# 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.

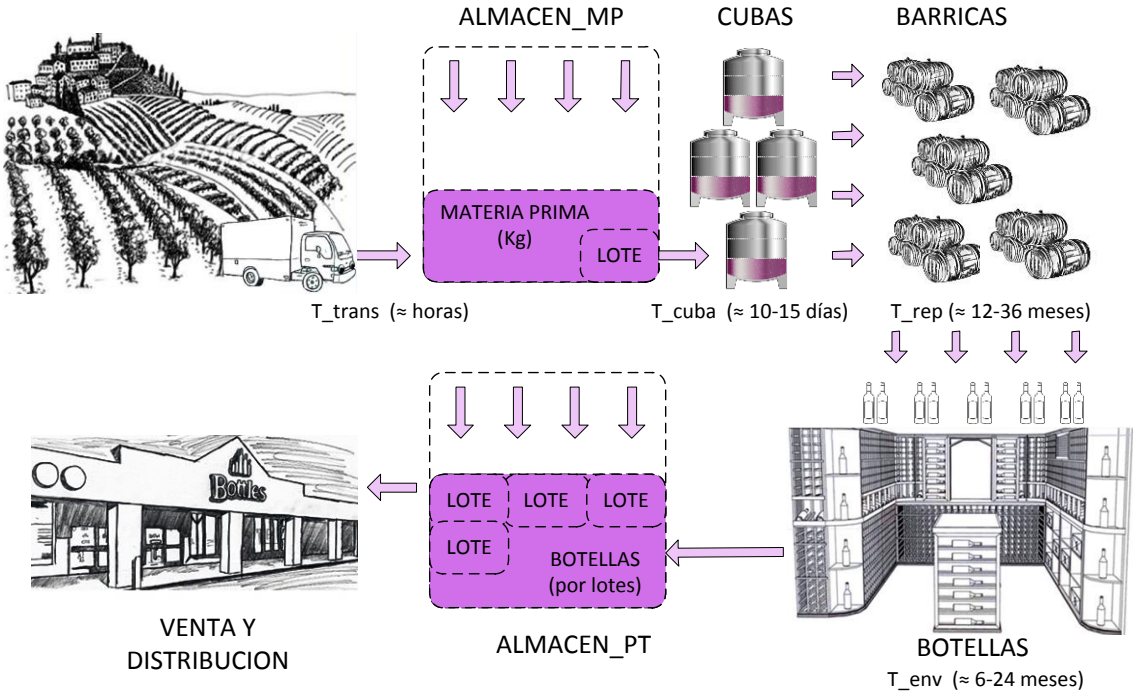



**Figura 18: Esquema conceptual de elaboración de vino**

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.

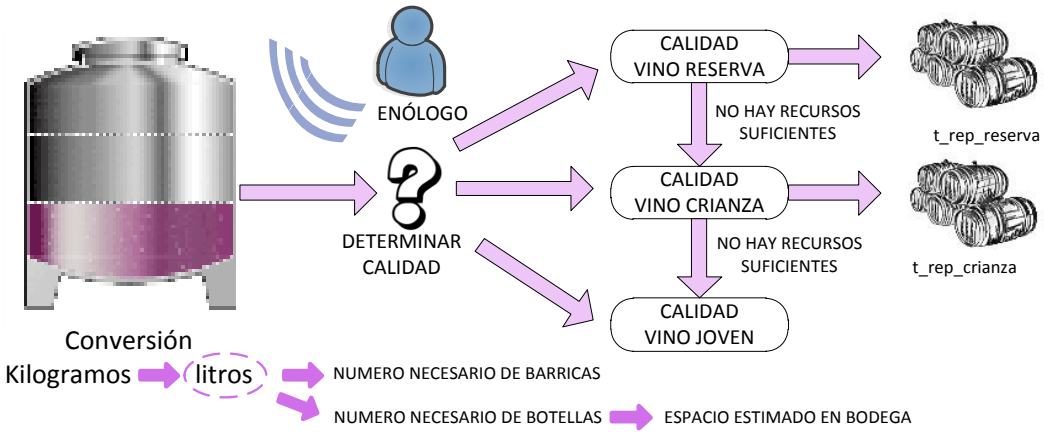



**Figura 19: Representación conceptual de la toma de decisión durante el descubre y análisis del vino**

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.

