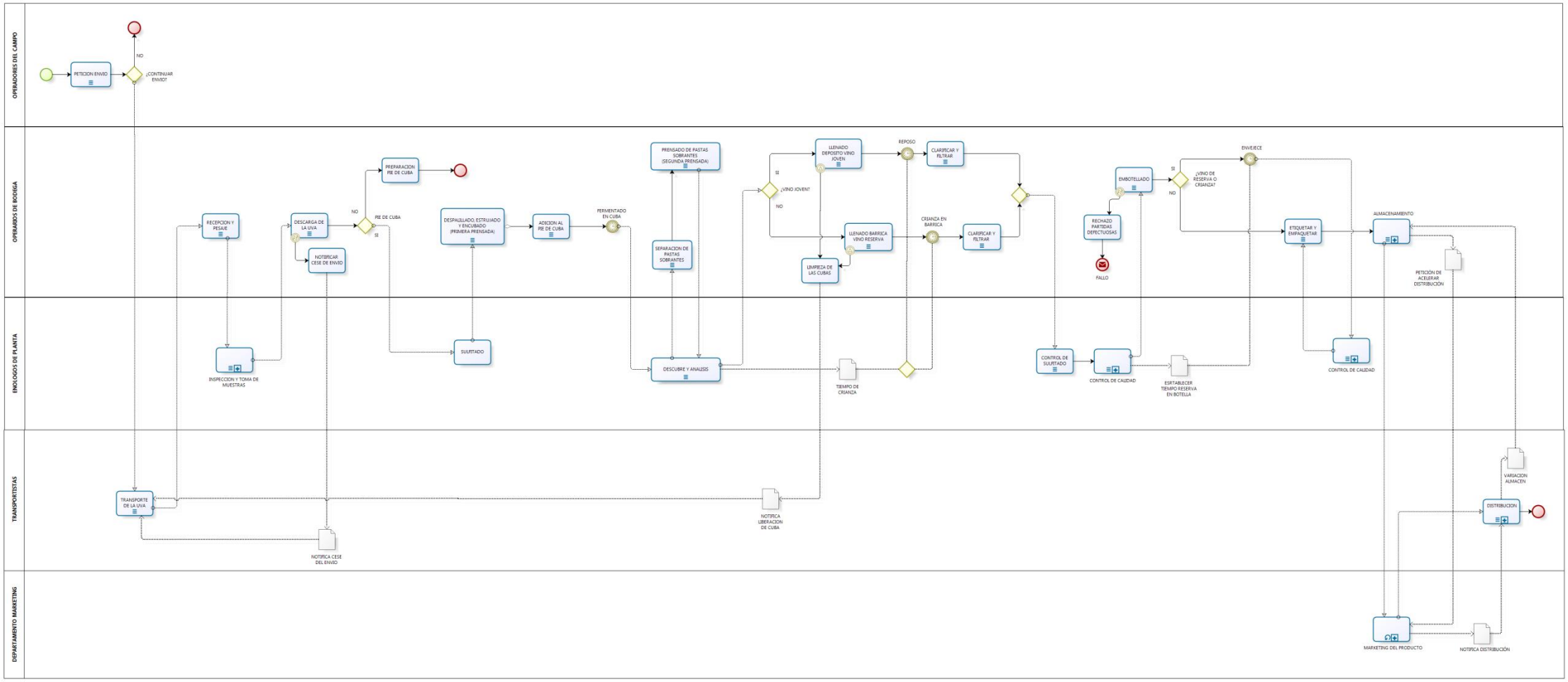


Figura 52: Proceso general aplicado a el esquema genérico de elaboración de vino.