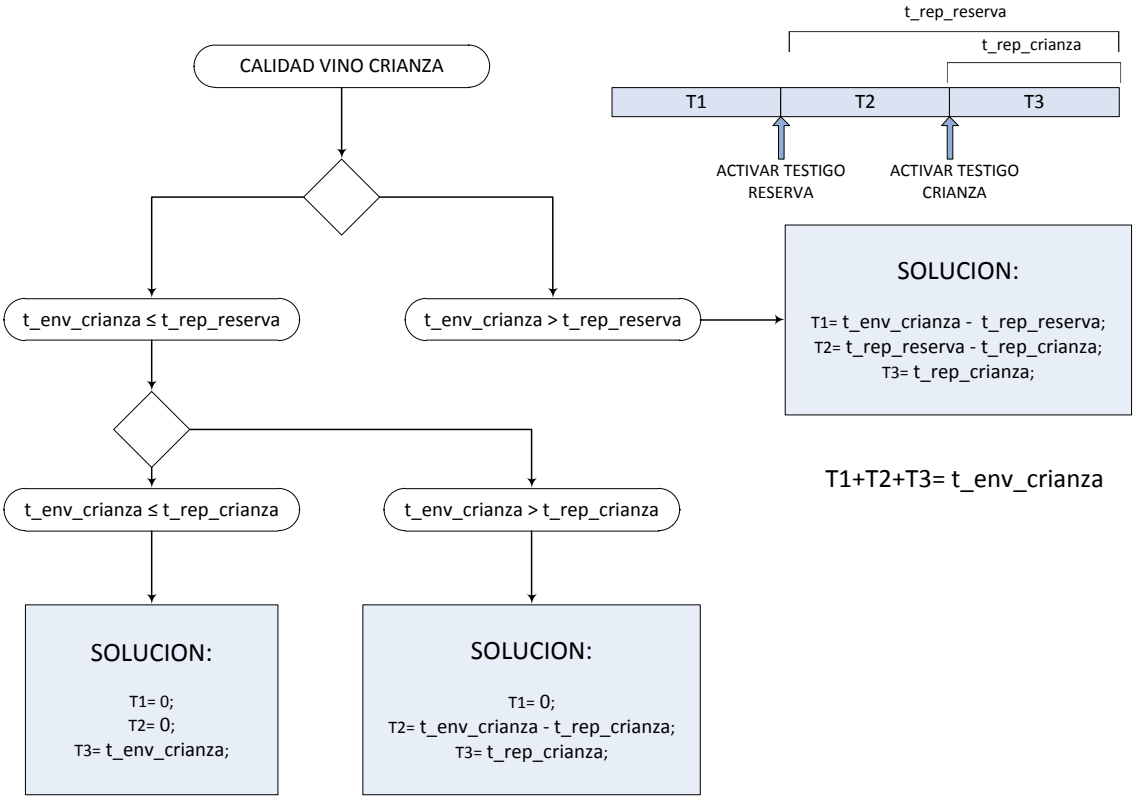



**Figura 20: Diagrama de activación de testigos para vino crianza**

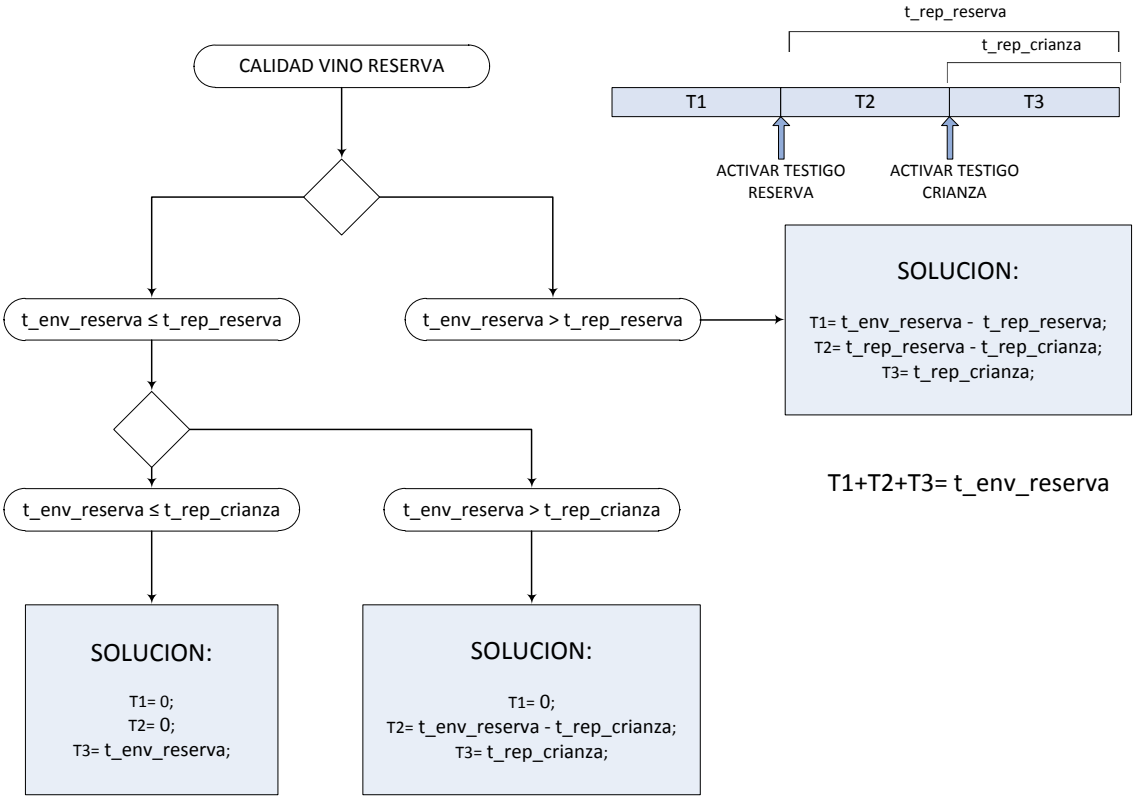



**Figura 21: Diagrama de activación de testigos para vino reserva**

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.

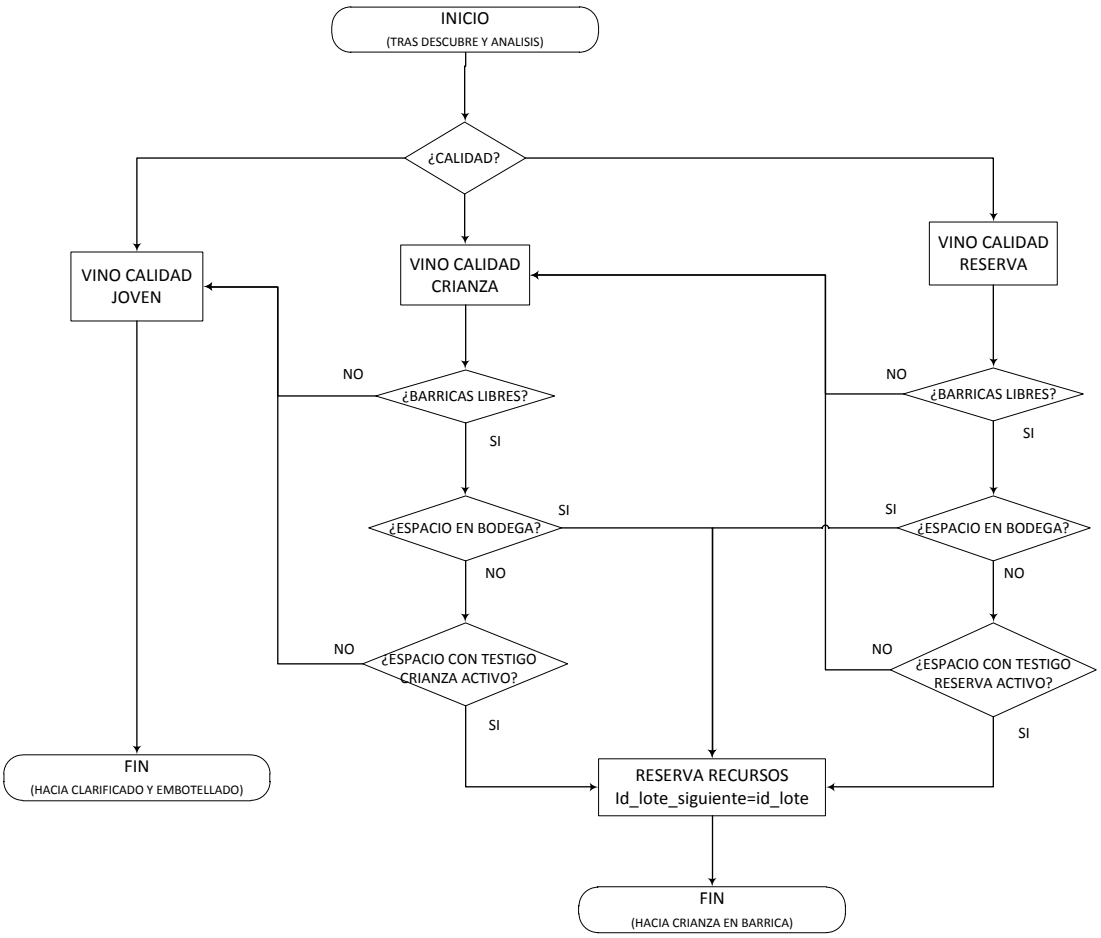



**Figura 22: Diagrama de flujo para la toma de decisiones sobre la calidad del vino**

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

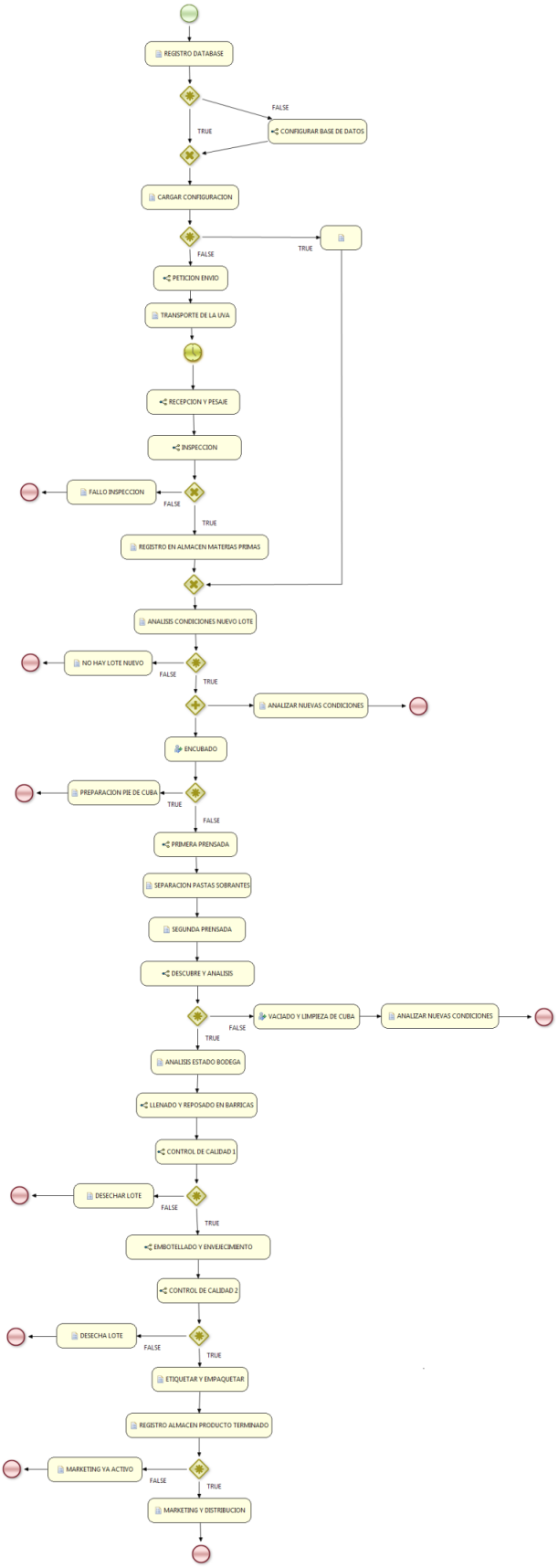



**Figura 23: Proceso general completo para el modelado de la elaboración genérica de vino.**

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.

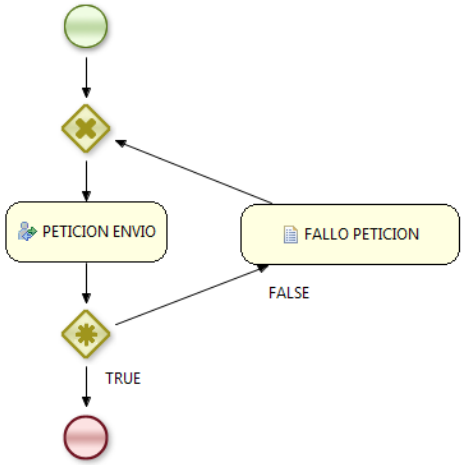



**Figura 24: Subproceso correspondiente a la petición de envío de materias primas.**

- *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.

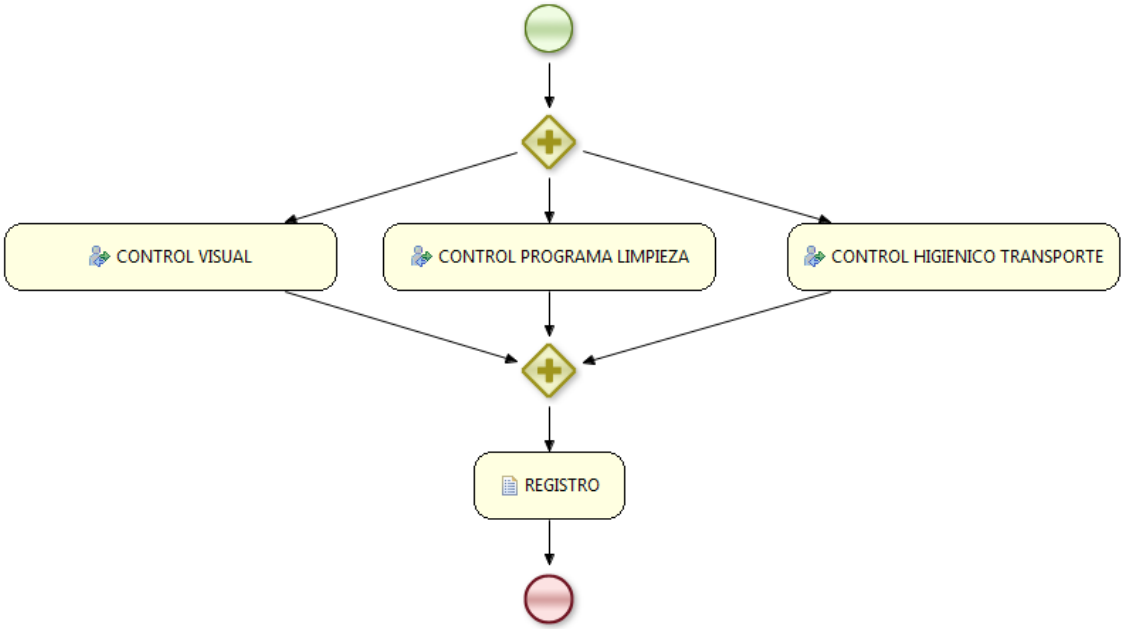



**Figura 25: Subproceso de inspección para la materia prima entrante en la bodega.**

- *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.

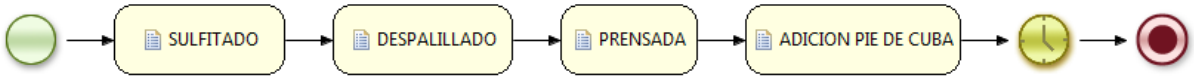



**Figura 26: Subproceso de prensado y reposo de vino.**

- 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.

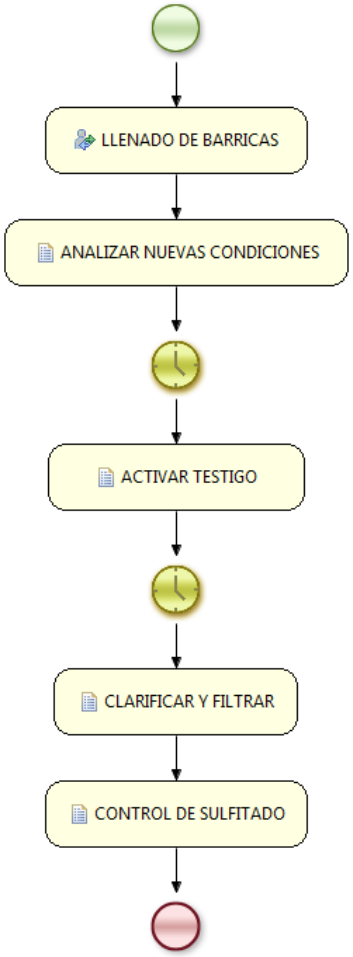



**Figura 27: Subproceso para el llenado y reposo en barricas.**

- *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.

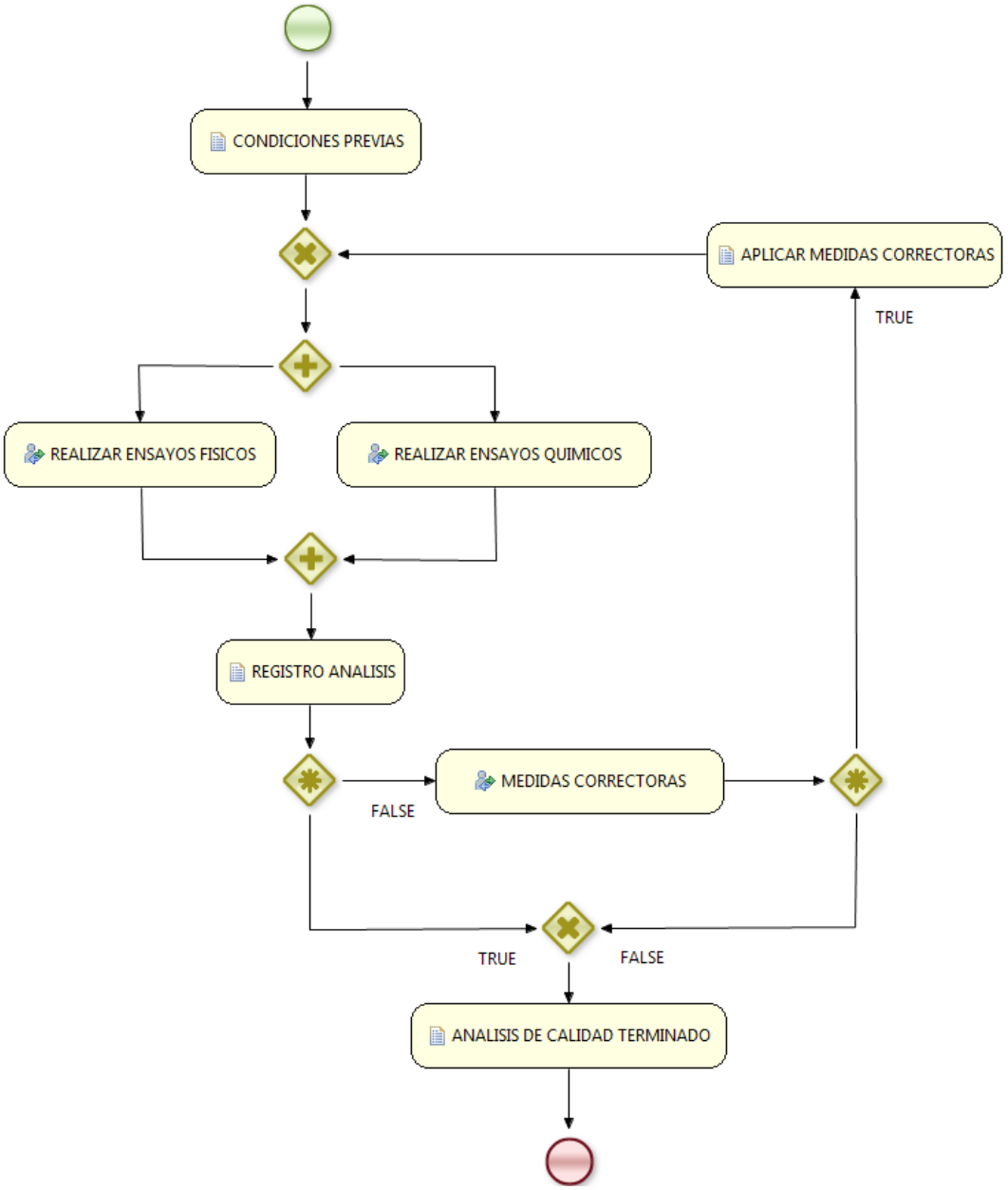



**Figura 28: Subproceso dedicado al control de calidad del vino.**

- *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.