Para el modelo representado en la figura anterior se ha tenido a bien incluir el diseño modular de alguna de sus secciones. Como se observó en el capítulo correspondiente al modelado en BPMN, existe la posibilidad de agrupar las definiciones por medio de nodos específicos para subprocesos. Estos elementos permitirán reducir la complejidad general por medio de estas sub-definiciones. En el caso concreto del proceso general definido anteriormente, los subprocesos se corresponden con los siguientes:

- *Inspección y toma de muestras*: este subproceso hace referencia a la etapa de análisis previo a la recepción de la materia prima en la bodega. Durante esta etapa se realizarán los controles específicos para establecer que la partida entrante cumple con los requisitos establecidos por la bodega. Tal y como se comentó en la sección anterior, estos requisitos han sido agrupados en tres grupos: los integrados dentro del control visual del producto, del control higiénico del transporte y del programa de limpieza. Todo ello permanece recogido en la siguiente figura correspondiente a la definición interna de este subproceso.

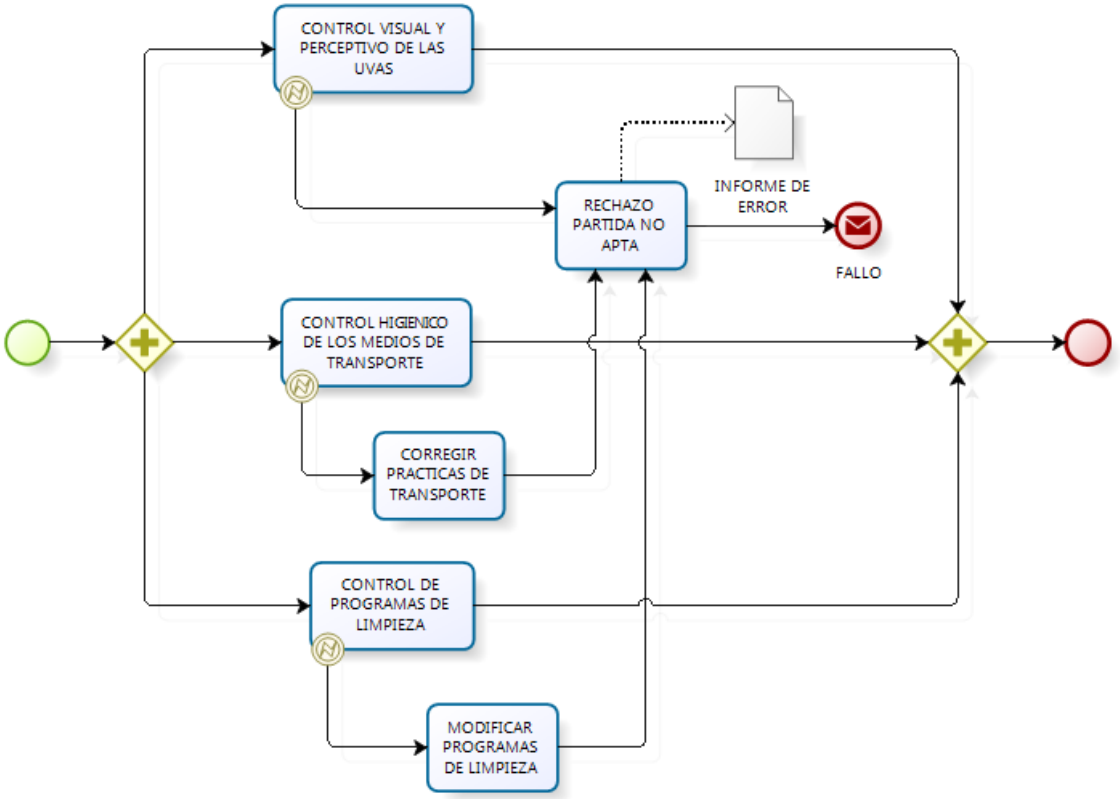


Figura 53: Subproceso correspondiente a la inspección y toma de muestras para la materia prima entrante.

- *Control de calidad*: este subproceso se corresponde con los estudios de control de calidad llevados a cabo tras los periodos de crianza en barrica y envejecimiento en botella. Durante esta etapa se realizarán ensayos físicos y químicos al producto a fin de evaluar las características intrínsecas del vino. Todo este proceso queda recogido en la figura presentada a continuación.