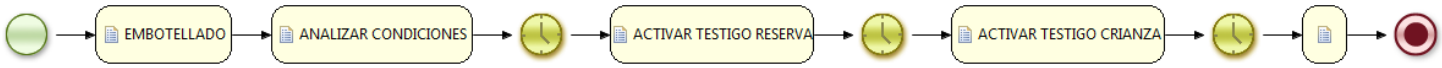



**Figura 29: Subproceso correspondiente al embotellado y envejecimiento del vino en botellas.**

- *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.

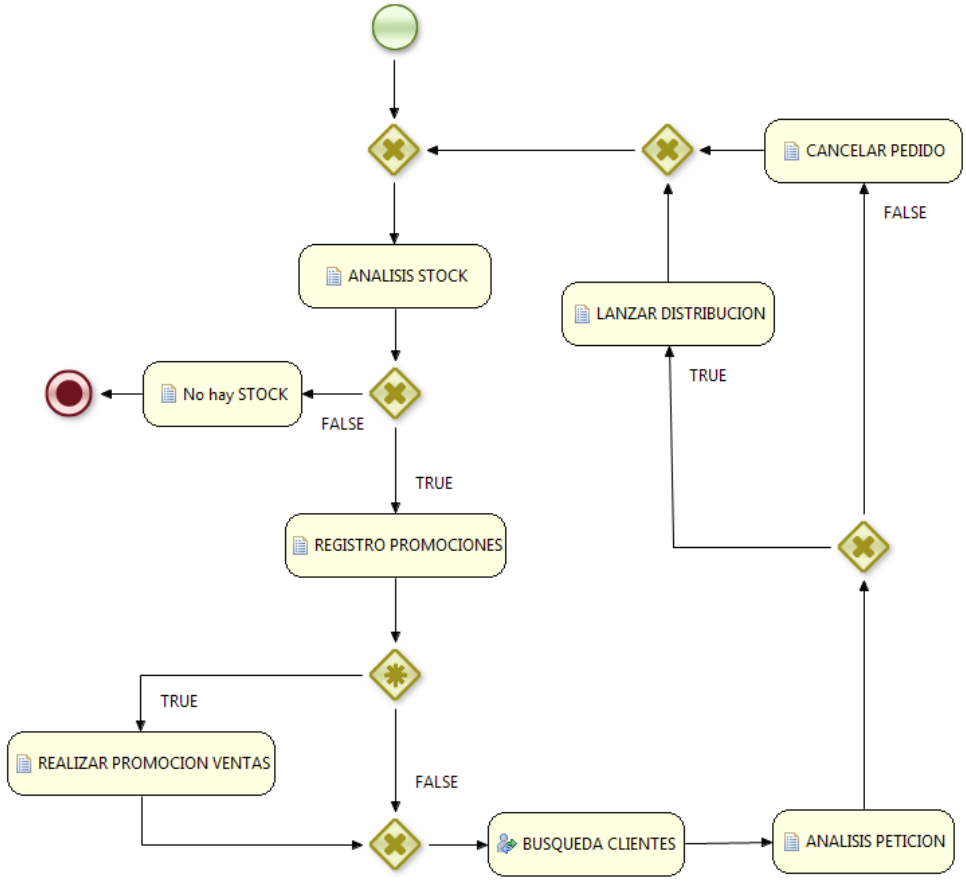



**Figura 30: Subproceso dedicado al marketing y la distribución del producto elaborado.**

## 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*.

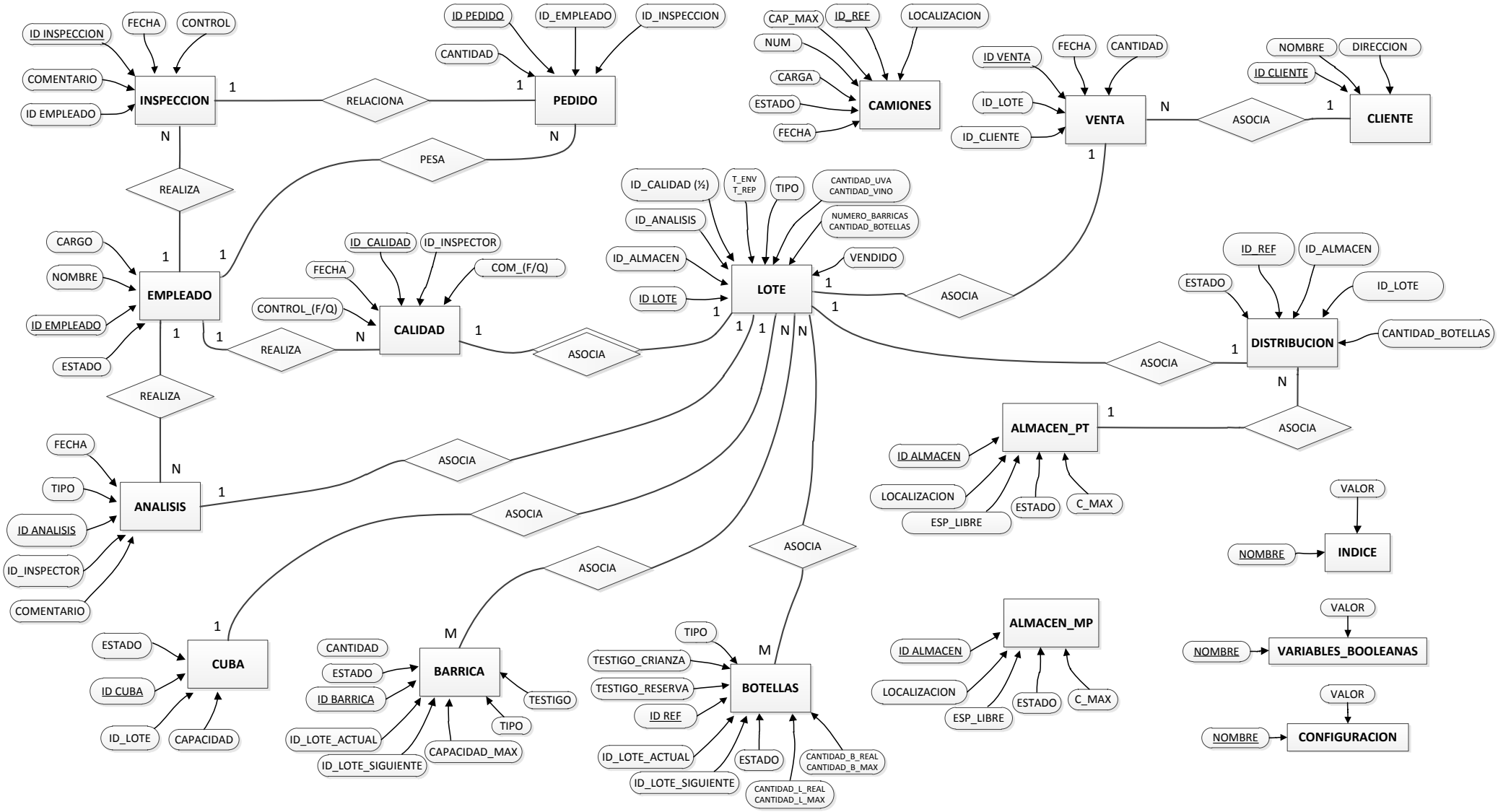



**Figura 31: Representación del modelo de entidad relación de la base de datos del sistema**

## 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:

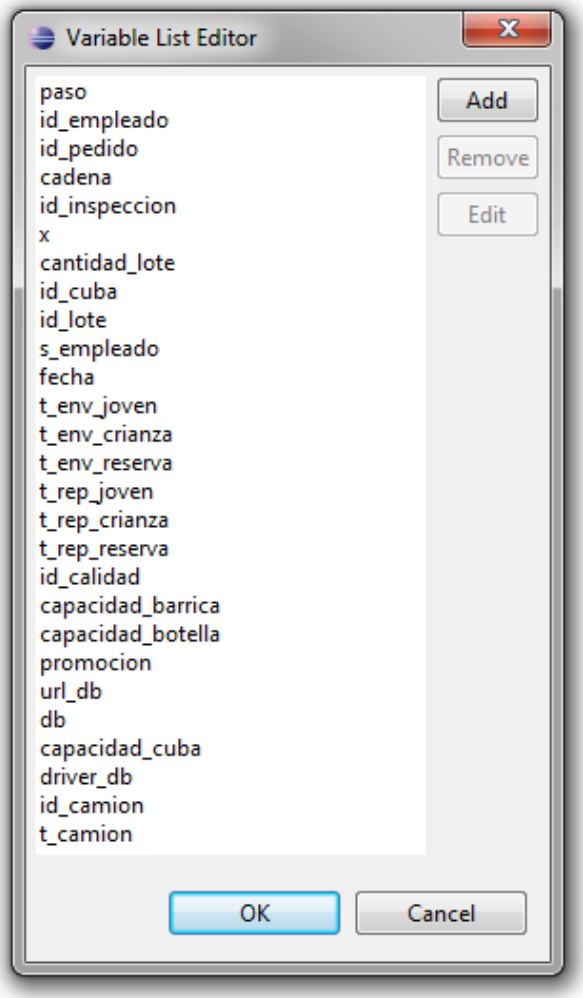



**Figura 32: Captura de pantalla sobre el editor de variables de Eclipse**

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:

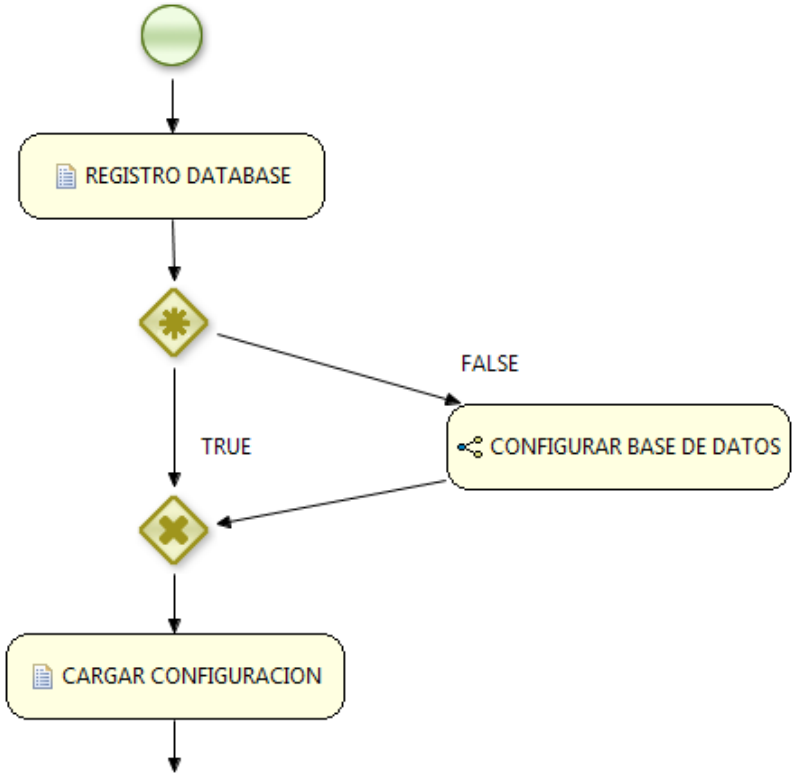



**Figura 33: Etapa inicial del proceso para la configuración general del sistema**

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.

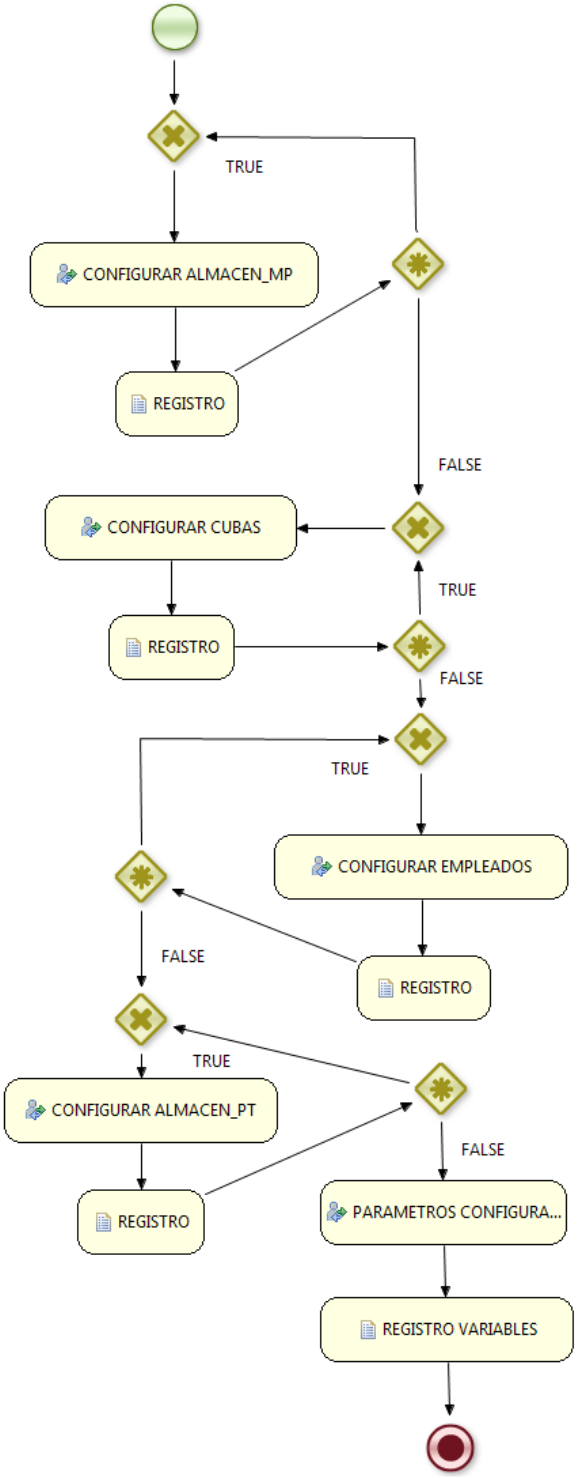



**Figura 34: Subproceso para la configuración de las condiciones iniciales de la bodega.**

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.

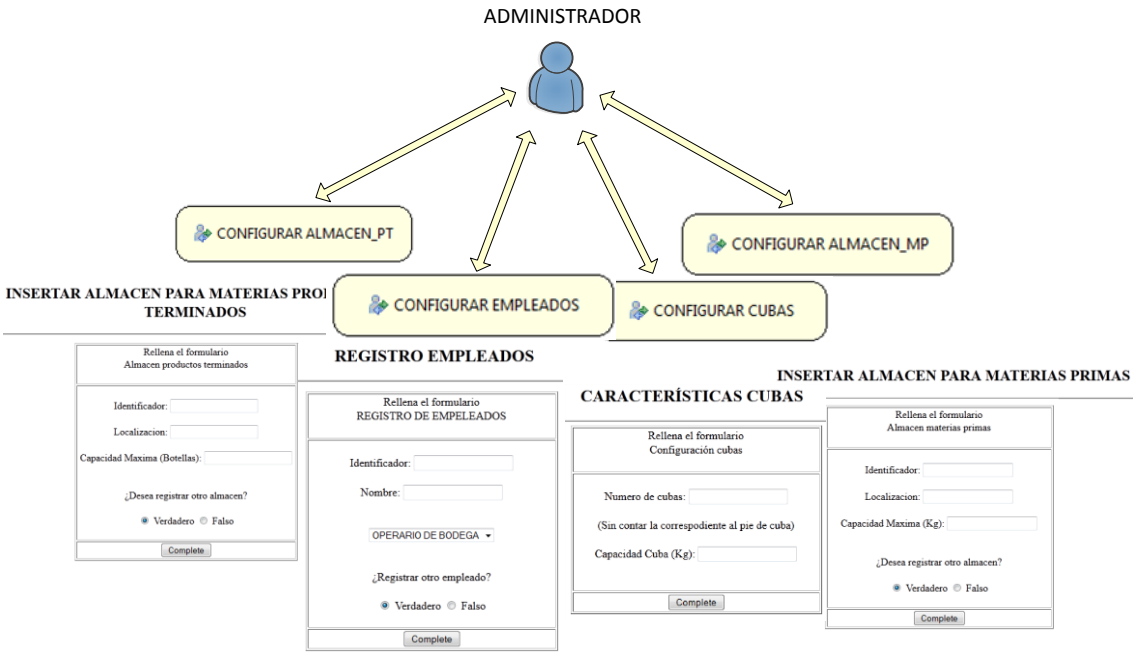
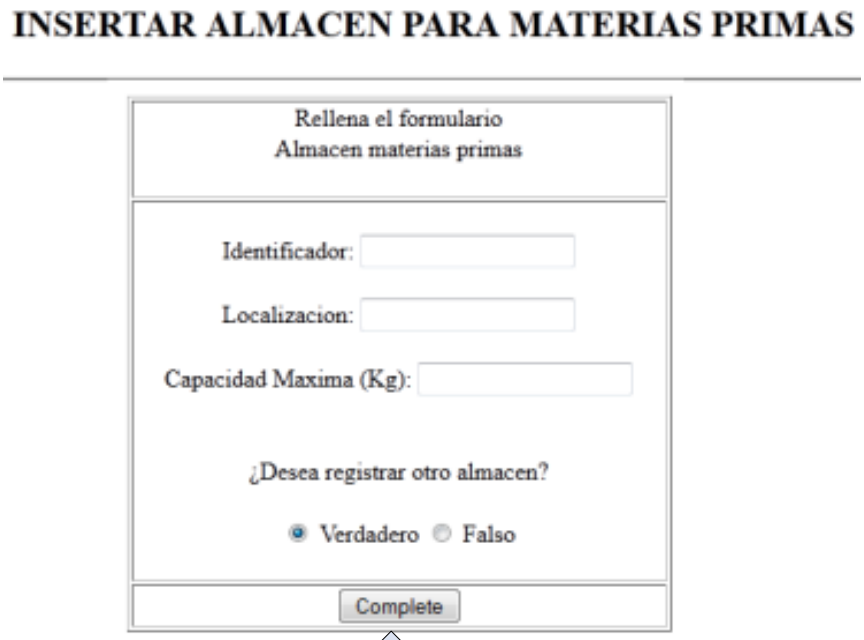