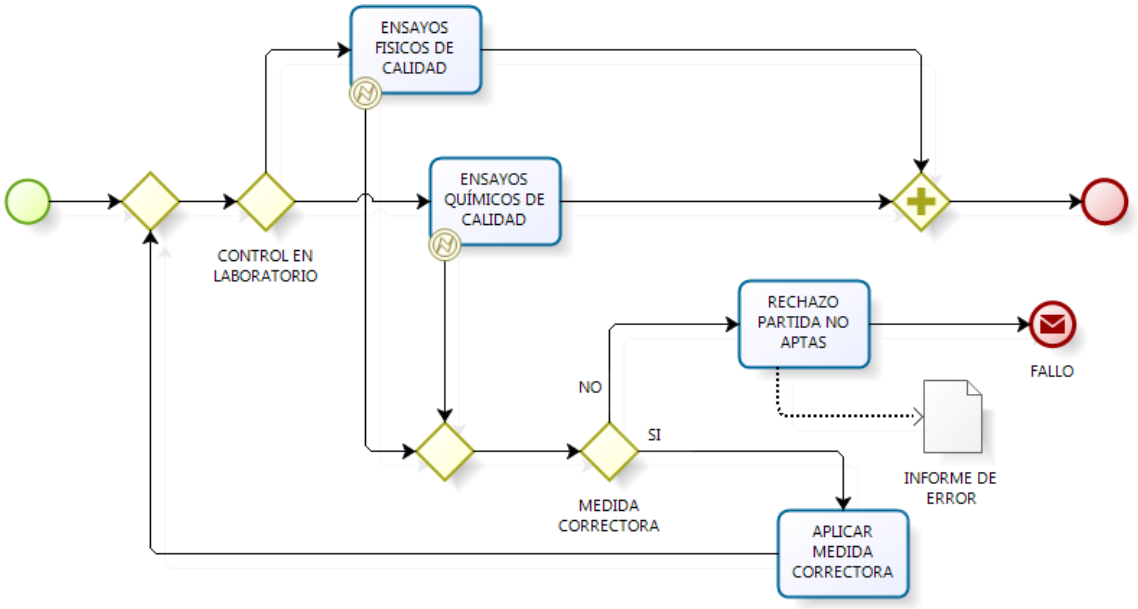


Figura 54: Subproceso correspondiente al control de calidad.

- *Almacenamiento*: este flujo de trabajo hace referencia al proceso de almacenamiento para el producto terminado. Para llevar a cabo esta tarea será necesario tener un control exhaustivo de las condiciones específicas de los espacios dedicados a este almacenamiento. Por otro lado, también se contemplará la posibilidad de reubicar o corregir posibles problemas relacionados con el almacenamiento del producto.