**Figura 35: Representación sobre la interacción del administrador en la configuración del sistema.**

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:

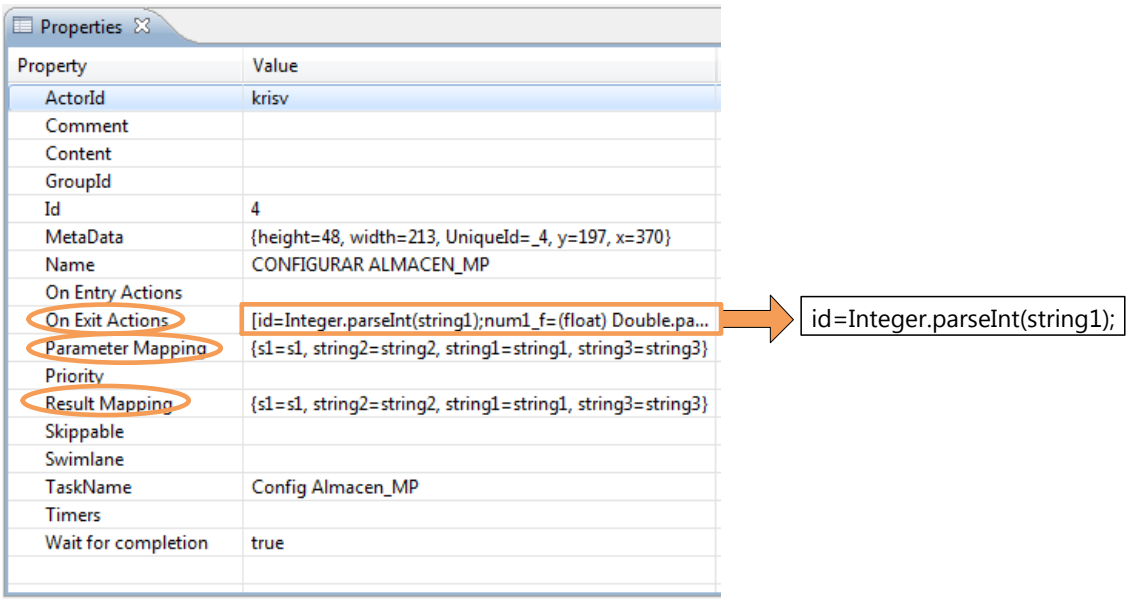



**Figura 37: Propiedades principales de las tareas de usuario y de su interacción con las variables del proceso.**

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.

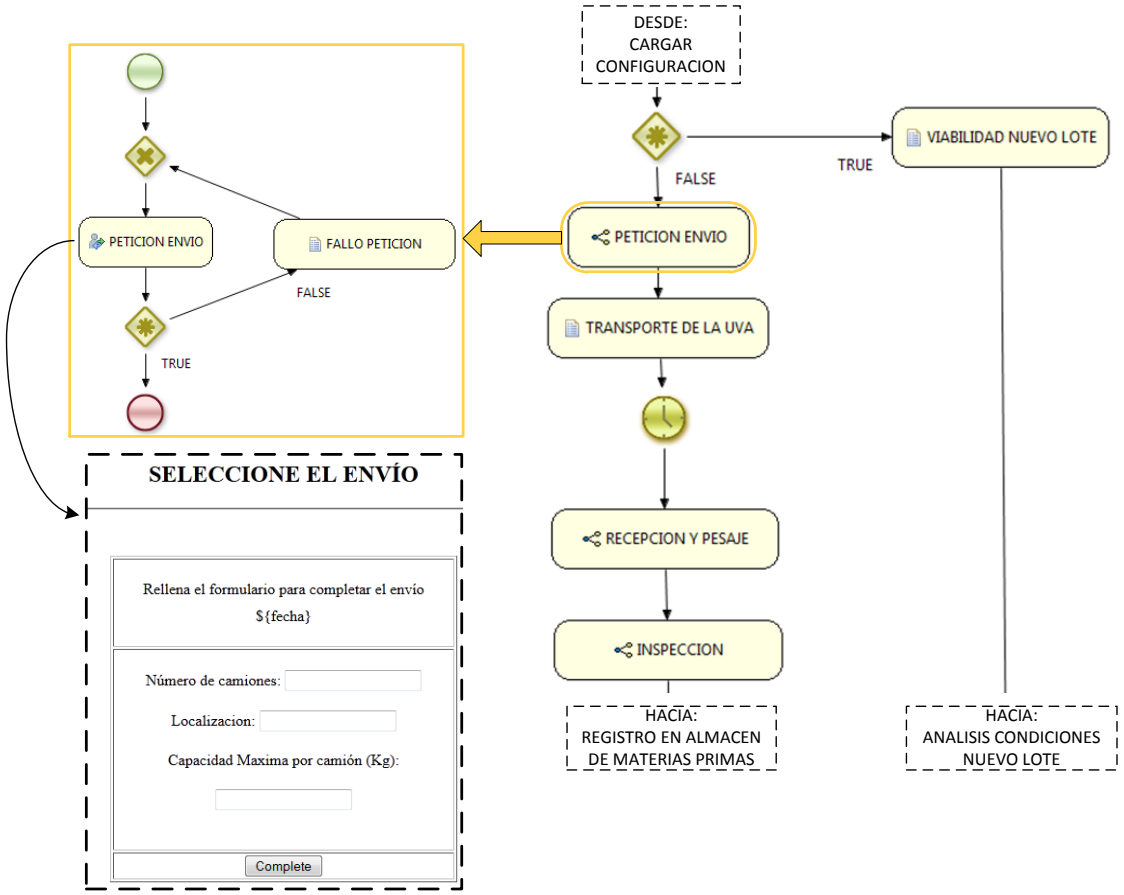



**Figura 38: Etapa inicial del proceso de elaboración de vino centrándose en la petición de envío**

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.