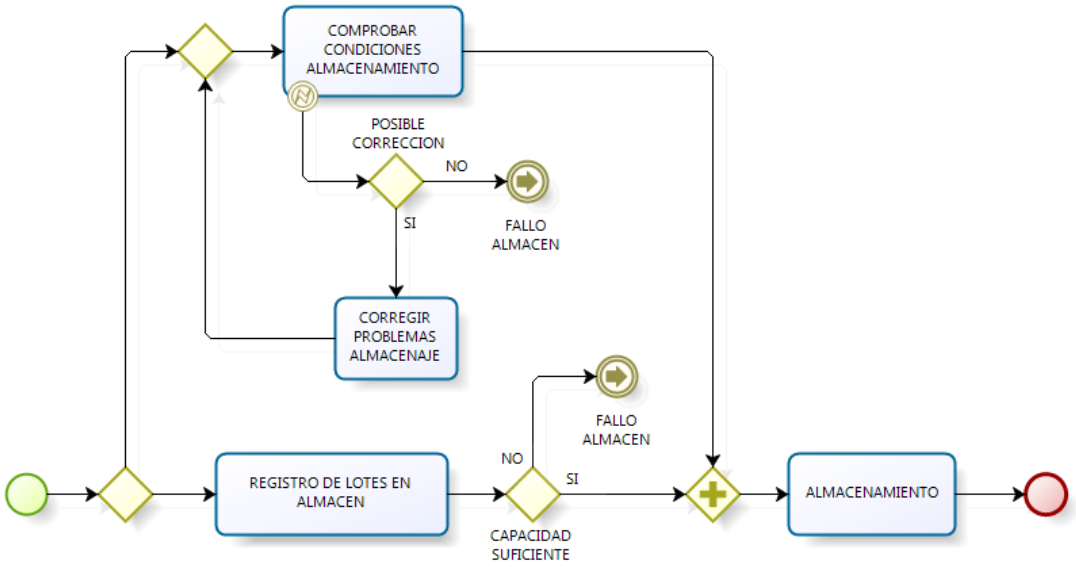


Figura 55: Subproceso correspondiente al almacenamiento del producto terminado.

- *Marketing del producto*: esta etapa define, de forma aproximada, cómo sería la labor realizada por el departamento de marketing y venta de la bodega. Este proceso contempla la realización de promociones sobre los productos elaborados como consecuencia de la acumulación de producto terminado en los almacenes. De forma adicional también se genera la correspondiente variación del stock en los almacenes, así como la hoja de distribución necesaria para hacer llegar el producto a los clientes a través de los transportistas.

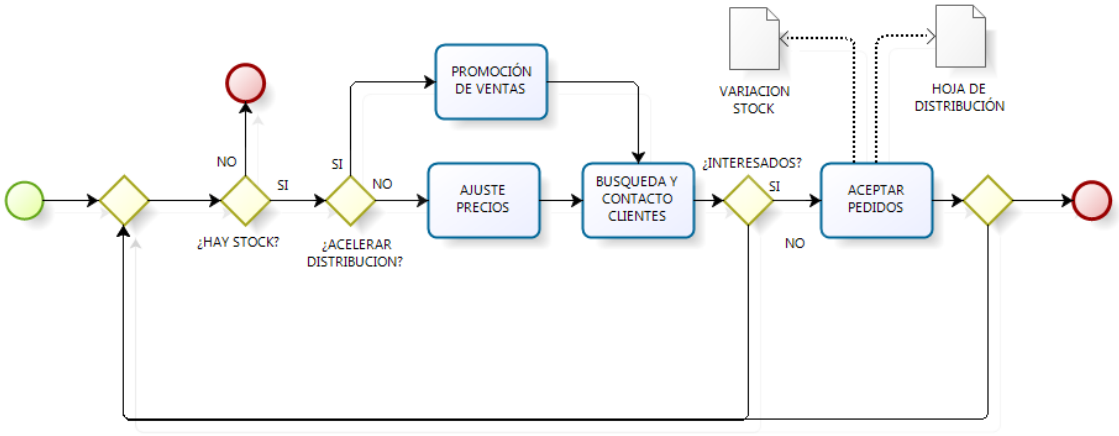


Figura 56: Subproceso correspondiente al marketing del producto.

Como se ha establecido anteriormente en este documento, el modelo anterior no representa la solución final adoptada por este proyecto. Sin embargo, y tal como se puede observar, este modelo establece una aproximación interesante y con bastante grado de exactitud acerca de cómo interpretar un sistema con estas características.

Figura 57: Capturas de pantalla relativa a los controles de visibilidad y notificaciones, modelado, análisis en ejecución, administración de tareas, estudio del rendimiento, etc.

La línea a seguir por el proyecto se cambió radicalmente a causa de las distintas deficiencias e incompatibilidades encontradas en el software propietario de Bizagi para la implementación y la ejecución de los procesos de negocio. Este software, denominado BPM Suite, aunque aportaba una solución potente y completa para la realización final del proyecto desarrollado, planteaba serios obstáculos que acabaron, finalmente, propiciando el uso de Eclipse como componente principal para el desarrollo del proyecto.