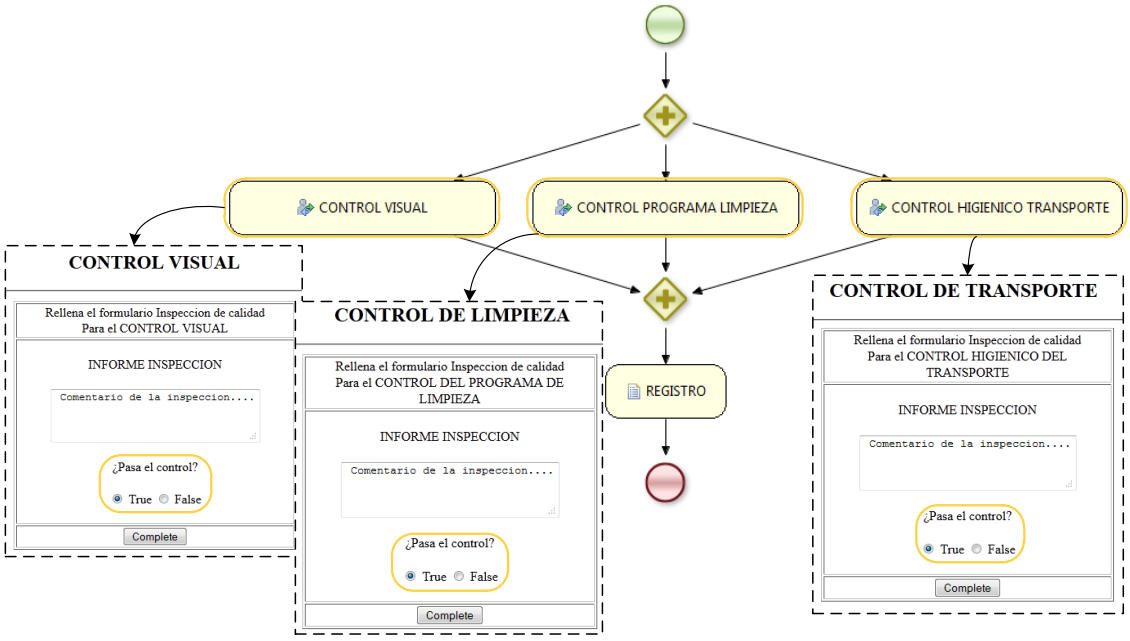



**Figura 39: Subproceso de inspección de materia prima y la interacción con el usuario asociada**

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.

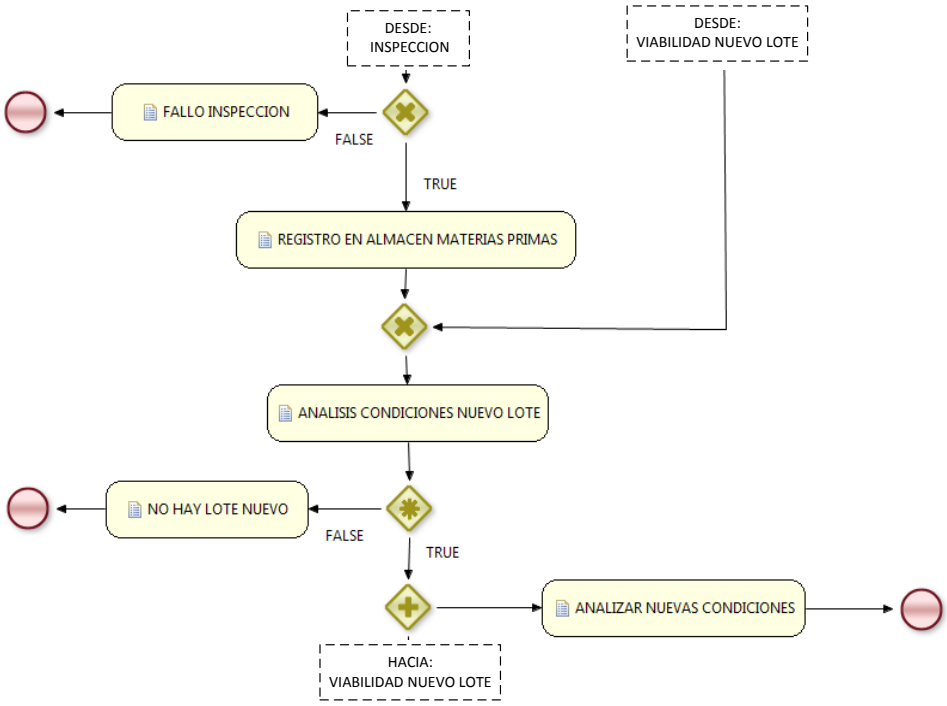



**Figura 40: Etapa de registro en almacén y evaluación de condiciones para un nuevo lote**

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.

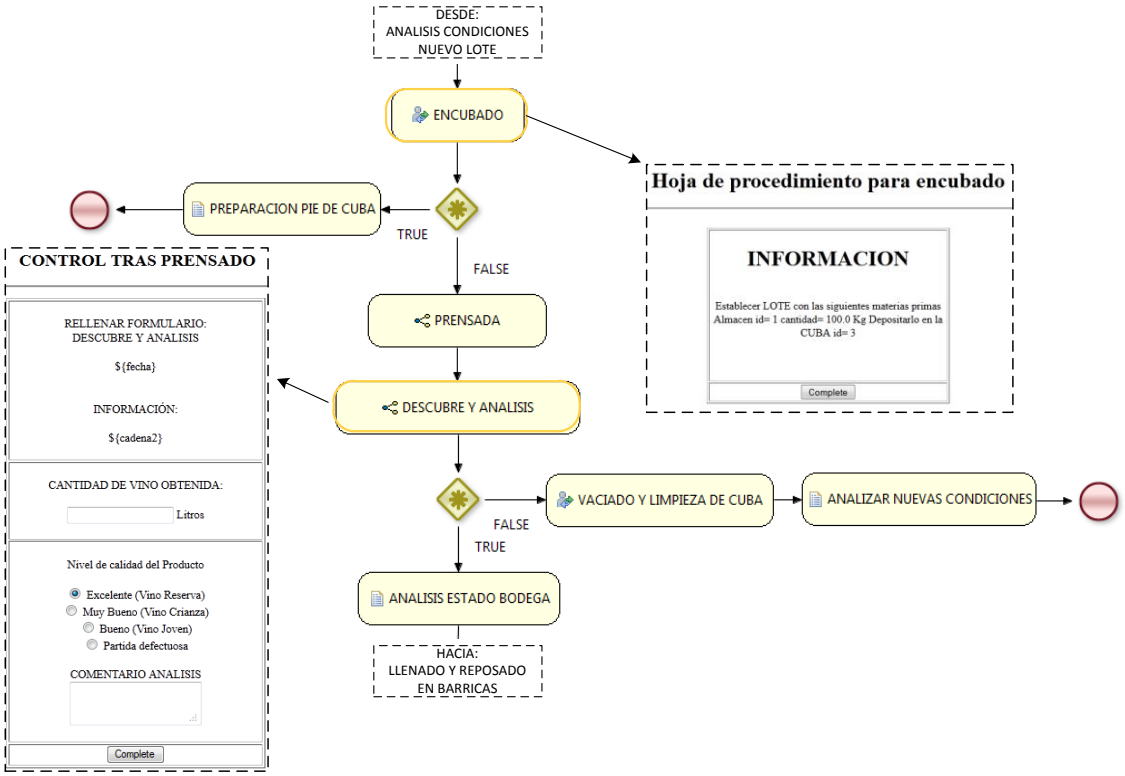



**Figura 41: Etapa de encubado y descubre del vino en elaboración**

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.

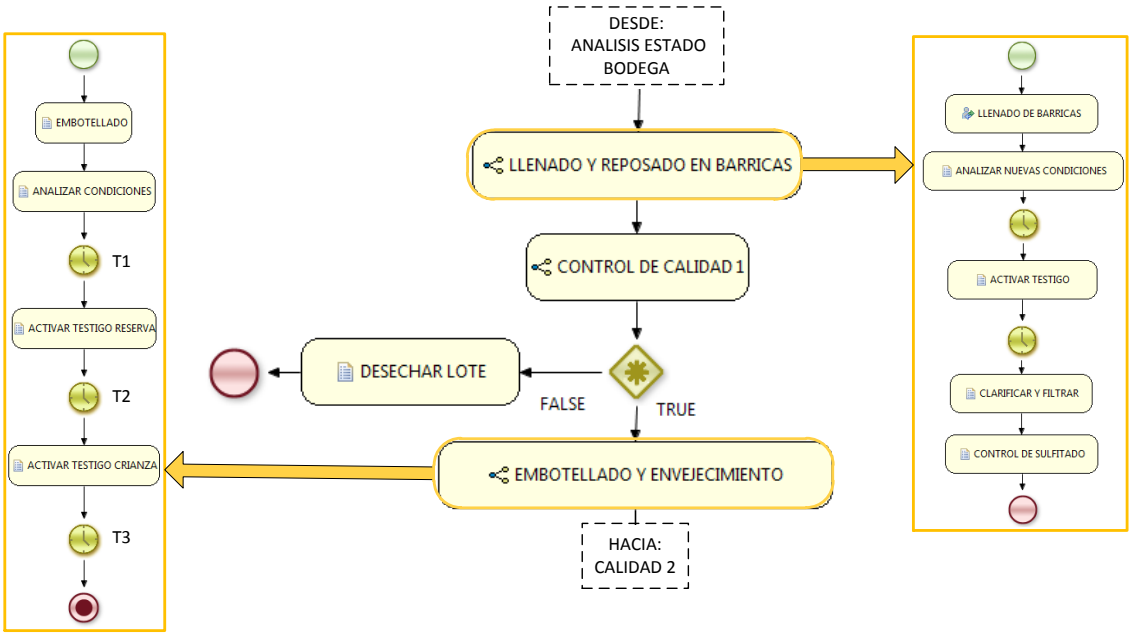



**Figura 42: Etapa para el llenado y el reposo en barrica así como el posterior embotellado y envejecimiento**

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.

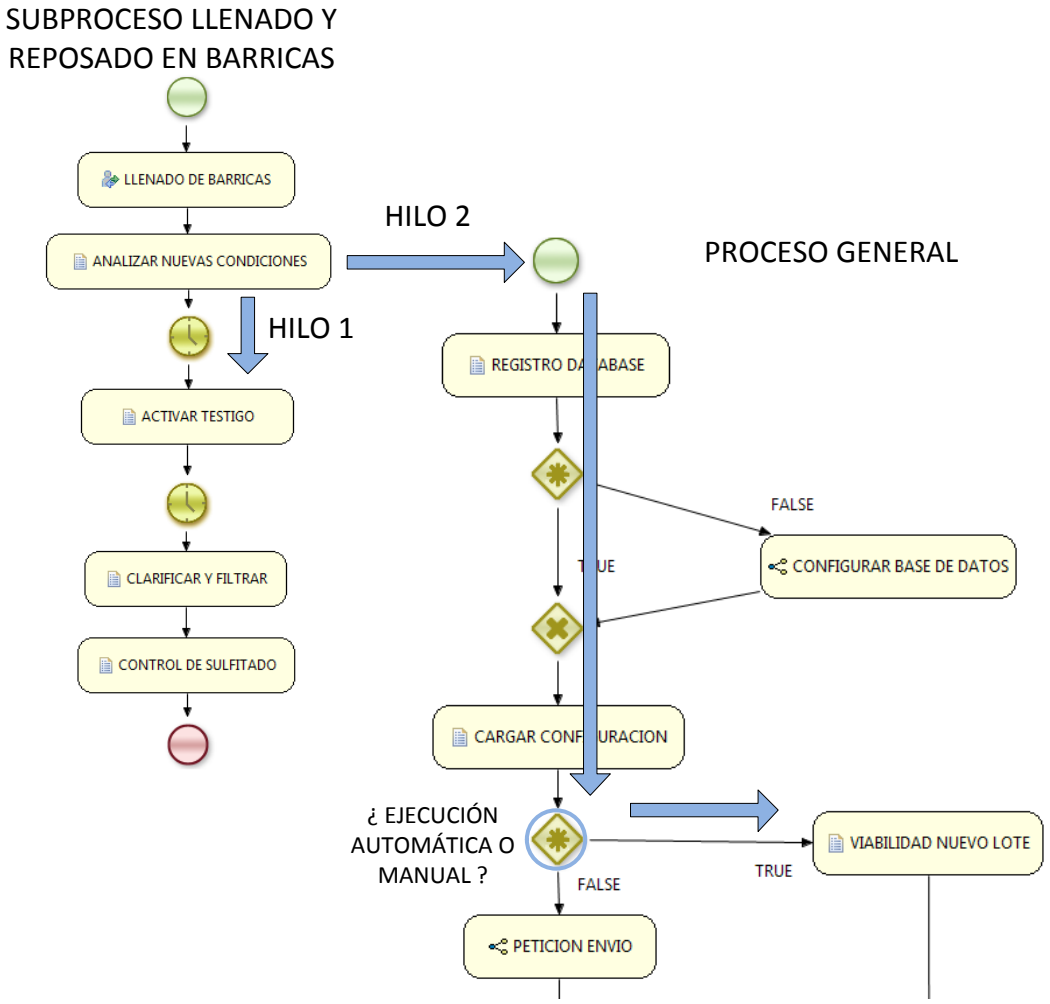



**Figura 43: Esquema conceptual sobre el lanzamiento de un nuevo hilo ante la liberación de recursos en la bodega**

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.

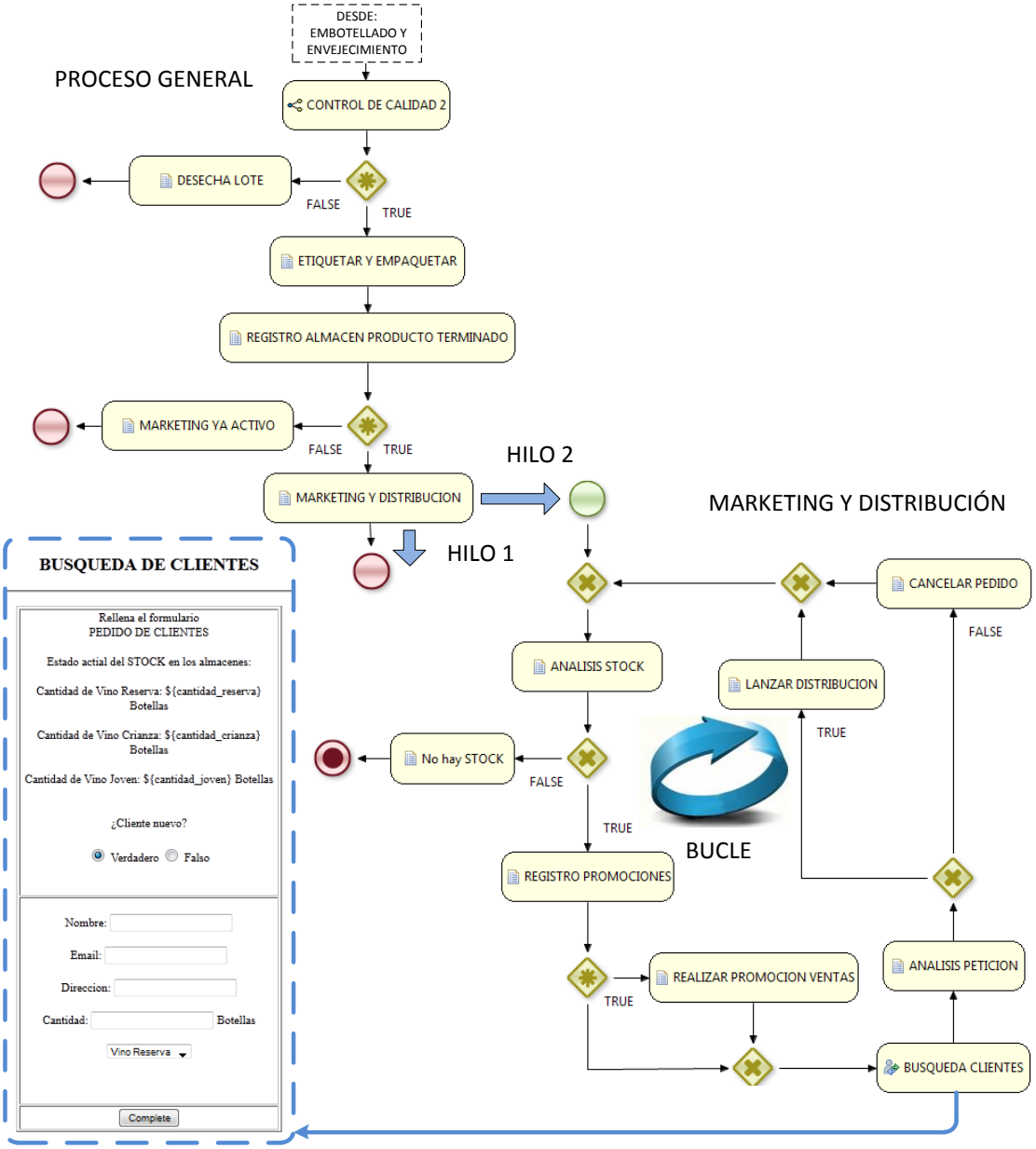



**Figura 44: Proceso final de marketing y distribución del producto elaborado**

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.